Ein Expertensystem als Experimentalsystem

Barbara Maier-Schicht, Günther Theiss & Theo Held

Bericht Nr. 87

Februar 1995

Arbeiten aus dem Sonderforschungsbereich 245
"Sprache und Situation", Heidelberg/Mannheim

Psychologisches Institut
der Universität Heidelberg
Hauptstr. 47-51
69117 Heidelberg

Die Arbeit entstand innerhalb des Teilprojektes A5 (Projektleiterin Prof. Dr. M. Wintermantel) des Sonderforschungsbereichs 245 "Sprache und Situation" an der Universität Heidelberg. Wir danken der Deutschen Forschungsgemeinschaft für die Förderung unserer Arbeiten.

ISSN 0941/990X

Inhaltsverzeichnis

Vorbemerkung

Dieses Papier entstand aus der Zusammenarbeit zwischen den Bereichen Psychologie, Informatik und Mathematik an den Universitäten Heidelberg und Karlsruhe im Sonderforschungsbereich 245 "Sprache und Situation", Projekt A5 "Darstellung technischer Abläufe in dialogischen Instruktionen".

Das erste Kapitel enthält eine kurze Einführung in das Gebiet ***Expertensysteme***. Anschließend wird die ***Expertensystemschale knoX*** vorgestellt. Diese Schale wurde verwendet, um ***das Expertensystem Destillation*** zu entwickeln, das im Projekt als Experimentalsystem verwendet wurde. Die Modellierung und Realisierung dieses Systems ist Inhalt des dritten Kapitels.

Eine Dokumentation des Systems und ein Handbuch erscheint ebenfalls in der Veröffentlichungsreihe des Sonderforschungsbereichs 245 (Held & Maier-Schicht, 1994).

1 Einführung in Expertensysteme

Expertensysteme stellen eine neue Technologie aus dem Bereich der *Künstlichen Intelligenz* dar, die es erlauben soll, bisher scheinbar kaum zu beherrschende Probleme zu lösen und die gefundene Lösung dem Anwender verständlich zu machen. Der Aufbau dieser Systeme, die im allgemeinen auf einer einheitlichen Architektur basieren, soll hier behandelt werden. Dabei werden die einzelnen Komponenten eines Expertensystems erklärt und ihre Funktion erläutert. Die Vorzüge dieser Programme, die auch als wissensbasierte Systeme bekannt sind, werden im Hinblick auf ihre möglichen Einsatzgebiete aufgezeigt. Hierbei ist die Repräsentation von Wissen eines der zentralen Probleme der KI-Forschung. Um der Forderung nach einer möglichst adäquaten Darstellung von Expertenwissen gerecht zu werden, wurden verschiedene Repräsentationsformalismen entwickelt. Die wichtigsten dieser Formalismen werden anhand von Beispielen erläutert. Zudem wird eine Einordnung der inzwischen zahlreich verfügbaren Werkzeuge für die in der Regel sehr zeit- und kostenaufwendige Entwicklung von Expertensystemen aufgezeigt.

1.1 Was versteht man unter einem Expertensystem?

In den 60er Jahren versuchten Softwareingenieure, allgemeine Methoden zu entwickeln, die den komplizierten Prozeß des menschlichen Denkens simulieren sollten. Obwohl sie an manchen Stellen interessante Fortschritte machten, blieb der große Erfolg versagt. Je mehr Probleme ein einzelnes Programm lösen konnte, um so schlechter waren die Lösungen der einzelnen Probleme.

Deshalb wurde ein anderer Weg versucht, um Computerprogramme *intelligent* zu machen. Man konzentrierte sich nun auf die Entwicklung von Methoden, die in spezialisierten Programmen eingesetzt wurden. Wieder brachte diese Strategie Erfolge, ohne aber den erhofften Durchbruch zu schaffen.

Dies gelang erst Mitte der 70er Jahre den Forschern als sie von der Erkenntnis ausgingen:

> *"Um ein Programm intelligent zu machen, muß es mit einer Menge erstklassigem spezifischem Wissen über ein Problemgebiet versehen werden."* (Waterman, 1986, S. 4)

Diese Erkenntnis führte zur Entwicklung von Computerprogrammen, die Experten in einem eingeschränkten Problemgebiet waren und die Vorläufer der heutigen *Expertensysteme* darstellten.

Was sind Expertensysteme ?

Wie so oft in jüngeren Wissenschaftsgebieten existiert auch hier noch keine einheitliche Definition des Expertensystembegriffes. Statt einer knappen und prägnanten Begriffsklärung muß man meist auf längere Beschreibungen der Eigenschaften von Expertensystemen zurückgreifen. Eine ausführliche Definition versuchen z.B. Brachman in dem Artikel *What are Expert*

Systems ? (Brachman, 1983), in dem die Anforderungen an Expertensysteme detailliert beschrieben werden.

Raulefs (Raulefs, 1982) definiert Expertensysteme als Systeme, die Tätigkeiten von menschlichen Experten unterstützen bzw. teilweise mechanisieren. Ein Experte ist dabei ein Spezialist für ein bestimmtes, eingegrenztes Problemgebiet, der auf diesem Gebiet Probleme lösen und die Lösung erklären kann. Hieraus ergeben sich die folgenden Anforderungen an einen Experten oder an ein System, das ihn unterstützen oder *'ersetzen'* soll:

- Der Experte soll aus Vorgaben präzise Problemstellungen formulieren können.
- Die Problemlösung muß korrekt und vollständig sein.
- Die Antworten müssen verständlich sein.
- Der Lösungsweg muß erklärt werden, damit eine Einschätzung der Verläßlichkeit der Antwort möglich ist.
- Der Experte erteilt Hilfe bei der Anwendung der Lösung.

Heutige Expertensysteme können bisher noch nicht allen diesen Anforderungen gerecht werden, weshalb sie oftmals nur zur Unterstützung von menschlichen Experten eingesetzt werden.

In der Literatur werden die Begriffe wissensbasiertes System und Expertensystem synonym benutzt. Dabei ist der Begriff des wissensbasierten Systems allgemeiner, während man mit dem Begriff des Expertensystems zum Ausdruck bringen möchte, daß das Wissen hier letztendlich vom Experten stammt.

1.2 Architektur von Expertensystemen

Im Laufe der Zeit haben sich bei Expertensystemen verschiedene Komponenten entwickelt, die inzwischen bei jedem Expertensystem vorkommen. Die verschiedenen Komponenten und ihre Beziehungen untereinander legen den inneren Aufbau eines solchen Systems fest und werden allgemein als seine Architektur bezeichnet. Die Architektur eines Expertensystems ist in Abb. 1 dargestellt.

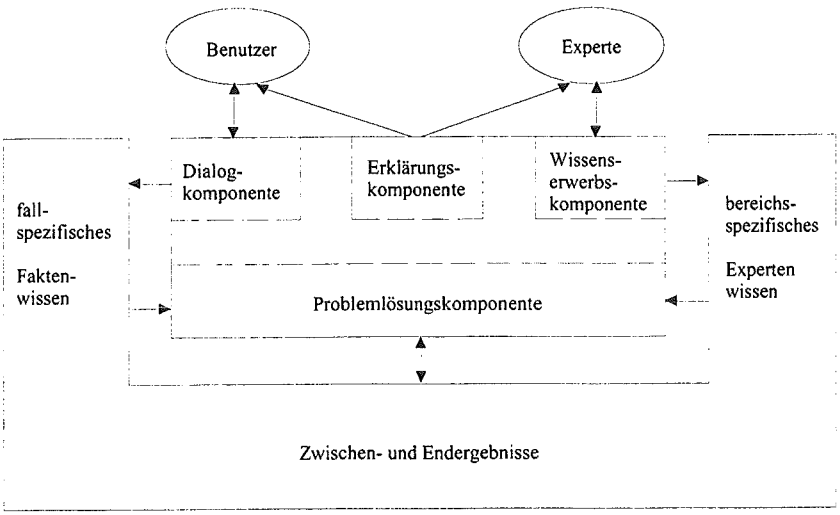


Abb. 1: Architektur eines Expertensystems nach Puppe (Puppe, 1988, S. 6)

Die beiden Hauptbestandteile sind die anwendungsspezifische Wissensbasis (außen) und das bereichsunabhängige Steuersystem (innen).

Die **Wissensbasis** enthält drei Arten von Wissen:

- ***bereichsspezifisches Expertenwissen***, das sich während einer Konsultation nicht ändert,
- ***fallspezifisches Faktenwissen***, das der Benutzer während einer Konsultation eingibt,
- ***Zwischen- und Endergebnisse***, die das System während einer Konsultation herleitet.

Das **Steuersystem** besteht aus folgenden Komponenten:

- Die ***Wissenserwerbskomponente*** ermöglicht dem Experten, neues Wissen in das System einzugeben und ersteres jederzeit zu ändern. Dazu können neben einfachen Texteditoren auch interaktive Wissensbasiseditoren eingesetzt werden, die bereits auf ein bestimmtes Anwendungsgebiet zugeschnitten sind.
- Die ***Dialogkomponente*** führt den Dialog mit dem Benutzer, der an seine sprachlichen Ausdrucksmöglichkeiten und Denkgewohnheiten angepaßt sein sollte. Diese Anforderung ist bei Expertensystemen besonders zu beachten, da ihr Erfolg oft entscheidend von der Qualität der Benutzeroberfläche abhängt.
- Die ***Problemlösungskomponente***, die auch Inferenzkomponente genannt wird, versucht, aus dem Wissen in der Wissensbasis schrittweise eine Lösung des gestellten Problems herzuleiten. In fast allen Anwendungsgebieten ist der Lösungsraum, ähnlich wie beim Schachspiel, viel zu groß, um alle Möglichkeiten systematisch zu generieren und überprüfen zu können. Um trotzdem eine Lösung des Problems finden zu können, erfolgt eine Einschränkung des Lösungsraums mit Hilfe des Expertenwissens, so daß immer nur die aussichtsreichsten Wege berücksichtigt werden.
- Die ***Erklärungskomponente*** muß das Systemverhalten für den Benutzer und den Experten jederzeit transparent und nachvollziehbar machen. Die Erklärungsfähigkeit ist eine Grundvoraussetzung für die Akzeptanz eines Expertensystems. Sie resultiert aus der Trennung von Wissen über das Problem und der Methode, wie das Problem gelöst werden soll. Der Benutzer verlangt für die Lösungsvorschläge des Systems, auf denen oftmals gravierende Entscheidungen basieren, eine rationale Erklärung der getroffenen Entscheidung, um diese zumindest teilweise nachvollziehen zu können. Die meisten Expertensysteme sind derzeit nur in der Lage, die Vorgehensweisen des Systems transparent zu machen. Sie können erklären, wie das System zu einer bestimmten Lösung gekommen ist und warum es eine bestimmte Aktion ausgeführt hat. Diese Erklärungen unterstützen im wesentlichen den Experten bzw. Wissensingenieur bei der Fehlersuche. Bei der Realisierung einer Erklärungskomponente dürfen grundsätzliche Eigenschaften, wie Verständlichkeit, Partnermodellierung, Prägnanz oder Dialoggedächtnis nicht außer acht gelassen werden. Die rudimentäre Erklärungsfähigkeit existierender Systeme ist jedoch schon ein gewaltiger Fortschritt im Vergleich zu konventionellen Programmen. Das Problem, gute Erklärungen zu produzieren, ist aber eher erkannt als schon gelöst.

1.3 Methoden der Wissensrepräsentation

Die Art der Darstellung von Wissen ist eine der wichtigsten Aspekte eines Expertensystems. Man bezeichnet die formale Sprache, in der das Expertenwissen in einem System dargestellt wird, als Wissensrepräsentation. Das Expertenwissen weist selbst in eng abgegrenzten Anwendungsgebieten unterschiedliche Strukturen auf. So verschiedene Arten von Wissen, wie

z.B. Beschreibung von Objekten, Beziehungen zwischen Objekten, typischen Situationen, Entscheidungsregeln und Vorgehensweisen, kommen praktisch in jedem Anwendungsgebiet vor und werden wie selbstverständlich von menschlichen Experten benutzt. Häufig ist dieses Wissen auch noch mit Unsicherheiten behaftet, unvollständig und hypothetisch. Für die verschiedenen Wissensarten wurden verschiedene Repräsentationsformen entwickelt, von denen die drei wichtigsten hier kurz erläutert werden sollen.

1.3.1 Logische Wissensrepräsentation

Die Logik ist die wohl am meisten untersuchte und am besten verstandene Art der Wissensrepräsentation. Da die Aussagenlogik nicht besonders mächtig[1] ist, verwendet man in der Regel meistens die Prädikatenlogik erster Stufe[2]. Sachverhalte werden hier durch Formeln dargestellt, die aus Konstanten, Variablen, Funktionen, Prädikaten, Quantoren und logischen Konnektoren aufgebaut sind.

Mit allgemeingültigen, logischen Ableitungsregeln lassen sich aus einer gegebenen Menge von Fakten neue Fakten herleiten.

Die beiden nachfolgenden Fakten beschreiben die Abfahrt und Ankunft eines Zuges an einem Ort zu einer bestimmten Zeit.

abfahrt(z1,heidelberg,10.00) und

ankunft(z1,karlsruhe,10.35)

Interpretation:
Der Zug z1 fährt in Heidelberg um 10.00 Uhr ab. Der Zug z1 kommt in Karlsruhe um 10.35 Uhr an.

Aus diesen Fakten kann nun mit Hilfe der *Formel*

$\forall$ZUG, $\forall$ORT1, $\forall$ORT2, $\forall$T1, $\forall$T2:

abfahrt(ZUG,ORT1,T1) &
ankunft(ZUG,ORT2,T2)
$\rightarrow$ verbindung(ZUG,ORT1,ORT2,T1,T2)

eine Zugverbindung zwischen zwei Orten abgeleitet werden.

Interpretation:

Wenn ein ZUG an einem ORT1 zur Zeit T1 abfährt und einen ORT2 zur Zeit T2 erreicht,
dann gibt es eine Zugverbindung zwischen dem ORT1 und dem ORT2 durch den ZUG, mit der Abfahrtszeit T1 am ORT1 und der Ankunftszeit T2 am ORT2.

Die Vorteile der logischen Darstellung sind die wohldefinierte Syntax und Semantik sowie die relativ einfachen Ableitungsregeln. Dennoch ist diese Darstellungsform für den Experten wenig geeignet, da die Übersichtlichkeit der Wissensbasis schnell verloren geht, Kontrollwissen [3] ausgeklammert ist und nur sicheres[4], monoton wachsendes[5] und zeitloses Wissen[6] dargestellt werden kann.

Durch die Einschränkung der Prädikatenlogik erster Stufe auf weniger mächtige Teilmengen (z.B. Hornklauseln) kann man sie auch als Programmiersprache verfügbar machen. Ein Beispiel hierfür ist die Programmiersprache PROLOG.

[1] Die Aussagenlogik beinhaltet Verknüpfungen ($\vee$, $\wedge$, $\neg$ und $\rightarrow$) und als Ableitungsregel den Modus Ponens.
[2] Hier wird die Aussagenlogik erweitert um Prädikate, Variablen und Quantoren ($\forall$, $\exists$).
[3] Z. B. Wissen darüber, wann welche Regel wie abgearbeitet wird.
[4] Jede Schlußfolgerung gilt als sicher. Es finden keine statistischen oder heuristischen Bewertungen statt.
[5] Das Wissen nimmt im Laufe der Zeit immer zu, im Gegensatz zu Mechanismen, die Schlußfolgerungen wieder rückgängig machen können.
[6] Fakten und neue Schlußfolgerungen haben keinen Zeitbezug und sind damit immer gültig.

1.3.2 Regelorientierte Wissensrepräsentation

Regeln sind wegen ihrer natürlichen Ausdrucksweise

wenn <Bedingung> *dann* <Aktion>

die am meisten verbreitete Darstellungsform. Sie eignen sich besonders gut zur Darstellung von Vorschriften über die Verarbeitung von Wissen (prozedurales Wissen). Ein Regelsystem besteht aus einer Regelmenge, einer Datenbasis und einem Regelinterpreter.

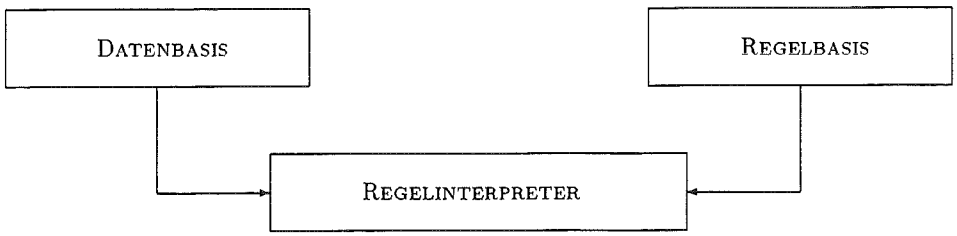


Abb. 2: Regelsystem

Das Wissen wird in Form von *Fakten* und *Regeln* dargestellt. Während *Fakten* Tatsachen darstellen, geben die *Regeln* an, wie *Fakten* miteinander verknüpft werden müssen. Der Regelinterpreter steuert den Ablauf. Er bestimmt, welche Regeln in einer jeweiligen Situation anwendbar sind, wählt eine davon aus, die dann *feuert*, d.h. deren Aktionsteil ausgeführt wird. Für die Auswertung von Regeln gibt es prinzipiell zwei Strategien:

- ***Vorwärtsverkettung*** *(forward chaining)*: Dabei werden alle Regeln angewendet, deren Bedingungen erfüllt sind.
- ***Rückwärtsverkettung*** *(backward chaining)*: Ausgehend von einem Ziel werden alle Regeln angewendet, die dieses Ziel in ihrem Aktionsteil haben. Sind Parameter der Bedingung noch unbekannt, so werden sie als neue Unterziele generiert und ebenso ermittelt oder vom Benutzer erfragt.

Beispiel einer Regel:

wenn fahrwunsch(ORT1,ORT2,T1,KLASSE,GATTUNG) und
zugverbindung(ZUG,ORT1,ORT2,T1,T2) und
zuggattung(ZUG) = GATTUNG und
entfernung(ORT1,ORT2) = ENTFERNUNG
dann fahrpreis = tarif(ENTFERNUNG,KLASSE) + zuschlag(ENTFERNUNG,GATTUNG).

Interpretation:

Wenn ein Fahrgast zu einer Zeit T1 von einem ORT1 zu einem ORT2 mit einem ZUG einer bestimmten GATTUNG und einer bestimmten KLASSE fahren möchte,
dann ergibt sich der Fahrpreis aus dem Tarifbeitrag und einem Zuschlag.

Während der Tarifbeitrag in Abhängigkeit von der Entfernung und der gewünschten Klasse ermittelt wird, ist der Zuschlag von der Entfernung und der Zuggattung abhängig.

Vorteile der regelorientierten Wissensdarstellung sind neben der natürlichen Ausdrucksweise, Modularität, Darstellung von unsicherem Wissen (Sicherheitsfaktoren) und die Trennung von Inhalt und Anwendung einer Regel. Bei dem Entwurf eines Regelsystems muß man lediglich angeben, was beim Vorliegen einer bestimmten Bedingung zu tun ist, und nicht wie

die Regeln abzuarbeiten sind. Dadurch wird das inkrementelle Erweitern der Wissensbasis wesentlich vereinfacht.

Der Nachteil der regelbasierten Darstellung ist der doch etwas schwer verständliche Kontrollfluß. Bei größeren Wissensbasen ist die Abarbeitung der Regeln nicht mehr so leicht nachvollziehbar.

1.3.3 Objektorientierte Wissensrepräsentation

Die objektorientierte Wissensdarstellung ist noch recht jung. Sie eignet sich besonders gut zur Beschreibung von Objekten und Zuständen (deklaratives Wissen). Das Wissen wird hierbei in Form von Frames, Instanzen und Behaviour dargestellt.

- ***Frames*** beschreiben Klassen von Objekten dadurch, daß sie relevante Attribute der Objekte festlegen. Diese Attribute werden *Slots* genannt. Für die Slots eines Frames können nun weitere Angaben gemacht werden. So kann man z.B. angeben, welche Werte ein Slot annehmen darf, wie der Wert für einen Slot zu ermitteln ist, mit welcher Frage der Benutzer nach dem Wert befragt werden soll oder welche Aktionen auszuführen sind, wenn ein Zugriff auf den Slot erfolgt. Diese Aktionen werden als *Behaviour* definiert.
- ***Instanzen*** stellen konkrete Objekte einer Klasse dar. Sie haben alle Eigenschaften der Klasse, der sie angehören.
- ***Behaviour*** sind Prozeduren, die benutzer-, zeit- und/oder ereignisgesteuert aktiviert werden. Darin werden in der Regel prozedurale Abläufe implementiert.

In einem objektorientierten System kann man Klassen zu einem Netzwerk verbinden. In einem solchen Netzwerk werden Eigenschaften vererbt. Jede Instanz eines Frames besitzt nicht nur die Eigenschaften seiner Klasse, sondern erbt auch die Eigenschaften ihrer Vorgänger, deren Vorgänger usw. Dabei überschreiben Informationen der spezielleren Klasse die Informationen der Oberklasse. Der größte Vorteil der Vererbungshierarchie liegt in der effizienten Speicherung von Wissen, da allgemeinstes Wissen oben in der Hierarchie angegeben wird.

In Abb. 3 ist die Klasse *Zug* durch die Attribute *Name, Nummer, Gattung* und *Zuschlag* gekennzeichnet. Rechts neben den Attributen sind die Datentypen angegeben, die ihren Wertebereich festlegen. Als konkretes Objekt dieser Klasse ist der Zug mit dem Namen *"Heinrich der Löwe"*, der Nummer *522*, der Gattung *IC* und dem Zuschlag *6.00* definiert.

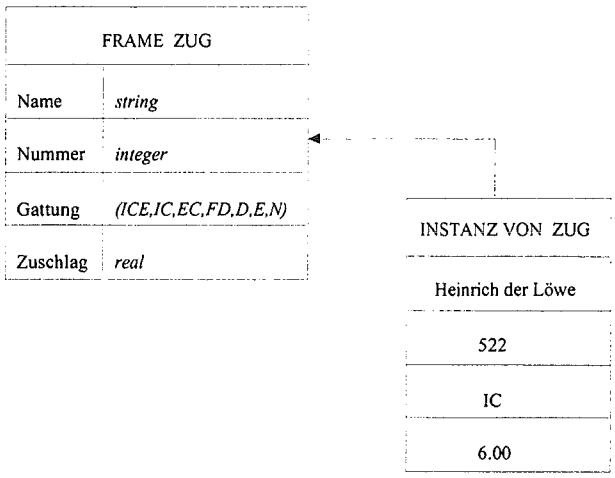


Abb. 3: Beispiel für ein Frame und eine Instanz

Abb. 4 zeigt eine Hierarchiebildung bei Zügen. Ein Zug gehört je nach Zuschlag in eine der drei Zugtyp-Klassen. Darin wird dann jeweils noch nach Zuggattung unterschieden. Allgemeine Informationen, die jeden Zug charakterisieren (z.B. Zugnummer), werden oben in der Hierarchie angegeben, und spezielle Informationen in den entsprechenden Unterklassen (z.B. Zuschlag in der Zugtyp-Klasse). Die allgemeinen Informationen sind, sofern sie nicht überschrieben wurden, auch in den untersten Klassen vorhanden.

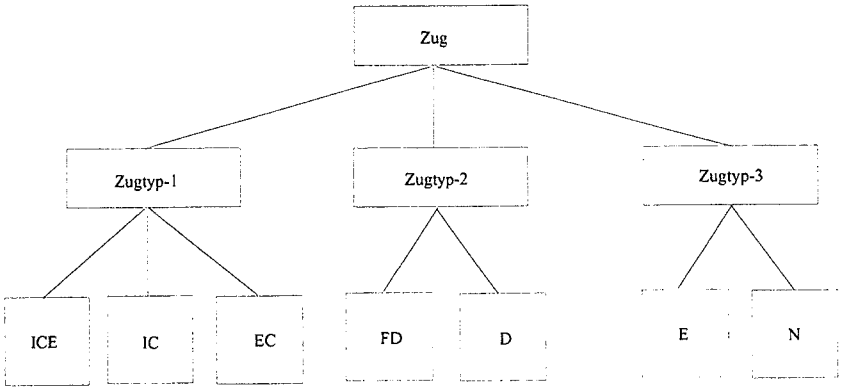


Abb. 4: Beispiel für die Klassenbildung

1.3.4 Hybride Wissensrepräsentation

Die klassischen Repräsentationsformen Logik, Regeln und Frames sind einzeln nicht in der Lage, die verschiedenen Bereiche des Wissens (deklarativ, prozedural) in einer natürlichen, für den Menschen übersichtlichen Weise darzustellen. Jeder dieser Formalismen hat seine Stärken in dem einen oder anderen Bereich. Erst eine *hybride* Repräsentationsform, die mehrere Darstellungsmöglichkeiten zuläßt, gestattet eine adäquate Erfassung von Expertenwissen. Obwohl es dafür keinen theoretisch oder technisch zwingenden Grund gibt, da alle Formalismen *gleichmächtig* sind, wird die Natürlichkeit der Darstellung bei einer hybriden Repräsentationsform erhöht. Dadurch wird auch verhindert, daß das Wissen des Experten in eine starre, fest vorgegebene Form gepreßt werden muß und dieser seine eigenen Arbeitsmittel und Vorgehensweisen beim Problemlösen im Expertensystem nicht wiederfindet.

Für die hybride Wissensrepräsentation sind Mechanismen vorzusehen, mit deren Hilfe die Konsistenz der Wissensbasis sichergestellt werden kann, und die helfen, Widersprüche zu erkennen.

1.3.5 Problembereiche der Wissensrepräsentation

Mit den hier vorgestellten Repräsentationsformen und dazugehörigen Verarbeitungsmechanismen lassen sich nicht alle Probleme darstellen und bearbeiten. So erfordert z.B. die Darstellung und Verarbeitung von zeitlichem und räumlichem Wissen spezielle Techniken, auf die jedoch hier nicht näher eingegangen wird. Obwohl auch dafür bereits Einzellösungen vorliegen, sind diese Gebiete noch größtenteils Gegenstand der Forschung. Gleiches gilt für die Behandlung von unsicherem und unvollständigem Wissen.

1.4 Einsatzgebiete von Expertensystemen

Die Einsatzmöglichkeiten der Expertensysteme können im Augenblick sicher noch nicht komplett abgeschätzt werden. Zum Teil werden viele Anwendungen als lösbar angesehen, die erst noch beweisen müssen, daß sie mit Expertensystemen machbar sind, zum anderen werden jedoch auch viele Aufgaben, die für Expertensysteme geeignet wären, noch nicht auf ihre Eignung hin untersucht. Allgemein ist jedoch zu sagen, daß Expertensysteme vor allem dort eingesetzt werden, wo das Wissen unstrukturiert ist und keine Algorithmen vorhanden sind (*diffuse* Gebiete) oder wenn die bekannten Algorithmen zu komplex und zu aufwendig sind.

Ein Anwendungsgebiet eignet sich vor allem dann für den Einsatz von Expertensystemen, wenn einige, der in der folgenden Aufzählung genannten Punkte, auf das Problem zutreffen. Dabei sollten

- der Problembereich nicht zu groß,
- die Lösbarkeit des Problems durch den Experten auf hohem Niveau notwendig,
- geringe Schwierigkeiten bei der Datenerfassung vorhanden,
- das Problem relativ statisch,
- der Experte verfügbar und kooperativ,
- praktischer Nutzen zumindest zu erwarten

sein. Sollten einige dieser Punkte auf ein Problemgebiet zutreffen, so kann eine Lösung durch ein Expertensystem sicher in Erwägung gezogen werden.

Bereits bekannte Anwendungsgebiete für Expertensysteme sollen in den folgenden Abschnitten kurz erläutert werden.

Diagnosesysteme

Die Diagnoseproblematik, die in jeglicher Art der Fehlersuche auftritt, zeigt sich sowohl aus Herstellersicht als auch aus Anwendersicht.

Der Hersteller muß für seine Produkte eine hohe Verfügbarkeit garantieren, da er oftmals bei Ausfällen Konventionalstrafen zu zahlen hat. Der Kundendienst, der zur Aufrechterhaltung dieser hohen Verfügbarkeit bereitgestellt werden muß, bringt erhebliche Kosten mit sich. Zudem sind die verfügbaren Service-Techniker, langfristig als "Flaschenhals" anzusehen, da diese Spezialisten teuer sind und nicht in ausreichendem Umfang zur Verfügung stehen. Ein weiterer Punkt wird sein, daß mit der immer weiter fortschreitenden Ausweitung des Marktes die Sicherstellung eines weltweiten Services für viele Hersteller zu einem großen Problem wird.

Eine andere Sicht auf diese Problematik hat sicher der Anwender, der, um eine genaue Terminierung seiner Produktion vornehmen zu können, ebenfalls eine hohe Ausfallsicherheit fordern wird. Durch den Ausfall von Maschinen werden ihm Deckungsbeiträge entgehen, die er bei der Finanzierung der Maschine bereits eingeplant hat. Um Fehler schnell beheben zu können, werden oftmals ungeübte Maschinenbediener dazu eingesetzt, den Fehler zu beseitigen, was oftmals zu weitaus größeren Störungen führen kann, als der eigentliche Fehler selbst. Um gegenüber seinen Kunden nicht in Verzug zu geraten, muß der Anwender also immer größere Kapazitäten und Zwischenläger einplanen, um auf einen eventuellen Störungsfall flexibel reagieren zu können.

Ein Konzept zur Lösung dieser Problematik wäre es, eine Telediagnose mit Expertensystemunterstützung einzusetzen, bei der Störungsdaten zentral beim Hersteller registriert werden, um eventuell auftretende Probleme einer gewissen Produktreihe bekannt zu machen und dem Maschinenbediener vor Ort ein Diagnose-Expertensystem zur Hand zu stellen, das die Fehlerursache analysiert und Vorschläge zur Fehlerbeseitigung gibt.

Planungssysteme

Aufgrund der immer größeren Produktvielfalt und der gewünschten niedrigen Zwischenlagerkapazitäten ist immer mehr eine flexible Fertigung notwendig. Konventionelle Planungswerkzeuge (wie Entscheidungstabellenprogramme) ermöglichen diese Planung, jedoch nur bis zu einem gewissen Grad. Das Ziel sollte deshalb sein, mit Hilfe von Expertensystemen diese Schranken, die die existierenden Programme bieten, zu überwinden und die Lösungsmöglichkeiten zu erhöhen.

Konfigurationssysteme

Der Bereich der Konfigurationssysteme beschäftigt sich grundsätzlich mit zwei verschiedenen Aufgabenbereichen. Hier ist zum einen die *Anlagenneukonfigurierung*, die sich mit der Planung einer neuen Anlage beschäftigt, und zum anderen die *Anlagenumkonfigurierung*, die eine bereits existierende Anlage im Betrieb in einem optimalen Zustand zu halten versucht, zu sehen.

Aufgrund der zunehmenden Flexibilität in der Produktion nimmt die Anlagenvielfalt der Hersteller immer mehr zu. Zudem sind gewisse Kombinationsmöglichkeiten nicht zulässig, so daß es oftmals schwierig ist, für den Kunden eine optimale Konfiguration seiner Anlage zu erstellen. Um wettbewerbsfähig zu bleiben, ist es jedoch notwendig, die Angebote relativ schnell zu erstellen. Hierbei herrscht meist eine Informationsdifferenz zwischen den Abteilungen Vertrieb und Technik, denn das Wissen über zulässige Konfigurationen ist zwar bei den Technikern vorhanden, die Vertriebsleute jedoch, die meist auch noch dezentral organisiert und angesiedelt sind, können dem Kunden keine Auskunft über die mögliche Konfiguration geben. Hier können Expertensysteme den Vertriebsberater unterstützen, da sie jeweils das aktuelle Wissen über freigegebene Konfigurationen haben sollten.

Im Rahmen der Umkonfigurierung der Anlagen, die ja heute nicht nur für die Produktion eines Artikels gedacht sind, ist es oft notwendig, umfangreiche Erfahrungsdaten zu berücksichtigen und basierend auf ihnen das optimale Programm für die neue Betriebsphase zu erstellen. Die hierbei anfallenden Daten sind meist zu umfangreich, um vom Experten in vollem Maße berücksichtigt werden zu können. Dies ist eine Aufgabe für ein Expertensystem, das diese Datenmenge durchaus verarbeiten kann.

Beratungssysteme

Als Beratungssysteme bezeichnet man entscheidungsunterstützende Systeme, die einen Kunden bei der Wahl einer zu treffenden Entscheidung beraten sollen. Ein bekanntes Beispiel hierfür sind die Anlageberatungssysteme, die bei Banken entwickelt werden, oder Fahrplanauskunftssysteme, die einen zeit- oder kostenminimalen Fahrweg vorschlagen.

Interpretationssysteme

Interpretationssysteme sollen aus gegebenen Informationen neue Erkenntnisse ziehen und dabei aus einer gegebenen großen Menge von Daten eine möglichst benutzerrelevante Abstraktion der gewünschten Information gewinnen. Ein Beispiel hierfür wären Verkehrsvorhersagen für Stauprognosen u.a.

Überwachungssysteme

Überwachungssysteme übernehmen die Kontrolle sicherheitsrelevanter Aktionen, die durch Anlagenbediener durchgeführt werden sollen. Es ist dabei daran gedacht, Anlagenbedienungen zu überwachen, um bereits vor der Aktivierung einer Aktion, die zu einem sicherheitsrelevanten Problem führen kann, dieselbe zu verbieten und dem Bediener einen Hinweis auf den möglicherweise folgenden Fehler zu geben.

Tutoringsysteme

Durch Tutoringsysteme wird das Wissen erfahrener Experten Berufsanfängern oder sonstigen Personen, die sich in ein Fachgebiet einarbeiten möchten, zur Verfügung gestellt. Das Expertensystem unterstützt hierbei Lernende beim Einüben gewisser Tätigkeiten. Es eignet sich insbesondere auch zur Sicherung des betrieblichen Know-how über ein gewisses Problemgebiet in einer formalen Form.

Der Versuch, Lernen und Lehren durch den Einsatz von Maschinen zu unterstützen, ist nicht neu und kann bis in die 20er Jahre zurückverfolgt werden, als die ersten Lehrmaschinen entwickelt wurden. Während sich anfangs das Hauptinteresse nur auf die Maschine (Hardware) richtete, wurde in den 50er Jahren die Maschinensteuerung, d.h. das auf der Maschine ablaufende Programm (Software), zunehmend in den Mittelpunkt gestellt. Es fand also eine Schwerpunktsverlagerung in der Betrachtung von Lehrsystemen weg von der Lehrmaschine und hin zum Lehrprogramm statt. Die 60er und 70er Jahre waren dann durch den Einsatz von Computern zur Steuerung der Lehrmaschinen geprägt, ohne dabei größere Erfolge zu erzielen. Erst durch die zunehmende Leistungsfähigkeit der Microcomputer, die außerdem auch immer preisgünstiger wurden, änderte sich die Situation in den 80er Jahren, und es fand ein Wechsel vom kostspieligen Zentralrechner zum preisgünstigen und damit vielseitig und dezentral einsetzbaren Mikrocomputer statt.

Das Programm der traditionellen Lehrmaschinen als auch der computerbasierten Lehrsysteme beruht auf einem Lehr-Algorithmus. Die präsentierten Lehrinhalte wurden in einer festen Abfolge von Programmschritten programmiert. Lehrstoff und Lehrmethode mußten vom Programmierer im voraus festgelegt werden und wurden im Algorithmus zusammen fest programmiert. Die Programme waren wenig transparent und schwer zu warten. Da sich die Lehrprogramme auch überwiegend nur auf die bloße Reproduktion von Wissen beschränkten und außerdem nicht die Fähigkeit besaßen, sich dem Lernenden anzupassen, stießen sie in der Praxis nicht auf die gewünschte Resonanz.

Mit dem Aufkommen der *Künstlichen Intelligenz* hat sich auch ein Wandel im Programmieren von Lehrsystemen eingestellt, der in der Trennung von Wissen (Lehrstoff) und Wissensverarbeitung (Lehrmethode), dem wohl bedeutendsten Merkmal wissensbasierter Systeme, zum Ausdruck kommt. Lehrsysteme, die mit Methoden der Künstlichen Intelligenz programmiert werden, werden allgemein als ***Intelligente Tutoringsysteme*** (ITS, engl. intelligent tutoring systems, dt. auch intelligente tutorielle Systeme, intelligente Tutorensysteme, wissensbasierte Lernsysteme, wissensbasierte Lehrsysteme) bezeichnet. Sie enthalten neben dem reinen Lehrstoff auch die Lehrmethoden, die auf den Lehrstoff angewendet werden. Durch diese Trennung können diese Lehrmethoden mit unterschiedlichem Lehrstoff verwendet werden.

Es sei an dieser Stelle nochmals darauf hingewiesen, daß die *Künstliche Intelligenz* die Programme *nicht intelligenter* macht ! Sie bietet lediglich neue Methoden an, mit deren Hilfe Programme so gestaltet werden können, daß sie einem Benutzer als intelligent erscheinen, d.h. z.B. Entscheidungen treffen können und sich an den Benutzer anpassen. In der Literatur wird vielfach der Eindruck erweckt, daß wissensbasierte Systeme intelligent sind, dazulernen, Entscheidungen treffen können, in der Lage sind, ihre Entscheidungen zu erklären und sich an den Benutzer anpassen, ohne daß darauf hingewiesen wird, daß dies so programmiert werden muß. Die Programme können immer nur so intelligent sein, wie sie programmiert wurden. Dabei spielt das verwendete Wissen über das Problemgebiet die entscheidende Rolle.

Dieses Wissen kann jedoch nicht von der Informatik bereitgestellt werden. Die Entwicklung von Tutoringsystemen stellt deshalb eine interdisziplinäre Aufgabe dar, an der Informatiker, Psychologen und Erziehungswissenschaftler gleichermaßen mitwirken müssen. Die Rolle des Informatikers darf dabei nicht überbewertet werden, wie es anfangs sehr oft der Fall war, da die Informatik keine Lehr- und Lernmethoden entwickelt, sondern lediglich Methoden und Werkzeuge zur Strukturierung, Formalisierung und Implementierung von Wissen zur Verfügung stellt. Die fachliche Kompetenz bei den anzuwendenden Lehr- und Lernmethoden liegt allein bei den Psychologen und Erziehungswissenschaftlern. Die Aufgabe des

Informatikers ist es, das Wissen zu strukturieren und für eine rechnerinterne Darstellung zu formalisieren sowie schließlich zu implementieren.

1.5 Entwicklung von Expertensystemen

Seit dem Bau der ersten Expertensysteme Mitte der 70er Jahre wurden eine Reihe von Vorgehensweisen und Werkzeugen entwickelt, um den Entwicklungsprozeß zu systematisieren und den Entwicklungsaufwand zu reduzieren. Trotz einiger Erfolge ist die Entwicklung von Expertensystemen immer noch sehr zeit- und kostenaufwendig. Das Hauptproblem dabei ist die Wissensakquisition, die von vielen Spezialisten als der "Flaschenhals" bei der Entwicklung von Expertensystemen bezeichnet wird (Karbach & Linster, 1990).

Obwohl sich die Forschung zu Beginn hauptsächlich auf die Entwicklung von Formalismen zur adäquaten Darstellung von Wissen auf einem Rechner konzentrierte, erkannte man doch relativ schnell, daß auch der Transfer des Wissens vom Experten in die Wissensbasis methodisch unterstützt werden muß. Gegenwärtig gibt es zwei grundsätzlich verschiedene Ansätze zur methodischen Entwicklung von Expertensystemen, die beide auf dem allgemeinen Phasenmodell des Knowledge Engineering aufbauen. Während die Praxis noch weitgehend vom Einsatz des Prototypings geprägt ist, gewinnen modellbasierte Ansätze zunehmend an Bedeutung.

1.5.1 Das Phasenmodell des Knowledge Engineering

Die Begriffe Knowledge Engineering, Wissensakquisition und Wissenserhebung werden in der Literatur nicht einheitlich verwendet. Die Definitionen, die den folgenden Betrachtungen zugrunde gelegt werden, stammen aus (Karbach & Linster, 1990).

> *"**Knowledge Engineering** ist der gesamte Erstellungs- und Wartungsprozeß eines wissensbasierten Systems, von ersten Vorstudien über die Machbarkeit des Systems, der Auswahl von geeigneten Werkzeugsystemen, über die Wissensakquisition bis hin zur Integration des Systems und zur späteren Wartung und eventuellen Erweiterung der Wissensbasis."* (S. 9)

Der Entwickler eines wissensbasierten Systems wird als *Knowledge Engineer* oder Wissensingenieur bezeichnet.

> *"Die **Wissensakquisition** ist die Erhebung des Wissens aus verschiedenen Wissensquellen, wie Experten, Büchern, Handbüchern oder Lexika, sowie die anschließende Umsetzung dieses Wissens in eine operationale Wissensbasis. Die Wissensakquisition beinhaltet auch die Wartung der Wissensbasis nach der Auslieferung an den Kunden."* (ebenda)

Der Entwicklungsprozeß von Expertensystemen wird analog zum System- und Software-Engineering in verschiedene Phasen unterteilt (s. Abb. 5).

Die ***Wissenserhebung*** ist der erste Schritt im Erstellungsprozeß eines Expertensystems. In dieser Phase wird das für die Problemlösung relevante Wissen aus der Literatur oder von Experten gesammelt und möglichst uninterpretiert dokumentiert. Dazu bedient man sich verschiedener Erhebungstechniken, wie z.B. Interviews, Beobachtungen des Experten beim Problemlösen und indirekten Techniken zur Erhebung von implizitem Wissen.

Das erhobene Wissen wird in der nächsten Phase, der ***Interpretationsphase***, analysiert, strukturiert und interpretiert. Die Interpretation des Wissens führt zu einem Modell der Expertise, das als *mentales konzeptuelles Modell* bezeichnet wird (Karbach & Linster, 1990).

Diesem Modell kommt eine große Bedeutung zu, da es die direkte Vorgabe für die Implementierung ist.

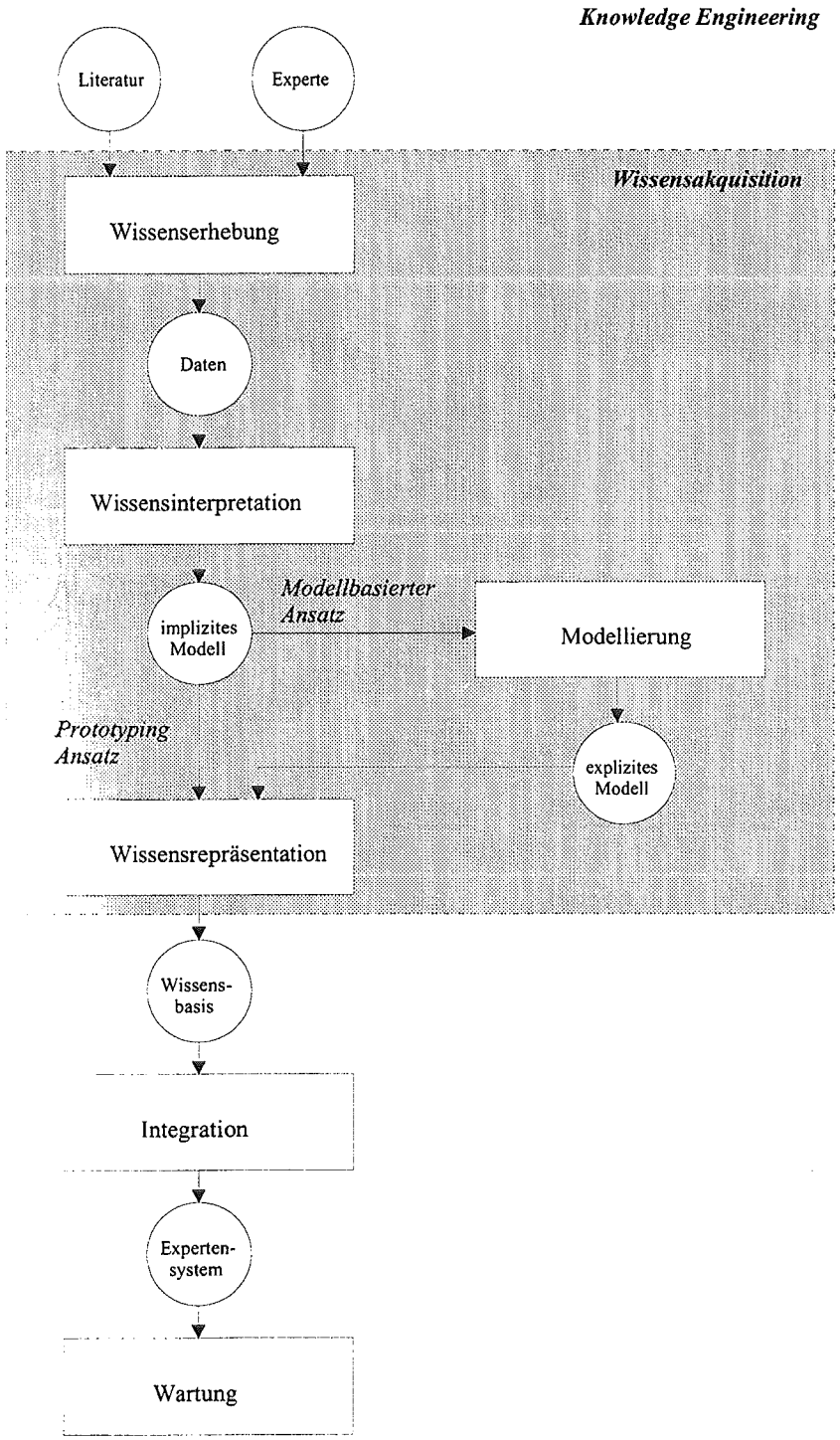


Abb. 5: Allgemeines Vorgehensmodell des Knowledge Engineering (Karbach & Linster, 1990, S. 10)

An dieser Stelle unterscheiden sich nun die beiden gegenwärtig verwendeten Vorgehensmodelle für die Erstellung wissensbasierter Systeme. Während im Prototyping-Ansatz das konzeptuelle Modell implizit bleibt und sofort auf die Wissensbasis abgebildet wird, schließt sich im modellbasierten Ansatz die Phase der ***Modellierung*** an, in der das konzeptuelle Modell explizit beschrieben wird.

Das analysierte und interpretierte Wissen, das als Modell im Prototyping-Ansatz implizit und im modellbasierten Ansatz explizit vorliegt, wird in der ***Wissensrepräsentationsphase*** auf die Repräsentationsformalismen der Wissensbasis abgebildet. Dabei ensteht die operationale Wissensbasis.

Das Expertensystem wird in der nächsten Phase in seine Umgebung ***integriert*** und während seines gesamten Lebenszyklus ***gewartet***.

Die einzelnen Phasen des Erstellungsprozesses lassen sich nicht eindeutig voneinander trennen. So sind z.B. die Erhebungs- und Interpretationsphase sehr stark miteinander verzahnt und beeinflussen sich gegenseitig. Die Phasen werden demzufolge auch nicht streng sequentiell, sondern meist iterativ durchlaufen.

1.5.2 Prototyping

Die Philosophie des Prototyping-Ansatzes ist es, nach einer Orientierungs- und Strukturierungsphase die erhobenen Daten direkt in eine operationale Wissensbasis abzubilden und dadurch möglichst schnell einen Prototypen des Expertensystems zu konstruieren. Der Prototyp wird dann sukzessive verfeinert und erweitert, bis er das gewünschte Verhalten aufweist. Dieses Vorgehen bezeichnet man auch als ***inkrementelle*** oder ***evolutionäre Entwicklung*** (Karbach & Linster, 1990).

Als klassischer Vertreter des Rapid-Prototyping-Ansatzes bei der Entwicklung von Expertensystemen gilt die Vorgehensweise nach Buchanan *et al* (Buchanan, 1983). Der Entwicklungsprozeß wird dabei in fünf Phasen unterteilt, zwischen denen zyklische Interaktionen stattfinden (s. Abb. 6).

Während der ***Identifikationsphase*** werden die wesentlichen Aspekte des Problembereichs identifiziert und analysiert sowie die am Projekt beteiligten Personen und die benötigten Ressourcen festgelegt.

In der Phase der ***Konzeption*** werden die Konzepte und Relationen des Problembereichs gesammelt und in der nächsten Phase, der ***Formalisierung***, auf die Repräsentationsformalismen des verwendeten Entwicklungswerkzeuges abgebildet. Aus dem formalisierten Wissen wird dann der eigentliche Prototyp des Expertensystems ***implementiert***.

In der ***Testphase*** wird überprüft, ob der Prototyp auch das gewünschte Verhalten aufweist. Sollte dies nicht der Fall sein, muß in frühere Phasen des Entwicklungsprozesses zurückgekehrt werden. Zunächst versucht man, durch Verfeinerung der Wissensbasis das gewünschte Verhalten zu erreichen, bevor man ein Redesign des formalisierten Wissens oder gar konzeptuelle Änderungen in der Identifikations- und Konzeptionsphase vornimmt.

Beim Rapid-Prototyping-Ansatz werden die einzelnen Aufgaben des Knowledge-Engineering - Analyse, Interpretation und Repräsentation des Wissens - sehr stark miteinander vermischt. Daraus ergeben sich auch die wesentlichen Nachteile dieses Ansatzes, ***keine explizite Modellierung der Expertise*** und ***fehlende Dokumentation des Projektfortschrittes***. Die Wartung eines so erstellten Expertensystems ist sehr schwierig und aufwendig.

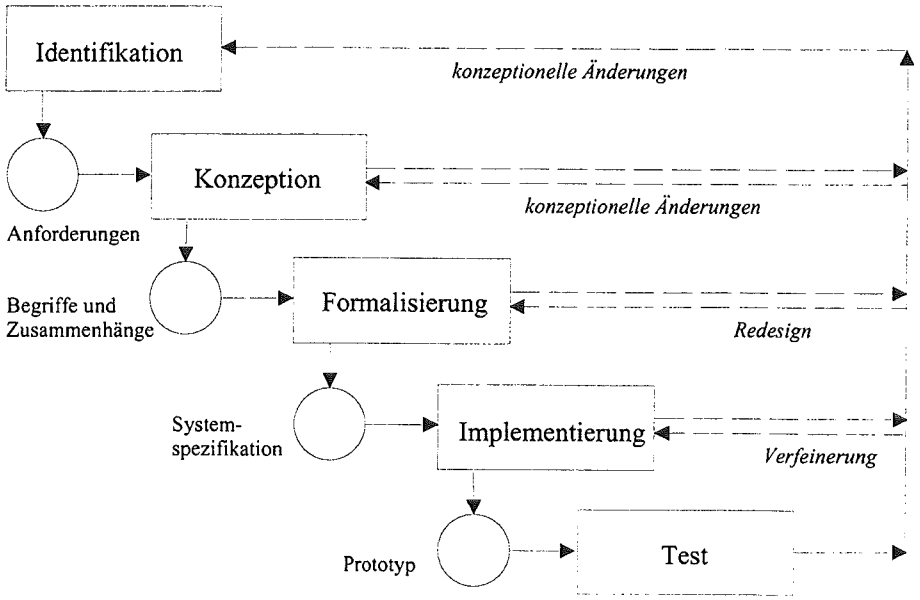


Abb. 6: Rapid-Prototyping-Vorgehensweise nach Buchanan et al. (Buchanan, 1983, S. 139)

Um diese Nachteile zu beheben, wurden von verschiedenen Autoren andere Vorgehensweisen vorgeschlagen, die aber alle auf der Rapid-Prototyping-Philosophie basieren. Sie unterscheiden sich von der Vorgehensweise von Buchanan *et al.* u.a. dadurch, daß sie explizite Dokumente für jede Entwicklungsphase fordern. Außerdem wird auch eine abstrakte Beschreibung des Problemlösungsprozesses gefordert, ohne jedoch anzugeben, wie dies erfolgen soll. Die Rolle des zu erstellenden Prototypen wird von den einzelnen Autoren unterschiedlich gesehen (Karbach & Linster, 1990).

1.5.3 Modellbasierte Entwicklung

Bei der modellbasierten Vorgehensweise wird die Entwicklung eines Expertensystems nicht mehr nur als ein reines Transferproblem von Expertenwissen in die Wissensbasis gesehen, sondern als eine Modellierung der Expertise. Das Ziel der modellbasierten Entwicklung ist die Beschreibung der Problemlösung auf einer höheren Abstraktionsebene. Dabei soll neben dem Problemlösungsverfahren auch das dazu benutzte Wissen beschrieben werden.

Die ersten modellbasierten Ansätze zur Entwicklung von Expertensystemen gehen auf die Einteilung der Anwendungsgebiete in Problemklassen und die Identifikation von typischen Aufgaben beim Lösen von Problemen zurück. Eine ausführliche Diskussion dieser Ansätze findet sich in (Puppe, 1990) und (Karbach & Linster, 1990).

Grundsätzlich lassen sich spezialisierte operationelle Modelle und allgemeine nicht-operationelle Modelle unterscheiden.

Erstere stellen dem Entwickler spezialisierte und bereits implementierte Problemlösungsverfahren mit einem festen Vokabular zur Formulierung des Wissens zur Verfügung, wie z.B. die ***Generic Tasks*** *(generischen Verfahren)* von Bylander und Chandrasekaran (Bylander & Chandrasekaran, 1987) . Sie sind durch ihre Spezialisierung wenig flexibel und lassen sich nur schwer miteinander kombinieren. Als eine allgemeine modellbasierte Entwicklungsmethode wissensbasierter Systeme wird gegenwärtig die im Rahmen eines ESPRIT-Projektes entwickelte ***KADS-Methodologie*** (**K**nowledge **A**cquisition and **D**ocumentation **S**tructuring) diskutiert (Burger & Mehling, 1992). Ihr liegt die Idee zugrunde, daß die Entwicklung wissensbasierter Systeme im wesentlichen eine ***Modellierungstätigkeit*** ist. Ein wissensbasiertes

System ist demzufolge nicht einfach ein mit Wissen gefüllter Behälter, sondern entsteht durch Analyse, Interpretation und strukturierte Abbildung eines Verhaltens aus der realen Welt auf die Elemente einer Wissensbasis.

Der Entwicklungsprozeß ist als eine Transformation von Modellen zu sehen, die Ergebnisse der einzelnen Entwicklungsschritte darstellen. Abbildung 7 zeigt die Transformation der verschiedenen Modelle. Im Gegensatz zum Wasserfallmodell des Software-Engineering werden die einzelnen Phasen nicht streng sequentiell durchlaufen, sondern in einer Folge von Zyklen, abhängig von den essentiellen Problemen und Risiken in den einzelnen Entwicklungsschritten (Schachter-Radig & Krickhahn, 1989).

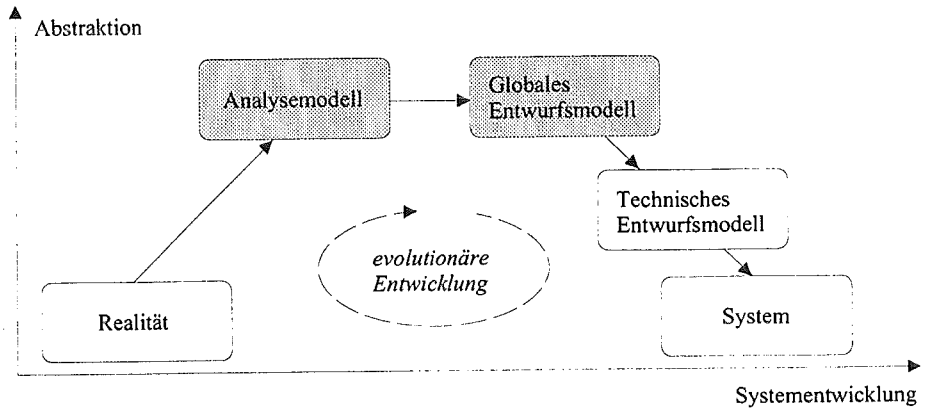


Abb. 7: Modellbasierte Vorgehensweise nach KADS (Burger & Mehling, 1992, S. 52)

Modellgestützer und modellgesteuerter Ansatz

Kennzeichnend für die KADS-Methodologie ist die modellgestützte und modellgesteuerte Vorgehensweise (Schachter-Radig & Krickhahn, 1989). Sie basiert auf der Benutzung von vordefinierten Repräsentationsschemata zur Modellierung des Wissens *(modellgestützter Aspekt)* und auf der Verwendung einer Bausteinbibliothek vorgefertigter Strukturen, wie z.B. Inferenzstrukturen für elementare Schlußfolgerungsverfahren und generische Interpretationsmodelle für häufig auftretende Problemlösungsaufgaben *(modellgesteuerter Aspekt)*. Diese Strukturen lassen sich für eine spezielle Anwendung beliebig verändern, erweitern und zu komplexeren Strukturen miteinander verknüpfen.

Der KADS-Lebenszyklus wird durch verschiedene Modelle charakterisiert. Dies sind das Analysemodell, das globale und das technische Entwurfsmodell.

Das Analysemodell

Während der Analysephase wird das zur Problemlösung benötigte Wissen aus dem Anwendungsgebiet erhoben, interpretiert und auf abstrakte Modelle abgebildet. Das Ergebnis der Analysephase bilden drei Modelle (Burger & Mehling, 1992):

- das *konzeptuelle Modell*, in dem das gesamte Wissen des Problembereichs modelliert wird,
- das *Kooperationsmodell*, das die Interaktionen zwischen System und seiner Umgebung beschreibt, und
- das *Anforderungsmodell*, das die funktionalen Anforderungen und Randbedingungen aus dem Einsatzbereich enthält, wie z.B. Hardware- und Softwareanforderungen, Mensch-Maschine-Schnittstelle usw.

Das konzeptuelle Modell ist in KADS am meisten entwickelt. Es sieht eine Beschreibung des in der Analysephase erarbeiteten Problemverständnisses in vier Ebenen vor, wobei jeweils unterschiedliche Aspekte von Wissen dargestellt werden. In der *Domänenschicht* werden alle für die Problemstellung relevanten Konzepte und Beziehungen zwischen ihnen dargestellt. Die Schlußfolgerungen, die auf der Basis dieser Konzepte und Beziehungen gemacht werden können, werden in der *Inferenzschicht* beschrieben. In der *Aufgabenschicht* werden die zu lösenden Aufgaben festgelegt und ihre Lösung aus den möglichen Inferenzen zusammengesetzt. Die *Strategieschicht* steuert den Problemlösungsprozeß, indem sie die einzelnen Aufgaben in der richtigen Reihenfolge ausführt und die Ausführung überwacht.

Das Analysemodell bildet die Grundlage für den Entwurf, der durch das globale und das technische Entwurfsmodell beschrieben wird.

Das globale Entwurfsmodell

Das globale Entwurfsmodell enthält eine implementierungsorientierte, aber noch völlig werkzeugunabhängige Systembeschreibung. Es unterteilt sich in

- ein *funktionales Modell*, das darstellt, was das System leistet und mit welchen Methoden dies erreicht wird, und in
- ein *physikalisches Modell*, in dem die einzelnen Methoden zu einer Systemarchitektur zusammengefügt werden.

Das funktionale Modell ist im wesentlichen eine Umsetzung des konzeptuellen Modells im Hinblick auf dessen Operationalisierung unter Einbeziehung der Randbedingungen, die im Anforderungsmodell beschrieben sind (Schachter-Radig & Krickhahn, 1989). Die einzelnen Aufgaben aus dem konzeptuellen Modell werden zunächst auf funktionale Blöcke abgebildet und anschließend im physikalischen Modell nach implementierungstechnischen Gesichtspunkten zu Modulen gruppiert.

Das technische Entwurfsmodell

Im technischen Entwurfsmodell wird nun festgelegt, wie die einzelnen Module mit dem ausgewählten Werkzeug realisiert werden. Es stellt somit die werkzeugabhängige Spezifikation der eigentlichen Implementierung dar.

Durch die Beschreibung der Problemlösung auf einer höheren Abstraktionsstufe und das schrittweise modellgestützte Vorgehen bleibt der Entwurfsprozeß in KADS transparent und jederzeit gut nachvollziehbar. Der Nachteil des noch verbleibenden Implementierungsaufwandes für die Realisierung des operationalen Systems ist demgegenüber als eher untergeordnet zu betrachten. Ein weiterer Vorteil von KADS, der ebenfalls aus der Verwendung von Modellen resultiert, ist die Wiederverwendbarkeit von Ergebnissen im gesamten Entwicklungsprozeß (Schachter-Radig & Krickhahn, 1989).

1.5.4 Integration von Prototyping und modellbasierter Entwicklung

Der große Vorteil des Prototyping-Ansatzes ist die relativ schnelle Entwicklung eines operationalen Systems, das man dem Anwender dann vorführen und ggf. durch Änderungen sukzessive an das gewünschte Verhalten anpassen kann. Demgegenüber stehen aber eine Reihe von Nachteilen, die sich insbesondere bei komplexen Problemstellungen in unübersichtlichen und kaum nachvollziehbaren Lösungen niederschlagen. Die Ursachen dafür liegen vor allem in

der starken Vermischung von Wissensanalyse und Wissenrepäsentation sowie der fehlenden Modellierung und Dokumentation der Expertise.

Prototyping sollte deshalb nicht als eigenständige Entwurfsmethode verstanden werden, sondern als ein *Hilfsmittel* zur Entwicklung wissensbasierter Systeme und stets in eine systematische Vorgehensweise integriert sein (Karbach & Linster, 1990).

Prototypen können in verschiedenen Phasen der Entwicklung eingesetzt werden. Zu Beginn der Modellierung können Prototypen z.B. dazu benutzt werden, die Anforderungen an das Expertensystem genau abzuklären und eine angemessene Benutzerschnittstelle zu entwerfen. Man bezeichnet diese Prototypen, die der funktionalen Spezifikation dienen, als ***explorative Prototypen.*** In der Modellierungsphase können ***experimentelle Prototypen*** eingesetzt werden, um verschiedene Teile des Modells zu validieren und alternative Lösungsmöglichkeiten durchzuspielen.

Die Integration des Prototypings in eine systematische Vorgehensweise als zusätzliches phasenunterstützendes Verfahren verspricht Erfolg, da sich die Vorteile beider Ansätze ergänzen. Ein Vorschlag zur Integration findet sich z.B. in (Karbach & Linster, 1990).

1.5.5 Arten des Wissenserwerbs

Die Wissensakquisition, auch Wissenserwerb genannt, kann praktisch auf verschiedene Arten erfolgen. Grundsätzlich unterscheidet man drei Arten des Wissenserwerbs:

- **indirekter Wissenserwerb**

  Beim indirekten Wissenserwerb befragt der Wissensingenieur *(Knowledge Engineer)* den Experten und baut dann nach seinem Verständnis der Problemlösung die Wissensbasis auf.

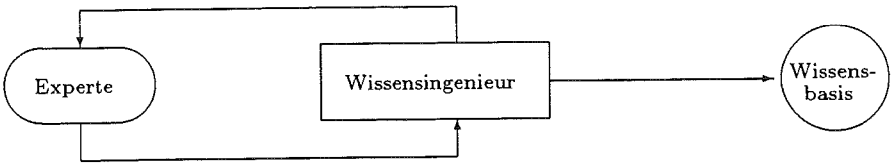


Abb. 8: Indirekter Wissenserwerb

- **direkter Wissenserwerb**

  Bei dieser Methode baut der Experte mit Hilfe einer komfortablen Wissenserwerbskomponente die Wissensbasis selbständig auf. Übertragungsverluste, die beim indirekten Wissenserwerb durchaus vorkommen können, werden dadurch eliminiert.

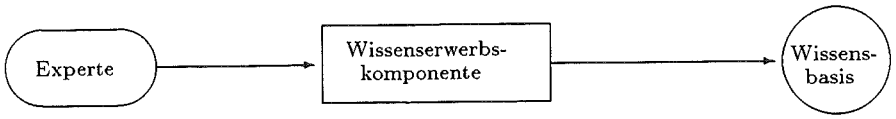


Abb. 9: Direkter Wissenserwerb

- **automatischer Wissenserwerb**

  Beim automatischen Wissenserwerb extrahiert das System sein Wissen aus verfügbarer Literatur oder Falldaten. Diese Methode ist zwar wünschenswert, derzeit jedoch noch nicht praxisreif.

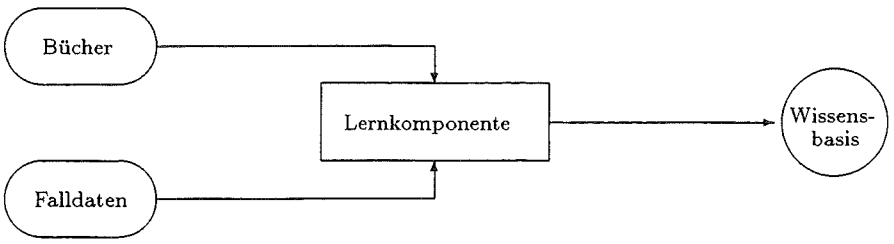


Abb. 10: Automatischer Wissenserwerb

Die meisten Expertensysteme werden zur Zeit noch nach der ersten Methode gebaut. Dabei kommt dem Wissensingenieur als *Dolmetscher* zwischen Experten und Expertensystem eine große Bedeutung zu. Da er einerseits hohe Kosten verursacht und zum anderen bei dieser Methode Übertragungsverluste nicht auszuschließen sind, ist der Übergang zum direkten Wissenserwerb anzustreben. Damit der Experte mit dem Expertensystem selbständig kommunizieren kann, ist der Einsatz einer komfortablen Wissenserwerbskomponente erforderlich.

1.5.6 Entwicklungswerkzeuge

Wie eingangs bereits erwähnt, ist die Entwicklung von Expertensystemen sehr zeitaufwendig und kostenintensiv. Der Entwicklungsaufwand läßt sich jedoch durch den Einsatz geeigneter Werkzeuge erheblich reduzieren. Es gibt nun eine Reihe von Kriterien, nach denen man die inzwischen doch sehr zahlreich vorhandenen Expertensystemwerkzeuge klassifizieren kann. Abb. 11 zeigt eine Einteilung nach dem Grad der Spezialisierung und der Verkürzung der Entwicklungszeit.

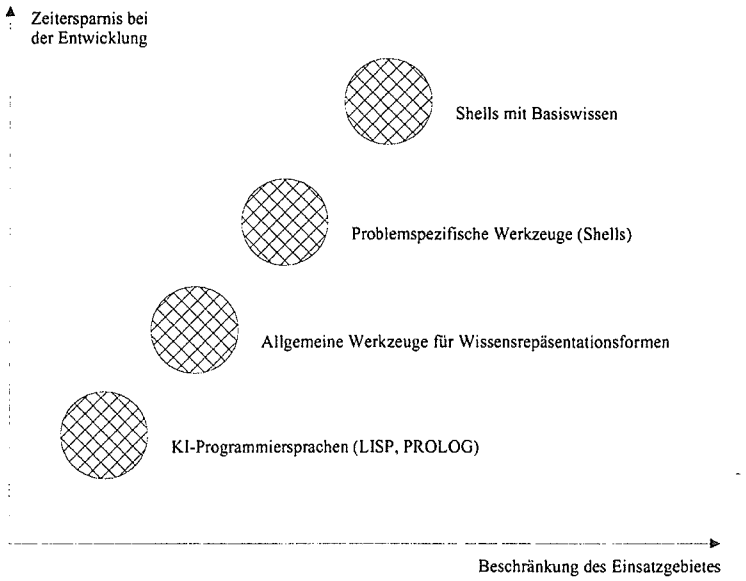


Abb. 11: Einteilung der Entwicklungswerkzeuge (Puppe, 1988, S. 7)

Die KI-Programmiersprachen, wie z.B. LISP oder PROLOG, sind flexible und allgemein einsetzbare Werkzeuge. Sie unterstützen die Problemlösung jedoch nur durch wenige vorgefertigte Konzepte und stellen somit hohe Anforderungen an den Wissensingenieur.

Allgemeine Werkzeuge für Wissensrepräsentationsformen stellen neben der Programmiersprache auch verschiedene Wissensrepräsentationstechniken (Regeln, Frames) und die dazugehörigen Verarbeitungsmechanismen (Vor- und Rückwärtsverkettung, Vererbung) zur Verfügung. Man unterscheidet hierbei zwischen einfachen und hybriden Werkzeugen, je nachdem ob eine oder mehrere Wissensrepräsentationsformen angeboten werden. Beispiele für derartige Werkzeuge sind KEE, BABYLON, NEXPERT-OBJECT, **knoX** etc.

Eine größere Unterstützung für den Wissensingenieur bieten problemspezifische Werkzeuge, auch Shells genannt. Sie beschränken sich auf eine bestimmte Problemklasse, wie z.B. Diagnose, Planung, Konfiguration und Simulation, und verfügen über typische Problemlösungsstrategien aus dem entsprechenden Anwendungsgebiet (z.B. MED2 für die Diagnose, s. Puppe, 1988).

Die größte Zeitersparnis beim Erstellen eines Expertensystems erreicht man durch die Verwendung von Shells, die bereits mit Basiswissen aus dem Anwendungsgebiet ausgestattet sind. Die Erstellung des Expertensystems reduziert sich dann auf die Ergänzung der Wissensbasis durch Spezialwissen.

Expertensystemschalen sollen es dem Experten ermöglichen, die Wissensbasis weitgehend selbständig aufbauen und ändern zu können. Er sollte sich im wesentlichen auf die logische Struktur und die abstrakte Lösung des Problems konzentrieren können und weniger auf den konkreten Ablauf des Programms auf dem Rechner.

1.6 Ausblick

Um den Einsatz von Expertensystemen kommerziell voranzutreiben, ist es notwendig, leistungsfähigerere Werkzeuge zu entwickeln, die den Entwickler, mehr als die bisherigen bei der Erstellung eines Expertensystems unterstützen. Hierzu muß vor allen Dingen der Flaschenhals der Expertensystementwicklung, der Wissenserwerb, methodisch besser unterstützt werden. Dabei wird sich die modellbasierte Vorgehensweise nach KADS durchsetzen, da der Entwicklungsprozeß nur durch eine explizite Modellierung der Problemlösung transparent wird und die Systeme wartbar und wiederverwendbar werden.

Bei der Wissensrepräsentation werden die hybriden Formen immer mehr an Boden gewinnen und mit der Zeit zum Standard werden. Nur mit ihnen kann dem Experten die Freiheit gegeben werden, sein Wissen in einer ihm adäquaten Weise in ein System einzubringen.

Aus der Sicht der Hardware wird die Parallelverarbeitung sicherlich Einzug in das Gebiet der Expertensysteme nehmen, da durch sie die Effizienzprobleme der heutigen Expertensysteme angegangen werden können.

2 Die hybride Expertensystemschale knoX

Die hybride Expertensystemschale **knoX** wurde konzipiert, um den hohen Entwicklungsaufwand bei der Erstellung eines Expertensystems zu reduzieren.

Der Begriff der Expertensystemschale (engl. expert system shell) wird in der Literatur nicht einheitlich verwendet. Während manche Autoren erst problemspezifische Werkzeuge als Shells bezeichnen, zählen andere auch die allgemeinen Werkzeuge dazu (vgl. Abb. 11).

Hier soll unter einer ***Expertensystemschale*** ein rechnergestütztes Werkzeug verstanden werden, das in Anlehnung an die allgemeine Architektur eines Expertensystems (s. Abbildung 1) das Steuersystem und eine zunächst noch leere Wissensbasis enthält. Dabei spielt es keine Rolle, ob das Steuersystem allgemein oder bereits problemspezifisch aufgebaut ist.

In Abbildung 11 wurde eine Klassifikation der verschiedenen Werkzeuge zur Entwicklung von Expertensystemen nach dem Grad der Spezialisierung und der Verkürzung der Entwicklungszeit dargestellt. Dementsprechend läßt sich knoX den *allgemeinen Werkzeugen* zuordnen, obwohl es bei der Unterstützung des Entwicklers über die Bereitstellung einer Wissensrepräsentationssprache mit zugehörigen Ableitungsmechanismen weit hinausgeht.

Die wesentlichen Merkmale von knoX werden im folgenden kurz erläutert. Eine genaue Beschreibung der Schale findet sich in Theiss (1992).

2.1 Leistungsbeschreibung

Durch die Verwendung der hybriden Expertensystemschale wird die Entwicklung eines Expertensystems auf den Aufbau der Wissensbasis reduziert. Die Schale verfügt bereits über eine Reihe von Inferenz- und Erklärungsmechanismen, die unabhängig von dem jeweiligen Anwendungsgebiet eingesetzt werden können. Bei wissensbasierten Systemen stehen besonders die Aufgaben

- *Aufbau einer Wissensbasis* und
- *Konsultation einer Wissensbasis*

im Vordergrund.

2.1.1 Aufbau einer Wissensbasis

Der Aufbau der Wissensbasis ist der erste Schritt bei der Entwicklung eines Expertensystems. Hierbei wird das Wissen über das entsprechende Aufgabengebiet in Form von vorgegebenen syntaktischen Einheiten (Frames, Instanzen, Behaviour, Regeln, Fakten und Prolog-Klauseln) vom Wissensingenieur und dem Experten im System eingegeben. Der Benutzer kann das Textfile der Wissensbasis, oft als ***externe Wissensbasis*** bezeichnet, mit einem Editor erstellen bzw. verändern.

Dieses File muß dann zur weiteren Verarbeitung mit Hilfe eines Compilers in eine interne Form, auch ***interne Wissensbasis*** genannt, übersetzt werden. Der Compiler prüft jeden Eintrag auf syntaktische Korrektheit und übersetzt ihn in ein internes Format, das von der Inferenzkomponente verarbeitet werden kann. Alle syntaktischen und semantischen Fehler in der Wissensbasis werden in einem Übersetzungsdurchlauf ermittelt und in einer Fehlerdatei abgelegt, die sich der Entwickler dann ansehen kann. Nach Beseitigung der Fehler muß die ganze Wissensbasis neu übersetzt werden.

Um den Entwicklungsaufwand beim Aufbau einer Wissensbasis zu reduzieren, können bereits fertige Teile der Wissensbasis in Include-Files abgelegt und getrennt übersetzt werden. Beim Übersetzen der Hauptwissensbasis werden sie dann als Code-File eingelagert. Damit müssen immer nur noch die geänderten und die ihnen übergeordneten Dateien, d.h. solche, in die sie eingebettet sind, übersetzt werden.

2.1.2 Konsultation einer Wissensbasis

Zu Beginn einer Konsultationssitzung wählt der Benutzer die Wissensbasis aus, die er bearbeiten möchte.

Während der Konsultation versucht das System, die in der Wissensbasis in Form einer Agendaliste vorgegebenen Ziele zu erreichen. Dazu leitet die Problemlösungskomponente aus dem Wissen in der Wissensbasis und den Eingaben des Benutzers neue Fakten ab, bis die Lösung des Problems gefunden wird. Alle vom System selbst abgeleiteten sowie vom Benutzer eingegebenen Fakten werden in einer ***dynamischen Wissensbasis*** (Cache) abgespeichert.

Der Benutzer hat während und nach der Konsultation die Möglichkeit, sich die Schlußfolgerungen des Systems zu verdeutlichen, oder eine Unterstützung, bei der Beantwortung einer vom System gestellten Frage anzufordern. Dazu stehen verschiedene Retrievalfunktionen zur Verfügung, mit denen Teile der statischen und dynamischen Wissensbasis jederzeit analysiert werden können. Während der Konsultation können mögliche Eingabewerte, eine nähere Beschreibung der vom System gestellten Frage sowie die Begründung der Benutzerbefragung abgerufen werden. Die Konsultation kann jederzeit abgebrochen und in ihrem aktuellen Stand in einer Datei abgespeichert werden, um sie zu einem späteren Zeitpunkt wieder fortzusetzen.

2.2 Systemarchitektur

Die Systemarchitektur der hybriden Expertensystemschale knoX ist in Abbildung 12 dargestellt. Sie entspricht im wesentlichen der allgemeinen Architektur eines Expertensystems aus Abbildung 1, auch wenn dies auf den ersten Blick durch die etwas andere Darstellung nicht unbedingt ersichtlich ist. Alle Komponenten aus der allgemeinen Architektur sind auch in knoX anzutreffen. Darüber hinaus wurden noch zwei weitere Komponenten aufgenommen, deren Bedeutung noch erläutert wird. Die Pfeile zwischen den Systemkomponenten zeigen den Informationsfluß im System.

Im folgenden werden die verschiedenen Komponenten der Schale zusammen mit den darin realisierten Funktionen beschrieben.

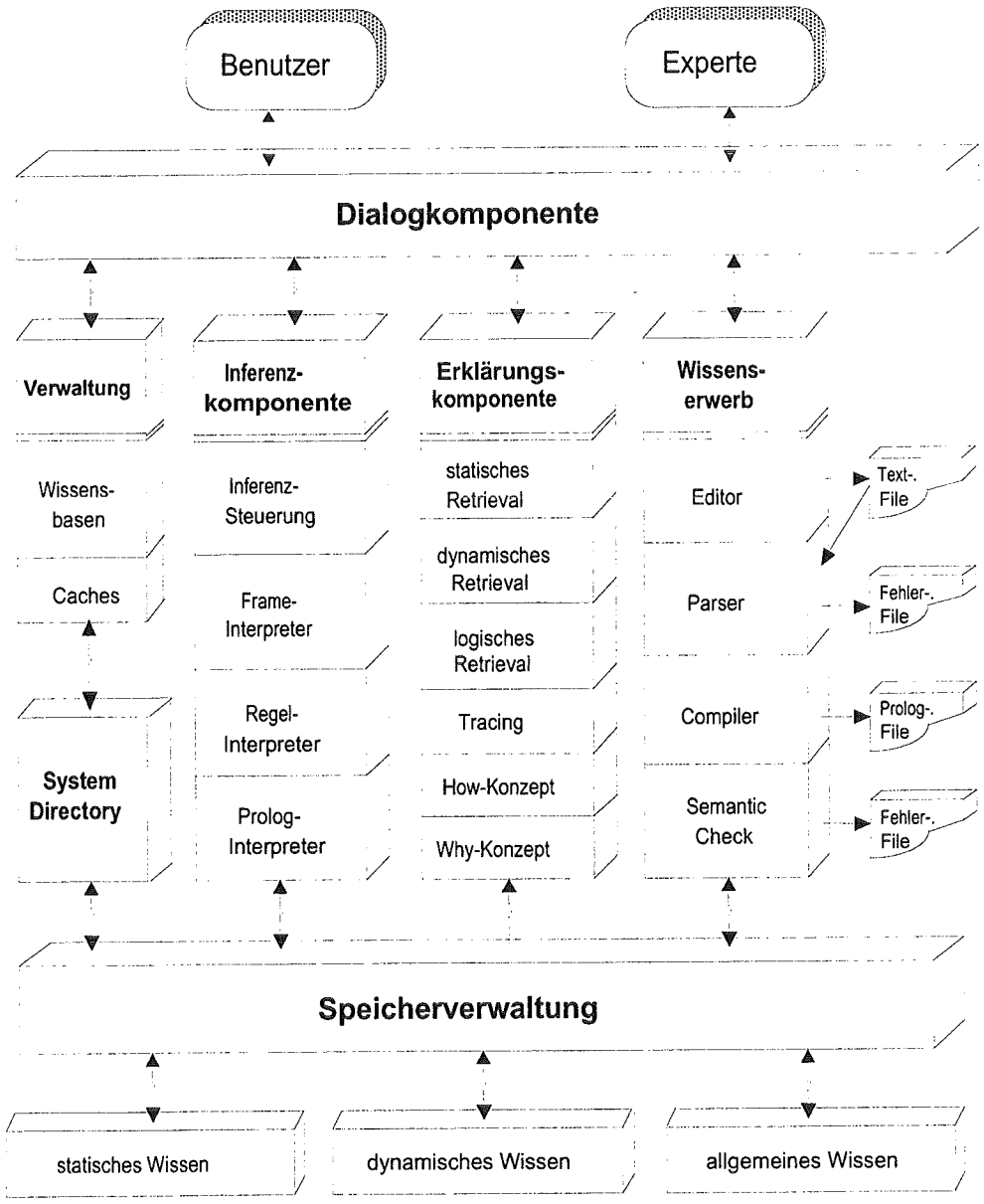


Abb. 12: Architektur der hybriden Expertensystemschale knoX

2.2.1 Dialog-Komponente

Die Dialogkomponente bildet die Schnittstelle zwischen dem System und dem Benutzer. Sie ermöglicht den Zugriff des Benutzers auf die verschiedenen Systemfunktionen und bearbeitet die Anfragen der Systemkomponenten an den Benutzer. Die Auswahl der Systemfunktionen erfolgt über Menüs (siehe dazu auch Held & Maier-Schicht, 1994, S. 7ff).

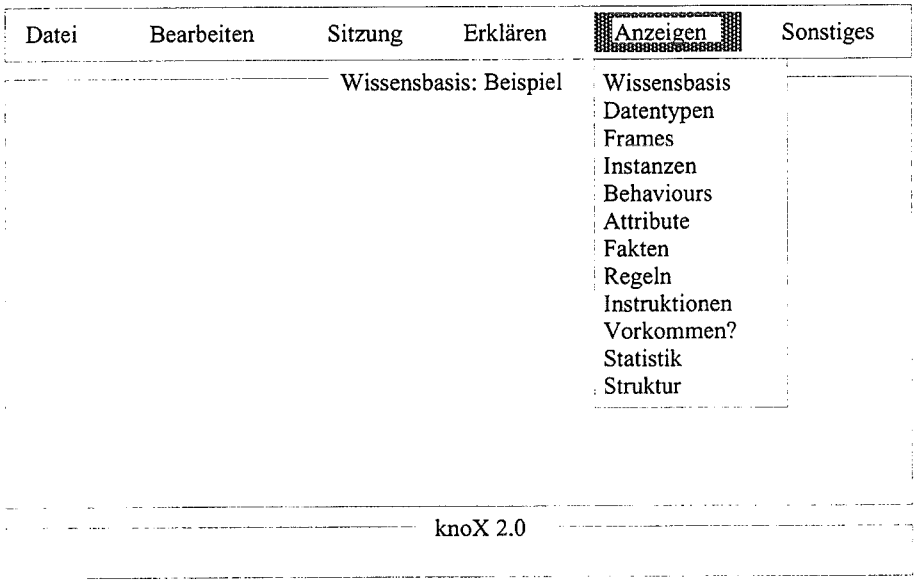


Abb. 13: Beispiel eines Menüs aus knoX

2.2.2 Speicherverwaltung

Um eine einheitliche Schnittstelle zwischen dem System und den veschiedenen Wissensarten zu haben, wurde eine Speicherverwaltung realisiert. Damit wird man unabhängig von der Speicherungsart des Wissens, so daß bei deren Änderung lediglich unterhalb der Speicherverwaltung Änderungen vorgenommen werden müssen, während das Restsystem davon unberührt bleibt. Dies ist deshalb sinnvoll, weil Expertensysteme in der Regel viel Speicherplatz benötigen und es durchaus vorstellbar ist, daß bei größeren Wissensbasen Teile des Wissens auf Datenbanken ausgelagert werden müssen. In der gegenwärtigen Realisierung der Schale wird das gesamte Wissen im *Workspace* des Prolog-Systems[7] gehalten.

Unterhalb der Speicherverwaltung unterscheidet man drei verschiedene Arten von Wissen. Das ***statische Wissen*** enthält die eigentliche Wissensbasis, die vom Wissensingenieur und dem Experten erstellt wird (*bereichsspezifische Expertenwissen*, vgl. Abb. 1). Das ***dynamische Wissen*** enthält alle vom Benutzer während der Konsultation eingegebenen und vom System selbst abgeleiteten Fakten (*fallspezifische Faktenwissen und Zwischen- und Endergebnisse*, vgl. Abb. 1). Das ***allgemeine Wissen*** enthält Informationen, die von der jeweiligen Wissensbasis unabhängig sind, wie z.B. vordefinierte Datentypen, Standardtexte usw. Jeder Zugriff auf die drei Wissensarten erfolgt ausschließlich über die Speicherverwaltung.

[7] knoX ist in der Programmiersprache Prolog implementiert und läuft deshalb in einer Prolog-Laufzeitumgebung.

2.2.3 Wissensbasisverwaltung

Alle mit der hybriden Schale erstellten Wissensbasen und die vom Benutzer abgespeicherten Sitzungen werden vom System in einem eigenen ***Directory*** selbst verwaltet. Der Benutzer kann innerhalb des ersten Menüs eine Wissensbasis auswählen und diese dann bearbeiten. Während bzw. nach der Konsultation kann der aktuelle Stand dieser Konsultation abgespeichert und später wieder geladen werden. Bei jedem Start des Systems wird das Directory aktualisiert, d.h. es werden alle Files aus dem Directory entfernt, die nicht mehr auf der Platte gefunden werden.

2.2.4 Wissenserwerbs-Komponente

Der Wissenserwerb der hybriden Schale erfolgt in mehreren Schritten. Zunächst wird mit einem ***Editor*** das Text-File der Wissensbasis erstellt bzw. ein bereits vorhandenes File verändert. Dieses File wird dann vom ***Parser*** eingelesen und auf syntaktische Korrektheit überprüft. Jeder Syntaxfehler wird mit einer entsprechenden Fehlermeldung gekennzeichnet und zusammen mit dem Eintrag, in dem er erkannt wurde, in ein Fehler-File eingetragen. Syntaktisch korrekte Einträge werden an den ***Compiler*** übergeben, der sie in eine interne Form übersetzt und in einem Binär-File abspeichert. Nach einer erfolgreichen Übersetzung wird die Wissensbasis auf semantische Korrektheit überprüft, und die dabei erkannten Fehler werden ebenfalls in das Fehler-File eingetragen.

2.2.5 Inferenz-Komponente

Die Aufgabe der Inferenzkomponente (***Problemlösungskomponente***) ist die Durchführung der Konsultation und die Steuerung des Inferenzprozesses. Die primäre Steuerung des Inferenzprozesses erfolgt durch die in der Wissensbasis enthaltenen Instruktionen. Für die unterschiedliche Suche nach Werten für Slots (Attribute) stehen zwei verschiedene Strategien zur Verfügung. Falls nichts anderes angegeben ist, versucht die globale Steuerung, einen Wert für einen Slot (Attribut) zunächst über Regeln, dann durch Befragung des Benutzers, und schließlich über einen Default-Wert zu bekommen. Die parameterabhängige Steuerung erlaubt, die Reihenfolge für die Suche nach einem Wert für jeden Slot explizit vorzugeben. Um die verschiedenen Darstellungsformen des Wissens verarbeiten zu können, werden drei verschiedene Interpreter verwendet. Dies sind der Frame-, Regel- und Prolog-Interpreter.

2.2.6 Erklärungs-Komponente

Die Erklärungskomponente soll die Analyse der Wissensbasis unterstützen, das Systemverhalten nachvollziehbar machen sowie Fragen der Inferenzkomponente an den Benutzer begründen und abgeleitete Fakten rechtfertigen können. Um diesen Anforderungen gerecht zu werden, sind alle notwendigen Konzepte realisiert. Mit dem ***statischen Retrieval*** können Teile der Wissensbasis analysiert werden, das ***dynamische Retrieval*** gestattet die Analyse der vom System selbst abgeleiteten und vom Benutzer eingegebenen Fakten, und das ***logische Retrieval*** ermöglicht, Teile der Wissensbasis unter ganz bestimmten Gesichtspunkten zu betrachten. Der ***Trace-Mechanismus*** macht das Systemverhalten während der Konsultation einer Wissensbasis nachvollziehbar und unterstützt somit die Fehlersuche bei falschen Ableitungen. Die ***Why-Erklärung*** liefert die Begründung für Fragen der Inferenzkomponente an den Benutzer, und die ***How-Erklärung*** rechtfertigt die vom System abgeleiteten Fakten. Dafür greift die Erklärungskomponente auf die von der Inferenzkomponente während der Konsultation erzeugten History-Protokolle zurück.

3 Das Expertensystem Destillation

3.1 Aufgabenstellung im Projekt A5

Im Finanzierungsantrag des SFB 245 für 1992 - 1994 an die DFG wurde die Aufgabe des Projekts A5 "Darstellung technischer Abläufe in dialogischen Instruktionen" so formuliert:

"Ziel des Projekts ist die Entwicklung und partielle Überprüfung eines Modells der sprachlichen Vermittlung technischen Wissens in dialogischen Situationen.
... Das gesamte, den technischen Ablauf des Verfahrens betreffende Expertenwissen hat theoretische, instrumentelle und handlungsbezogene Anteile.
... Das potentielle Planungsgerüst ist hierarchisch organisiert.
... Dieses Gerüst bildet ein Modell der kognitiven Grundlage der Instruktionsproduktion. Die einzelnen Äußerungen der Instruktion können entsprechend in diesem Gerüst lokalisiert werden." (Finanzierungsantrag 1992-1994, SFB 245, S. 216)

Bedeutung des Expertensystems für die Forschung in A5

Mit Hilfe eines Expertensystems sollte das oben beschriebene Modell partiell überprüft werden. Ziel war es, *"die empirischen Befunde mit dem Bereich technischer Expertensysteme in Verbindung zu bringen."* (Finanzierungsantrag 1992-1994, SFB 245, S. 229)
Es ging darum, *"die Möglichkeit, das Wissen über die Adaptivität von Instrukteuren auf Expertensysteme zu beziehen und unter Umständen hierfür nutzbar zu machen."* (ebenda)

Für die Adaptivität von Experten im Dialog sah das Projekt A5 mindestens 5 Indikatoren:

1. *"Experten adaptieren den Auflösungsgrad ihrer Darstellung an den Novizen.*
2. *Experten verwenden besondere sprachliche Markierungen bei nicht-standardisierten Handlungsschritten.*
3. *Experten konkretisieren ihre Darstellungen mit Hilfe von Verweisen auf allgemeines Handlungswissen aus dem Alltag.*
4. *Experten wechseln in ihrer Darstellung flexibel von Handlungswissen zu Sachverhaltswissen und umgekehrt.*
5. *Experten geben Um-Zu-Relationen nicht nur innerhalb des engen Kontexts des Apparats und der Handlung an, sondern ziehen in ihren Begründungen zusätzlich theoretisches Wissen heran." (ebenda)*

Das Projekt A5 wollte prüfen, ob das zuvor beschriebene *"Expertenwissen über Destillation in einem Expertensystem so eingesetzt werden kann, daß eine Adaptivität des Expertensystems im Sinne unserer Indikatoren erreichbar ist."* (ebenda)

Eine Analyse der Hörerangemessenheit in Instruktionen sollte anhand einer Untersuchung mit einem Expertensystem geleistet werden. Es war geplant, zur Realisierung eine Expertensystemschale einzusetzen, die an der Universität Karlsruhe im Fachbereich Informatik von G. Theiss u.a. entwickelt worden war.

Ein Ziel des Projekts war die Entwicklung und partielle Überprüfung eines Modells der sprachlichen Vermittlung technischen Wissens in dialogischen Situationen. Auf den theoretischen Überlegungen aufbauend, die in der ersten Förderperiode angestellt worden waren, sollten mit einem Experimentalsystem die Ergebnisse überprüft und das Modell weiterentwickelt werden.

Folgende Bedingungen sollte das Experimentalsystem erfüllen:

1. die theoretischen, instrumentellen und handlungsbezogenen Anteile des Wissens über Destillation, die im Projekt herausgearbeitet worden waren, als Modell im Experimentalsystem abbilden,
2. Experimente ermöglichen, die auf der ersten Stufe den Weg der Versuchspersonen durch die Wissenshierarchien liefern,
3. die Experimente protokollieren und Vorbereitungen für die Auswertung treffen,
4. Adaptivität im Sinne der Indikatoren des Projekts ermöglichen.

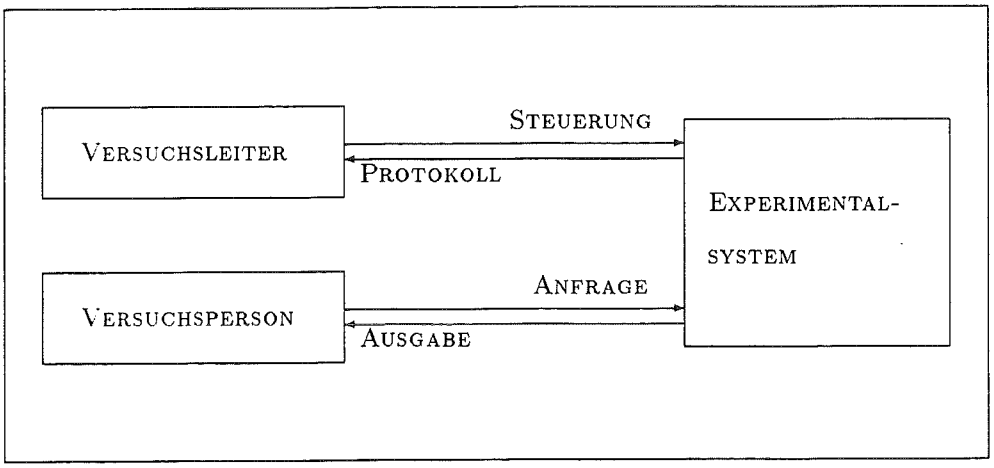


Abb. 14: Expertensystem als Experimentalsystem

3.2 Umsetzung der Aufgabenstellung

Das Ziel im Projekt A5 war also die Entwicklung und partielle Überprüfung eines Modells der sprachlichen Vermittlung technischen Wissens in dialogischen Situationen. Das Expertensystem sollte dabei als Experimentalsystem eingesetzt werden.

Auch andere Möglichkeiten für die Durchführung der Versuche wären denkbar gewesen, z.B. Hypertext oder algorithmische Programmierung in Pascal o.ä.

Aber besonders die Möglichkeiten, die in der Realisierung als Expertensystem steckten und weit über das in der ersten Stufe vorgesehene Modell hinausgingen, führten dazu, daß diese Vorgehensweise gewählt wurde.

Ein Expertensystem als Experimentalsystem

Die Gründe, die zu einer Realisierung des Experimentalsystems als Expertensystem mit Hilfe der Expertensystemschale knoX führten, waren folgende:

- **Entwicklungsmöglichkeiten**
  Alle Bedingungen, die das Experimentalsystem erfüllen sollte (s. 3.1), konnten über ein Expertensystem realisiert werden.
  Darüberhinaus bot ein solches System Möglichkeiten für den weiteren Ausbau, die in der ersten Testphase noch bei weitem nicht ausgeschöpft wurden. So war die Einbindung von Datenbanken, Grafiken u.ä. möglich, ebenso die Programmierung von Prozeduren in Prolog. Die ganze Bandbreite der möglichen Problemlösungsstrategien wurde in den ersten Versionen noch nicht eingesetzt, die Wissenserwerbskomponente wurde nicht verwendet, usw. Ein Ausbau in Richtung Tutoring-System war möglich (Modellierung eines Nutzer-Modells, Adaptivität gegenüber Nutzer und Aufgabenstellung).

- **Pflege und Änderung**
  Durch den geplanten modularen Aufbau des Systems konnte eine Pflege- und Änderungsfreundlichkeit erreicht werden, die den weiteren Umgang auch für Laien im Gebiet Expertensysteme ermöglichte (Trennung der Wissensbasis von den Verarbeitungsstrategien).

- **Interdisziplinäre Zusammenarbeit**
  Es bot sich hier eine Gelegenheit, in interdisziplinärer Zusammenarbeit mit der Universität Karlsruhe (Fachbereich Informatik) neueste Entwicklungen aus dem Bereich der Informatik in der Psychologie einzusetzen. Andererseits konnten die Ergebnisse aus dem SFB auch zu Rückwirkungen auf die Entwicklung von Expertensystemen in der Informatik, auf den Einsatz von psychologischen Erkenntnissen, z. B. bei der Entwicklung von Tutoring Systemen, oder dem Aufbau von Wissensbasen führen.
  Die hybride Expertensystemschale knoX, die an der Universität Karlsruhe entwickelt worden war, bot als allgemeines Werkzeug mit Frames, Regeln und Prolog-Klauseln drei verschiedene Wissensrepräsentationsformen sowie in der Problemlösungskomponente die entsprechenden Ableitungsstrategien.

Zusammenarbeit mit dem Institut für Mikrorechner und Automation der Universität Karlsruhe

Nach den Voruntersuchungen im Projekt A5 wurden in einer Diplomarbeit im Bereich Informatik, die vom SFB mitbetreut und an der Universität Karlsruhe geschrieben wurde, erste Schritte zur Implementierung geleistet (Pimentel, 1992):

- Auswahl der Repräsentationsformen,
- Formalisierung des Wissens,
- erste Implementierung mit knoX,
- Übersetzung und damit Validierung der ersten Testversion.

Im November 1992 konnte das Expertensystem den Vorstellungen des Projektes weiterangepasst werden. Als Gastwissenschaftlerin arbeitete die ehemalige Diplomandin E. Pimentel für einen Monat im SFB in Heidelberg. Die Vorstellungen über das Expertensystem wurden in der Auseinandersetzung zwischen Psychologen und Informatikern konkretisiert:

- die Menüs, aus denen der Benutzer auswählt,
- die Präsentation der Information,
- die Notwendigkeit von sog. Zwischentexten beim Wechsel des Benutzers von einer Wissenshierarchie zu einer anderen.

Durch die Einladung von Dipl. Inform. G. Theiss (Universität Karlsruhe), der an der Entwicklung von knoX maßgeblich mitgearbeitet hatte, konnten im Herbst 1993 folgende Punkte verbessert bzw. ergänzt werden:

- Verbesserung der Speicherverwaltung,
- Anpassung der Benutzermenüs,
- Einbindung von Grafik.

3.3 Entwicklung des Expertensystems Destillation

Bei der Entstehung des Expertensystems als Experimentalsystem können folgende Stufen der Modellierung (entspr. auch der modellbasierten Vorgehensweise nach KADS, s. Kap.1) identifiziert werden:

1. **das Analysemodell**, das möglichst das gesamte Wissen des Anwendungsgebietes umfaßt (hier also das Wissen über die Destillation und über das erforderliche Verhalten des Systems als Experimentalsystem),
2. **das globale Entwurfsmodell** mit der implementierungsorientierten Beschreibung des Systems, die aber noch völlig werkzeugunabhängig ist,
3. **das technische Entwurfsmodell**, in dem die Realisierung der einzelnen Teile und ihres Zusammenwirkens durch das Werkzeug (die Expertensystemschale) beschrieben wird.

Zur Implementierung des Experimentalsystems siehe Pimentel (1992) und Held & Maier-Schicht (1994).

3.3.1 Das Analysemodell

Das Analysemodell basiert auf den Untersuchungen, die das Projekt im ersten Förderungszeitraum durchführte. Es beinhaltet das Wissen aus dem Bereich Destillation, eine Vorgabe über das Zusammenwirken von System und Umwelt sowie die funktionalen Anforderungen für den Einsatzbereich.

Innerhalb des Analysemodells kann man nach KADS drei Stufen unterscheiden:

- das *konzeptuelle Modell*, in dem das deklarative und prozedurale Wissen beschrieben wird;
- das *Kooperationsmodell*, das die Beziehungen des Systems zur Umwelt enthält;
- das *Anforderungsmodell* mit den funktionalen Anforderungen, also z. B. den Bedingungen an Hard- und Software.

Die Informationen dieser drei Modell überlappen sich allerdings und können nicht strikt getrennt werden. Sie modellieren drei verschiedene Sichten auf eine Problemstellung.

Die Problemstellung läßt sich wie folgt darstellen

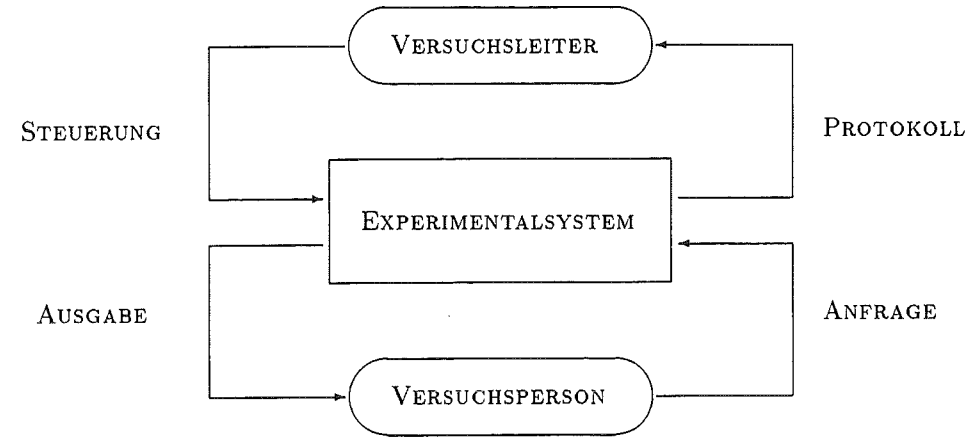


Abb. 15: Struktur des Experimentalsystem

Der Versuchsleiter (VL) gibt die Einstellungen des Expertensystems vor und legt damit das Verhalten des Systems fest. Die Versuchsperson (VP) stellt Anfragen an das System und erhält Antworten. Der gesamte Verlauf des Versuchs wird für die Auswertung durch den Versuchsleiter protokolliert.

Konzeptuelles Modell

Hier muß geklärt werden, welches Wissen erforderlich ist und wie das System auf der Basis dieses Wissens arbeiten soll. Wie sollen die Anfragen in Ausgaben umgesetzt werden unter Berücksichtigung der vom Versuchsleiter getroffenen Einstellungen ?

In den vorausgehenden Untersuchungen des Projekts wurden Experten aus unterschiedlichen Bereichen gebeten, Hörer unterschiedlichen Vorwissens zu instruieren. Die Aufgabe bestand darin, die Destillation von Äthanol so zu erklären, daß der Hörer daraus die notwendigen Informationen ziehen konnte, um die Handlung (die Destillationsapparatur aufzubauen und die Destillation durchzuführen) eigenständig auszuführen. Chemiker, Ingenieure und Chemielehrer bildeten die Gruppe der Experten, die Hörer setzten sich aus Leistungskursabsolventen und Laien zusammen. Weitere Informationen wurden aus den entsprechenden Chemielehrbüchern erarbeitet.
In diesen Voruntersuchungen ergab sich eine Einteilung des Wissens über Destillation in 3 Bereiche:

1. Informationen zum theoretischen Hintergrund der Destillation,
2. Informationen über die Destillationsapparatur mit Beschreibung der Bestandteile und ihrer Lokalisation innerhalb der Apparatur,
3. Informationen über die Handlungen, die zum Aufbau bzw. zum Betreiben der Apparatur erforderlich sind.

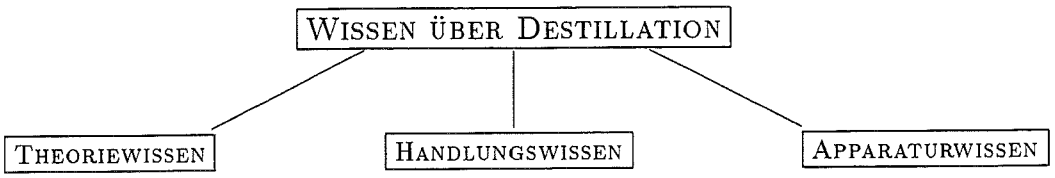


Abb. 16: Einteilung des Wissens über Destillation

Jeder Bereich war hierarchisch strukturiert, wenn es auch im Bereich Theorie in dieser ersten Phase nur eine Ebene gab. Diese Hierarchien waren über eine Baumstruktur vorgegeben. Jeder Knoten war durch einen Namen gekennzeichnet, und ein Text war ihm zugeordnet. Für jeden einzelnen Knoten war festgelegt, welcher Knoten die allgemeinere, die präzisere oder einfach weiterführende Informationen enthielt. Weiter waren für jeden Knoten die Einstiegsknoten beim Wechsel von einer Hierarchie in eine andere definiert.

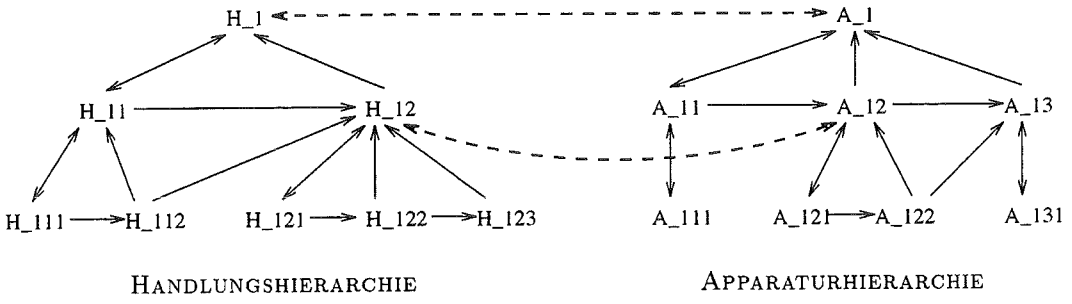
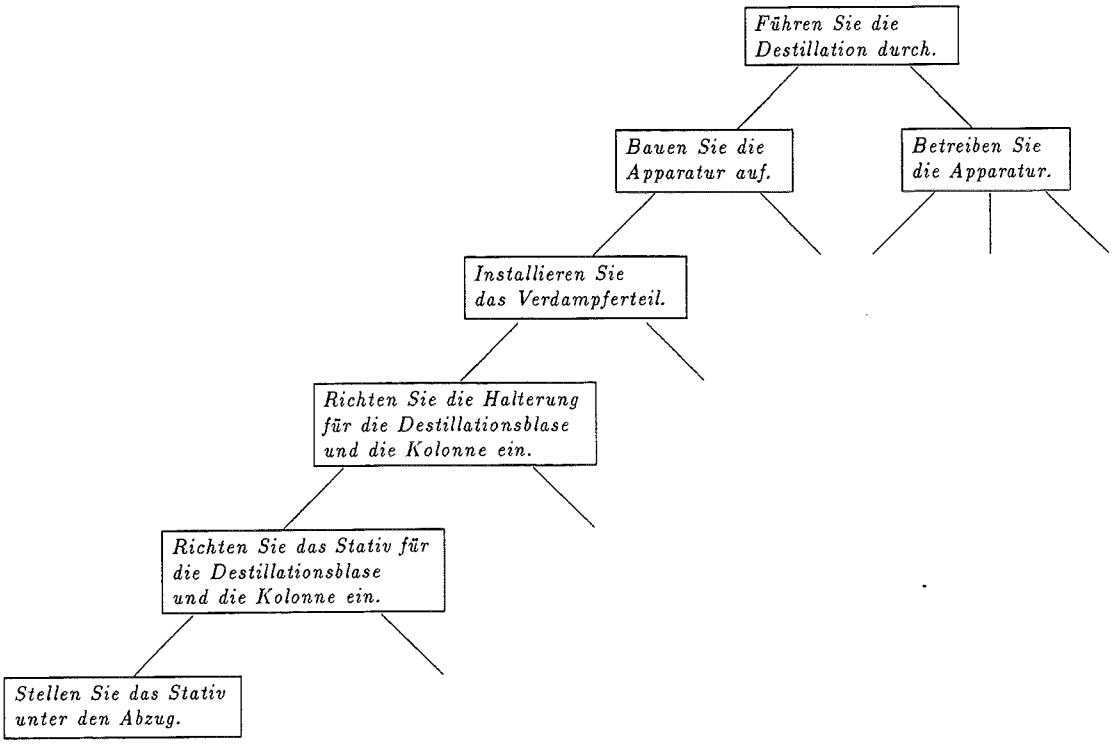


Abb. 17: Auszug aus der Hierarchiestruktur (Held & Maier-Schicht, 1994, S. 9).

Durchgezogene Linien stehen für hierarchische Verbindungen zwischen den einzelnen Informationseinheiten. Gestrichelte Linien zeigen Verbindungen zwischen den Hierarchien. Die Pfeilspitzen an den Verbindungslinien geben die möglichen Richtungen der Navigation durch die Wissensbasis an

Schema der Handlungshierarchie



In der Beschreibung der Struktur des Wissens über Destillation, des sog. *deklarativen Wissens*, war festgelegt, welcher Knoten mit Text angezeigt werden sollte, wenn ein Nutzer allgemeinere oder präzisere Information anforderte, oder wenn er in den Informationen auf derselben Abstraktionsebene einfach weitergehen wollte. Für jeden Knoten war jeweils ein Knoten in den beiden anderen Hierarchien ausgewählt, der bei einem Wechsel in diese Hierarchien als Einstieg benutzt werden sollte. Weiter war ein erklärender Zwischentext festgelegt worden, der beim Hierarchiewechsel zusätzlich ausgegeben werden sollte.

Der Bereich des *prozeduralen Wissens* wurde festgelegt zum einen über die Steuerungseinstellungen, die eine Anpassung an verschiedene Experimentalanordnungen leisten sollte, zum anderen über die Angabe der Möglichkeiten, die eine Nutzer haben sollte, um sich Informationen vom System ausgeben zu lassen.

Die *Problemlösung* bestand darin, unter den Bedingungen der Voreinstellungen aus den Anfragen die Ausgaben zu bestimmen, die weiteren Verarbeitungsmöglichkeiten anzubieten und den gesamten Verlauf zu protokollieren.

Schema des Experimentalsystems

Schematisch kann der Ablauf eines Experiments (einer Sitzung) folgendermaßen betrachtet werden:

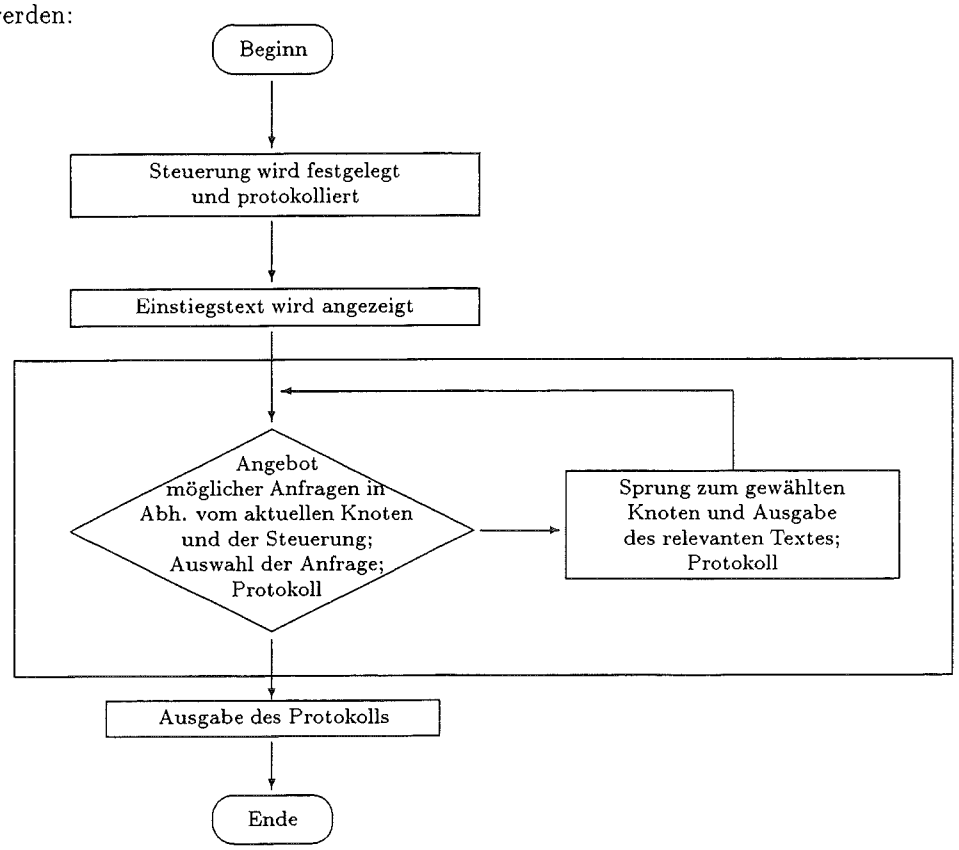


Abb. 17: Ablauf des Experiments

Beginn und Ende kennzeichnen die beiden Punkte, zwischen denen sich das Experiment abspielt.

Die Festlegung der Steuerung erfolgt vor dem eigentlichen Versuch durch den Versuchsleiter. Hier werden generelle Vorgaben gemacht über Versuch und Versuchsperson, darüber welches Abbruchkriterium gelten soll, welche Informationen zusätzlich ausgegeben werden sollen sowie Angaben über die Protokollierung. Auch diese Angaben werden im Protokoll festgehalten.

Im eigentlichen Experiment wird der Versuchsperson in Abhängigkeit von der eingestellten Steuerung und von der aktuellen Position in der Hierarchiestruktur ein Menü angeboten, das die möglichen Anfragen anzeigt. Nach der Auswahl eines Punktes wird dies mit dem Zeitpunkt im Protokoll festgehalten, und die gewünschte Information wird angezeigt.
Dieser Vorgang wiederholt sich, bis entweder eine vorgegebene Zeitdauer verstrichen ist oder der Nutzer die Sitzung beendet.

Mathematische Modellierung des Experimentalsystems

Ziel dieses Schrittes ist die Erstellung eines formalen Modells.

Das Experimentalsystem soll im folgenden mathematisch beschrieben werden. Eine solche Modellierung setzt eine bestimmte, mathematische Sichtweise voraus:
Das Experimentalsystem wird als System im mathematischen Sinn betrachtet, kann also z.B. beschrieben werden durch seine Elemente und die Beziehungen zwischen diesen Elementen. Oder es kann beschrieben werden durch seine Innen- und Außenstruktur usw. Welche Beschreibung gewählt wird, ist abhängig vom konkreten Fall, von der Problemstellung.

Für das vorliegende System können die drei Hierarchien mit ihren Knoten und den Beziehungen dazwischen (das sog. deklarative Wissen) als eine innere Struktur betrachtet werden, während die äußere Struktur (die "Umwelt des Systems", das prozedurale Wissen) durch die Steuerung und die Anfragen, die als Input das System beeinflussen, und durch Protokoll und Informationsausgabe bestimmt wird.

Die Aufgabe des Gesamtsystems besteht darin, nach dem Festlegen der Einstellungen die Anfragen zu bearbeiten und ein Protokoll zu erstellen.

Modellierung des deklarativen Wissens

Das deklarative Wissen, das das Experimentalsystem verarbeitet, umfaßt die Knoten, die Hierarchien, die Texte, in die das Wissen aufgespalten wurde, und die Beziehungen zwischen diesen Elementen.

Dieses deklarative Wissen bildet die Grundlage jeder Nutzung des Experimentalsystems.

Als Objekte können definiert werden

die Menge der Knoten $\mathcal{KM}$

$$\mathcal{KM} := \{k/k \; ist \; Knoten\} = \{k_1, k_2, k_3 \ldots\}$$

die Menge der Texte $\mathcal{TM}$

$$\mathcal{TM} := \{txt/txt \; ist \; Text\} = \{txt_1, txt_2, txt_3, \ldots\}$$

die Menge der Hierarchien $\mathcal{HM}$

$$\mathcal{HM} := \{h/h \; ist \; Hierarchie\} = \{\text{apparatur, handlung, theorie}\}$$

die Menge der möglichen Anfragen an das System $\mathcal{AM}$

$\mathcal{AM} := \{a/a \; ist \; Anfrage\} =$ {weiter, allgemeiner, präziser, Wechsel zu Apparatur, Theorie oder Handlung}

Entsprechend werden die Beziehungen oder Abbildungen zwischen den oben definierten Mengen festgelegt.

$f_{hier} : \mathcal{KM} \rightarrow \mathcal{HM}$	jedem Knoten wird eine Hierarchie zugeordnet: $f_{hier}(k) := h$
$f_{text} : \mathcal{KM} \times \mathcal{AM} \rightarrow \mathcal{TM}$	jedem Paar $(knoten, anfrage)$ wird ein Text zugeordnet: $f_{text}(k, a) := txt$
$f_{next} : \mathcal{KM} \times \mathcal{AM} \rightarrow \mathcal{KM}$	jedem Paar $(knoten, anfrage)$ wird ein nächster Knoten zugeordnet: $f_{next}(k_1, a) := k_2$, dabei kann gelten: $k_1 = k_2$ (Diese Abbildung beschreibt das gesamte Beziehungsgeflecht zwischen den Knoten.)

Modellierung des prozeduralen Wissens

Das prozedurale Wissen beschreibt, was durch das Experimentalsystem geleistet werden soll, und wie sich das System verhalten soll. Nachdem das System eingestellt ist, werden die Eingaben bearbeitet: Informationsausgabe und Protokollerstellung.

Als Elemente und Beziehungen kann man identifizieren:

die Menge der möglichen Experimentsteuerungen $\mathcal{ST}$

$\mathcal{ST} := \{st/st\ ist\ Experimentsteuerung\}$
(allgemeine Bedingung unter der das System arbeitet)

die Menge der möglichen Wege $\mathcal{WM}$ als Folge von Knoten

$\mathcal{WM} := \{w/w\ ist\ eine\ Knotenfolge\}$

die Menge der möglichen Protokolle als Folge von Protokolleinträgen (bestehend aus letzter Anfrage, aktuellem Knoten und Zeitpunkt)

$\mathcal{PM} := \{prot/prot\ ist\ ein\ Protokoll\}$

$f_{anfr} : \mathcal{KM} \rightarrow \mathcal{P}(\mathcal{AM})$ jedem Knoten wird die Teilmenge der möglichen Anfragen zugeordnet, die an diesem Knoten möglich sind
mit $\mathcal{P}(\mathcal{AM}) := \{\mathcal{A}/\mathcal{A} \subseteq \mathcal{AM}\}$ der Potenzmenge von $\mathcal{AM}$

Um die Bearbeitung der Anfragen zu beschreiben, werden die folgenden Prädikate[8] definiert:
$zust(\mathcal{Z}, \mathcal{T})$ mit Zustand $\mathcal{Z} = \mathcal{AM} \times \mathcal{KM} \times \mathcal{WM}$ und Zeit $\mathcal{T} = \mathbb{R}_0^+$

Zu jedem Zeitpunkt t läßt sich der aktuelle Zustand des Systems beschreiben durch die Angabe

1. der aktuellen Anfrage a,
2. des aktuellen Knotens k (dadurch ist die aktuelle Hierarchie $f_{hier}(k)$ und das aktuelle Menü $f_{anfr}(k)$ bestimmt),
3. des bisher zurückgelegten Weges w innerhalb der Hierarchien.

$anfr(\mathcal{AM}, \mathcal{T})$ die Anfrage $a \in \mathcal{AM}$ zur Zeit $t \in \mathcal{T}$

$ausg(\mathcal{AU}, \mathcal{T})$ die Ausgabe $au \in \mathcal{AU}$ zur Zeit $t \in \mathcal{T}$ mit $\mathcal{AU} = \mathcal{TM} \times \mathcal{P}(\mathcal{AM})$,

Modellierung der Problemlösung

Die Problemlösung kann formuliert werden durch

$$
\begin{aligned}
zust(< a_0, k, w >, t_0) \wedge anfr(a_1, t_1) \rightarrow \quad & ausg(< f_{text}(k, a_1), f_{anfr}(f_{next}(k, a_1)) >, t_1) \\
& \wedge\ zust(< a_1, f_{next}(k, a_1), w \circ f_{next}(k, a_1) >, t_1) \\
& \wedge\ prot \circ (< a_1, f_{next}(k, a_1), t_1 >) \\
& \text{mit } k \in \mathcal{KM},\ w \in \mathcal{WM},\ t_0, t_1 \in \mathcal{T}, \\
& a_0, a_1 \in \mathcal{AM} \text{ und } \circ \text{ als Listen-Konkatenation}
\end{aligned}
$$

[8] Prädikate sind hier Funktionen, die bei Ausführung die Werte (wahr/falsch) liefern.

Interpretation:

Wenn das System zum Zeitpunkt t_0 den Zustand $< a_0, k, w >$ hat,
(d. h. wenn der Zustand des Systems zum Zeitpunkt t_0 beschreibbar ist durch das Tripel $< anfrage_0, knoten, zurückgelegter\ weg >$),

und zum Zeitpunkt t_1 die Anfrage a_1 lautet,

dann wird der Text ausgegeben $f_{text}(k, a_1)$,
ebenso das Angebot der weiteren Schritte $f_{anfr}(f_{next}(k, a_1))$,

der Systemzustand wird zur Zeit t_1 neu beschrieben durch das Tripel
$< a_1, f_{next}(k, a_1), zurückgelegter\ weg \circ f_{next}(k, a_1) >$,

und ins Protokoll wird eingetragen $< a_1, f_{next}(k, a_1), t_1 >$.

Das Kooperationsmodell

Das Kooperationsmodell beschreibt die Verbindung zwischen System und Umgebung. Es legt fest, wie der Ablauf zwischen System und Nutzer aussieht.
Das Kooperationsmodell wird hier betrachtet als Modell zur Definition der Interaktion zwischen Versuchsleiter und Versuchsperson auf der einen Seite und dem Experimentalsystem auf der anderen Seite. Es bezieht sich auf die Beschreibung der Menüs, die die Möglichkeiten der Versuchssteuerung bzw. der Anfragemöglichkeiten dem Nutzer zur Auswahl anbieten. Dazu sollen Masken vom System angeboten und vom Nutzer ausgefüllt bzw. angeklickt werden.

Der Benutzer des Systems hat die Aufgabe sich über Destillation zu informieren, das heißt, sich einen Weg durch dieses Wissen zu suchen: er soll zwischen den Hierarchien springen können, von allgemeinerer Information zu präziserer und umgekehrt, auch soll es möglich sein, den bisherigen Weg wieder zurückzuverfolgen. Um den Sprung zwischen den Hierarchien nicht zu krass werden zu lassen, sind Zwischentexte vorgesehen.

Für die vorgesehenen Auswertungen sollen die Sitzungen mit dem Expertensystem protokolliert werden, d. h. zu jeder Aktion am Rechner sollte Zeitpunkt und Art der Aktion festgehalten werden. Zur genauen Beschreibung der Realisierung dieser Verbindung zwischen System und Umgebung siehe Held & Maier-Schicht (1994).

Das Anforderungsmodell

Folgende technischen Forderungen werden an das Experimentalsystem gestellt:

- Es soll auf einem PC lauffähig sein (z.B. 486er mit 8MB RAM).
- Das System soll *"anwendungsfreundlich"* sein, d. h. ohne große Übungszeit oder Verwendung eines Handbuchs soll die Bedienung des Systems verständlich sein. Das soll z.B. durch Einsatz von Menütechniken geschehen, deren Punkte mit der Maus anzuklicken sind.
  Die auszugebenden Texte, die die Funktion des Systems betreffen, sollen selbsterklärend den Nutzer durch das System führen.
- Ebenfalls müssen die Antwortzeiten so kurz sein, daß keine *"Wartezeiten"* den Benutzer ermüden.

3.3.2 Das globale Entwurfsmodell

Das globale Entwurfsmodell sieht die Abbildung der Elemente und Beziehungen aus dem Analysemodell auf Repräsentationsformen eines Expertensystems vor.
Die mathematische Modellierung wird also umgesetzt in Frames, Regeln und Prozeduren entsprechend den Definitionen des Analysemodells.

Beschreibung des deklarativen Wissens

Für die betrachteten Elemente des deklarativen Wissens bietet sich eine Abbildung in Frames an, die über spezielle Slots auch Beziehungen enthalten können.

FRAMEs ZUR BESCHREIBUNG DER KNOTEN

Für jeden Hierarchieknoten wird eine Instanz von Frame HANDLUNG (bzw. APPARATUR und THEORIE) definiert. Über die Slots werden Beziehungen definiert. Sie enthalten die Zuordnung zum Text f_{text} und f_{next} als Verweis für die Übergänge in die entsprechenden anderen Hierarchien, ebenso wie für die Verzweigungen "allgemeiner", "präziser", "weiter". Um die Benennungen übersichtlich zu gestalten, erhielten die Knoten Namen, die in den ersten beiden Zeichen den Verweis auf die entsprechende Hierarchie und in den darauffolgenden Ziffern den Platz innerhalb dieser Hierarchien enthalten. Dies dient nur der Übersichtlichkeit.[9]

FRAME HANDLUNGSKNOTEN	
name	string
weiter	instance(knoten)
allgemeiner	instance(knoten)
präziser	instance(knoten)
wechsel-a	instance(knoten)
wechsel-t	instance(knoten)

⟶

INSTANZ VON HANDLUNGSKNOTEN	
name	hh1142
weiter	hh12
allgemeiner	hh114
präziser	hh11421
wechsel-a	aa123
wechsel-t	tt1

FRAME APPARATURKNOTEN	
name	string
weiter	instance(knoten)
allgemeiner	instance(knoten)
präziser	instance(knoten)
wechsel-h	instance(knoten)
wechsel-t	instance(knoten)

⟶

INSTANZ VON APPARATURKNOTEN	
name	aa111
weiter	aa112
allgemeiner	aa11
präziser	aa1111
wechsel-h	hh111
wechsel-t	*nicht vorgesehen*

FRAME THEORIEKNOTEN[10]	
name	string
weiter	instance(knoten)
wechsel-h	instance(knoten)
wechsel-a	instance(knoten)

⟶

INSTANZ VON THEORIEKNOTEN	
name	tt4
weiter	tt5
wechsel-h	hh12
wechsel-a	aa13

[9] Bsp.: hh124 bezeichnet einen Knoten der Handlungshierarchie: es ist der 4. Nachfolger des Knotens hh12; er liegt auf der 3. Ebene.

[10] Da die Theoriehierarchie nur aus einer Ebene besteht, gibt es hier keine Angaben für "allgemeiner" und "präziser".

Eine andere Möglichkeit ist, die Struktur der Hierarchien durch Prolog-Fakten anzugeben.

Beispiel: Wird am Knoten1 die Anfrage "weiter" an das System gestellt, dann soll der Knoten2 angesprungen werden: $f_{next}(knoten1, weiter) := knoten2$

als Prolog-Fakt: $weiter(knoten1, knoten2)$

für die Knoten $aa11$ und $aa12$: $f_{next}(aa11, weiter) := aa12$

als Prolog-Fakt: $weiter(aa11, aa12)$

Aus Gründen der Übersichtlichkeit und Änderungsfreundlichkeit wurde die Darstellung durch Frames gewählt.

FRAME ZUR BESCHREIBUNG DER STEUERUNG

Im Frame STEUERUNG enthalten die Slots die einzelnen Komponenten der Versuchssteuerung. Die Versuchsklasse wird festgelegt, Kennungen zur Identifikation von Versuch, Versuchsperson und Protokoll werden vergeben. Kommentar, Knoten und Ende sind Einstellungen, die den Versuchsablauf mitbestimmen.

FRAME STEUERUNG	
klasse	(test-a, test-b, test-c)
identifikation	beliebig
protokoll	dateiname
versuchsperson	kennung
kommentar	(ja, nein)
knoten	(ja, nein)
ende	(normal, 10min, 15min, ...)

$\longrightarrow$

INSTANZ VON STEUERUNG	
klasse	test-a
identifikation	Versuch-1a
protokoll	test\prot1a.txt
versuchsperson	xy-1a
kommentar	ja
knoten	nein
ende	normal

FRAME ZUR BESCHREIBUNG DES SYSTEMZUSTANDES

Der Frame ZUSTAND markiert den aktuellen Zustand des Systems. Hier enthalten die Slots Angaben über die letzte Anfrage, die aktuelle Hierarchie, den aktuellen Knoten sowie Eintragungen über den bisher zurückgelegten Weg, d. h. die besuchten Knoten und damit die ausgegebenen Texte.

FRAME ZUSTAND	
anfrage	(weiter, präzis, ...)
hierarchie	(theorie, apparatur, handlung)
knoten	instance(knoten)
weg	knotenliste

$\longrightarrow$

INSTANZ VON ZUSTAND	
anfrage	weiter
hierarchie	apparatur
knoten	aa12
weg	[aa1,aa11,aa12]

Beschreibung des prozeduralen Wissens

Für die Abbildung der Beziehungen, die den Ablauf und die Verarbeitung betreffen, bieten sich Regeln und Prozeduren an:

wenn <Bedingung> *dann* <Aktion>

Die Abfrage des aktuellen Zustandes löst in Abhängigkeit von der Antwort die Ausführung von Prozeduren aus.

Beispiel:

wenn anfrage=präzis, dann starte die Prozedur `praezis`

Lautet die Anfrage des Nutzers "präzis", dann wird die Prozedur `praezis` gestartet.

Die Prozedur `praezis` bewirkt folgendes:
begin(`praezis`)
- aus der Instanz des aktuellen Systemzustands wird der Knoten geholt, der für die Weiterverarbeitung mit der Anfrage "präzis" angegeben ist;
- der zugehörige Text wird ausgegeben,
- der neue, aktuelle Zustand wird festgehalten und
- das jetzt aktuelle Menü wird angezeigt.

end(`praezis`)

3.3.3 Das technische Entwurfsmodell

Dieses Modell beschreibt die Umsetzung des globalen Entwurfsmodells in die werkzeugabhängige Realisierung des Experimentalsystems. Die zuvor definierten Repräsentationsformen werden abgebildet in Formen, die das verwendete Werkzeug anbietet.

Die Expertensystemschale knoX bietet als hybride Schale die drei Formen Frames, Regeln und Prozeduren samt den zugehörigen Inferenzstrategien an. Die in den früheren Abschnitten beschriebenen Elemente und Beziehungen lassen sich damit relativ leicht in das technische Entwurfsmodell umsetzen.

An einigen Beispielen soll diese Umsetzung beschrieben werden.

FRAME UND INSTANZ

```
frame handlung
slot weiter :     instance(handlung);
     allgemein :  instance(handlung);
     praezis :    instance(handlung);
     apparatur :  instance(apparatur);
     theorie :    instance(theorie);
```

```
instance hh1142 of handlung
    weiter = hh12
    allgemein = hh114
    praezis = hh11421
    apparatur = aa123
    theorie = tt1.
```

REGELN

Regeln werden vorwiegend dazu verwendet, Werte von Instanzen abzuleiten. Dazu werden hier Vorwärtsregeln eingesetzt:

<Vorwärtsregel> ::= *whenfound* <Prämisse> *then* <Konklusion>

Die Regel

```
whenfound anfrage(akt_zustand) = praezis then start(praezis).
```

z. B. bestimmt, daß das Behaviour `praezis` gestartet wird, falls die Instanz, die den aktuellen Zustand des Systems beschreibt, in dem Slot `anfrage` den Wert `praezis` enthält.
Dieser Slot enthält genau dann diesen Wert, wenn aus dem zuletzt angebotenen Menü der Punkt `praezis` vom Nutzer gewählt wurde.

PROZEDUR

```
behaviour praezis
is get_value(akt_zustand,knoten,Knoten) and

     start(bildschirm_ausgeben(Knoten,praez)) and
     get_value(akt_zustand,gebiet,Gebiet) and
     menue_angeben(Gebiet,Menue) and
     put_value(akt_zustand,anbieten,Menue)
     ).
```

Der Start des Behaviours `praezis` bewirkt, daß

- der Variablen `Knoten` der Wert des aktuellen Knotens zugeordnet wird, d.h. der Eintrag, der im Slot `Knoten` der Instanz `akt_zustand` steht,
- das Behaviour `bildschirm_ausgeben` aufgerufen wird,
- der Variablen `Gebiet` der Slot `gebiet` der Instanz `akt_zustand` zugeordnet wird,
- das Behaviour `menue_angeben` gestartet wird und
- der Wert der Variablen `Menue` in den Slot `anbieten` der Instanz `akt_zustand` eingetragen wird.

Die Umsetzung und Implementierung wird beschrieben im Benutzerhandbuch und der Dokumentation des Systems (Held & Maier-Schicht, 1994).

3.4 Fazit der Arbeit mit einem Expertensystem

Ein weiterer Aspekt der Entwicklung des Expertensystems ergab sich im weiteren Verlauf der Arbeit. K. Degele faßt ihn in ihrem Buch so zusammen:

> *"Die Nutzung von Expertensystemen beginnt nicht mit dem betrieblichen Einsatz, sondern mit der Zieldefinition. Damit steht die Frage nach dem Nutzungsmodus im Zentrum. Der Nutzen von Expertensystemen kann in dreifacher Weise festgemacht werden, nämlich in der Be-Nutzung, der Problemdurchdringung und der Aus-Nutzung.*
>
> *Diese Unterscheidung verfolgt den Zweck, das emergente Potential von Expertensystemen im Hinblick auf ihre Unterstützung beim Prozeß des Problemlösens herauszustreichen.*
>
> *Es besteht darin, daß Expertensysteme aus systemischer Perspektive als Problemlösungsmethode eingesetzt werden können und wurzelt im ebenenübergreifenden Anspruch der Zieldefinition.*
>
> *Expertensysteme werden konzipiert, um einen bestimmten Nutzen zu bringen. Dazu sind unterschiedliche Faktoren, Ebenen, Funktionen und Randbedingungen zu berücksichtigen. Zur sinnvollen Umsetzung der einzelnen Teilaspekte ist ein modularer Ansatz erforderlich. ...*
>
> *Dieser modulare Aufbau ist der Entwicklung von Expertensystemen inhärent. Gleichzeitig bietet er Nutzern die Möglichkeit der "Unterminierung" der Zielvorstellung, indem die ursprüngliche Rolle des Expertensystems (...) umdefiniert wird."* (Degele, 1994, S.241)

Für das Projekt bedeutete das den Zwang zur genauen Definition von Grundlagen, Aufbau und Funktionsweise des Expertensystems. Dies führte z. B. zur Diskussion der Frage, ob die Zwischentexte auch als hierarchisch geordnete Struktur zu betrachten seien (also als Knoten mit den daraus entstehenden Konsequenzen, wie Auswahlmöglichkeit von diesen

"Knoten" aus, welches Menü wird angeboten usw.) oder Betrachtung als Zwischentext ohne Knoteneigenschaften.
Durch den modularen Aufbau wurde die Pflege, die Änderungen und Erweiterungen des Systems stark vereinfacht. Zusätzliche Knoten konnten der Wissensbasis einfach hinzugefügt werden, ohne daß sich bei den Prozeduren oder Inferenzmechanismen etwas geändert hätte. Umstrukturierungen innerhalb der Hierarchien, wie eine stärkere Differenzierung oder ähnliches, waren sehr benutzerfreundlich zu erreichen.

3.5 Möglichkeiten der Weiterentwicklung

Es wurde bereits erwähnt, daß die Verwendung der Expertensystemschale knoX nur **eine** Möglichkeit der Realisierung des hier beschriebenen Experimentalsystems darstellt. Auch wird von dem Leistungsumfang, den knoX anbietet, bisher nur wenig genutzt. So ist es mit den vorhandenen Schlußfolgerungsverfahren durchaus möglich, das bisherige Experimentalsystem in Richtung eines adaptiven Lehrsystems, das sich an den Benutzer anpassen kann, weiterzuentwickeln.

Andererseits gibt es aber auch eine Reihe von Anforderungen, die von knoX noch nicht oder nur unzureichend erfüllt werden. Diese betreffen im wesentlichen die Mensch-Maschine-Schnittstelle, die in knoX bisher nur rudimentär realisiert ist. Das primäre Entwicklungsziel von knox war die umfassende Unterstützung der wichtigsten Wissensrepräsentationsformalismen inklusive der dazugehörigen Verarbeitungsmechanismen und weniger die Realisierung einer optimalen Benutzerschnittstelle, die ohne Zweifel für die Akzeptanz eines Systems sehr wichtig ist. Mit Fenstern, Menüs und speziellen Dialogboxen sind zwar die wesentlichsten Elemente einer Mensch-Maschine-Schnittstelle vorhanden, sie entsprechen jedoch nicht immer den Anforderungen, die heute an eine graphische Oberfläche, wie z. B. Windows, gestellt werden. Hier wären entsprechende Erweiterungen, wie das gleichzeitige Arbeiten mit mehreren Fenstern. Scrollen in einem Fenster, Unterstützung durch Hypertext usw., wünschenswert.

Zur Verbesserung der Verarbeitungsgeschwindigkeit sollten Texte, die zum wesentlichen Bestandteil des Experimentalsystems gehören, auf eine externe Datenbank ausgelagert werden. Dadurch würde die Organisation der Wissensbasis vereinfacht und der Arbeitsspeicher entlastet. Prinzipiell ist dies mit knoX möglich, es wurde bisher aber noch nicht realisiert.

Das Hauptproblem bei fast allen Erweiterungen stellt die Laufzeitumgebung von knoX dar. KnoX wurde in MProlog entwickelt, das neben vielen Vorteilen nur sehr eingeschränkte Erweiterungsmöglichkeiten bietet. Es ist nur unter DOS lauffähig, und die Kommunikation mit anderen Applikationen (z. B. Aufruf von Programmen einer anderen Programmiersprache) führt häufig zu Problemen.

Deshalb sollten folgende Punkte bei einer Weiterentwicklung des Experimentalsystems berücksichtigt werden:
- Umstellung von knoX auf ifProlog;
- Einbindung einer Datenbank zur Verwaltung der Texte;
- Erweiterung der Einbindung von Grafik in die Präsentation;
- Grafik zur Orientierungshilfe innerhalb des Systems;
- Scrollen für Text und Grafik;
- Anpassung der grafischen Oberfläche, Anbindung an Windows;
- Adaption des XPS an den Benutzer (Benutzerprofil).

Literatur

Brachman, R. J. (1983). What are expert systems ? In: F. Hayes-Roth et al. (1983). Building Expert Systems (S. 31-57). Reading, Ma: Addison Wesley.

Buchanan, B. G. et al. (1983). Constructing an expert system. In: F. Hayes-Roth et al. (1983). Building Expert Systems (S. 127-167). Reading, Ma: Addison Wesley.

Burger, C. & Mehling, D. (1992). Computer aided knowledge acquisition (CAKE): Eine Notwendigkeit für modellbasierte XPS-Entwicklungen - Erfahrungen aus einer KADS Anwendung im Bereich Produktionsleittechnik (KI. Nr. 1) (S. 51-55). München: Oldenbourg.

Bylander, T. & Chandrasekaran, B. (1987). Generic tasks for knowledge-based reasoning: The "Right" level of abstraction for knowledge acquisition. Technical report 87-TB-KNOWAC: Ohio State University.

Degele, K. (1994). Der überforderte Computer. Zur Soziologie menschlicher und künstlicher Intelligenz. Frankfurt/M.: Campus Forschung.

Held, Th. & Maier-Schicht, B. (1994). Benutzerhandbuch und Dokumentation eines Experimentalsystems auf der Basis der Expertensystemschale knoX. Heidelberg: Psych. Inst. d. Universität (Arbeiten aus dem SFB 245 "Sprache und Situation" HD/MA).

Karbach, W. & Linster, M. (1990). Wissensaquisition für Expertensysteme: Techniken, Modelle und Werkzeuge. München: Hanser.

Lehner, K. (1990). Wissensbasierte Lehrsysteme. München: Oldenbourg.

Pimentel, E. (1992). Entwurf und Realisierung eines Informationssystems am Beispiel der Destillation. Diplomarbeit, Karlsruhe: Universität.

Puppe, F. (1988). Einführung in Expertensysteme. Berlin, Heidelberg: Springer.

Puppe, F. (1990). Problemlösungsmethoden in Expertensystemen. Berlin, Heidelberg: Springer.

Raulefs, P. (1982). Expertensysteme. In W. Brauer (Hrsg.), Informatik Fachberichte Nr. 59, S. 61-98. Berlin, Heidelberg: Springer.

Schachter-Radig, M. & Krickhahn, R. (1989). KBSM - Strukturen und Modelle: Basis für einen wiederverwendbaren Entwurf wissensbasierter Systeme. In Brauer, W. & Freska, C. (Hrsg.). Wissensbasierte Systeme. Tagungsband 3. internationaler GI-Kongress (S. 426-435). Berlin, Heidelberg: Springer.

Sonderforschungsbereich 245 "Sprache und Situation" (1991). Finanzierungsantrag 1992 - 1993 - 1994.

Theiss, G. (1988). Entwurf und Implementierung einer hybriden Expertensystemschale. Diplomarbeit, Karlsruhe: Universität.

Theiss, G. (1992). Benutzerhandbuch der hybriden Expertensystemschale knoX. Karlsruhe: Universität.

Waterman, D. A. (1986). A guide to expert systems. Reading, Ma: Addison Wesley.

Verzeichnis der Arbeiten aus dem Sonderforschungsbereich 245

Heidelberg/Mannheim

Nr. 1 Schwarz, S., Wagner, F. & Kruse, L.: Soziale Repräsentation und Sprache: Gruppenspezifische Wissensbestände und ihre Wirkung bei der sprachlichen Konstruktion und Rekonstruktion geschlechtstypischer Episoden. Februar 1989.

Nr. 2 Wintermantel, M., Laux, H. & Fehr, U.: Anweisung zum Handeln: Bilder oder Wörter. März 1989.

Nr. 3 Herrmann, Th., Dittrich, S., Hornung-Linkenheil, A., Graf, R. & Egel, H.: Sprecherziele und Lokalisationssequenzen: Über die antizipatorische Aktivierung von Wie-Schemata. April 1989.

Nr. 4 Schwarz, S., Weniger, G. & Kruse, L. (unter Mitarbeit von R. Kohl): Soziale Repräsentation und Sprache: Männertypen: Überindividuelle Wissensbestände und individuelle Kognitionen. Juni 1989.

Nr. 5 Wagner, F., Theobald, H., Heß, K., Schwarz, S. & Kruse, L.: Soziale Repräsentation zum Mann: Gruppenspezifische Salienz und Strukturierung von Männertypen. Juni 1989.

Nr. 6 Schwarz, S. & Kruse, L.: Soziale Repräsentation und Sprache: Gruppenspezifische Unterschiede bei der sprachlichen Realisierung geschlechtstypischer Episoden. Juni 1989.

Nr. 7 Dorn-Mahler, H., Grabowski-Gellert, J., Funk-Müldner, K. & Winterhoff-Spurk, P.: Intonation bei Aufforderungen. Teil I: Theoretische Grundlagen. Juni 1989.

Nr. 8 Dorn-Mahler, H., Grabowski-Gellert, J., Funk-Müldner, K. & Winterhoff-Spurk, P.: Intonation bei Aufforderungen. Teil II: Eine experimentelle Untersuchung. Dezember 1989.

Nr. 9 Sommer, C. M. & Graumann, C. F.: Perspektivität und Sprache: Zur Rolle von habituellen Perspektiven. August 1989.

Nr. 10 Grabowski-Gellert, J. & Winterhoff-Spurk, P.: Schreiben ist Silber, Reden ist Gold. August 1989.

Nr. 11 Graf, R. & Herrmann, Th.: Zur sekundären Raumreferenz: Gegenüberobjekte bei nicht-kanonischer Betrachterposition. Dezember 1989.

Nr. 12 Grosser, Ch. & Mangold-Allwinn, R.: Objektbenennung in Serie: Zur partnerorientierten Ausführlichkeit von Erst- und Folgebenennungen. Dezember 1989.

Nr. 13 Grosser, Ch. & Mangold-Allwinn, R.: Zur Variabilität von Objektbenennungen in Abhängigkeit von Sprecherzielen und kognitiver Kompetenz des Partners. Dezember 1989.

Nr. 14 Gutfleisch-Rieck, I., Klein, W., Speck, A. & Spranz-Fogasy, Th.: Transkriptionsvereinbarungen für den Sonderforschungsbereich 245 „Sprechen und Sprachverstehen im sozialen Kontext". Dezember 1989.

Nr. 15 Herrmann, Th.: Vor, hinter, rechts und links: das 6H-Modell. Psychologische Studien zum sprachlichen Lokalisieren. Dezember 1989.

Nr. 16 Dittrich, S. & Herrmann, Th.: „Der Dom steht hinter dem Fahrrad." – Intendiertes Objekt oder Relatum? März 1990.

Nr. 17 Kilian, E., Herrmann, Th., Dittrich, S. & Dreyer, P.: Was- und Wie-Schemata beim Erzählen. Mai 1990.

Nr. 18 Herrmann, Th. & Graf, R.: Ein dualer Rechts-links-Effekt. Kognitiver Aufwand und Rotationswinkel bei intrinsischer Rechts-links-Lokalisation. August 1990.

Nr. 19 Wintermantel, M.: Dialogue between expert and novice: On differences in knowledge and means to reduce them. August 1990.

Nr. 20 Graumann, C. F.: Perspectivity in Language and Language Use. September 1990.

Nr. 21 Graumann, C. F.: Perspectival Structure and Dynamics in Dialogues. September 1990.

Nr. 22 Hofer, M., Pikowsky, B., Spranz-Fogasy, Th. & Fleischmann, Th.: Mannheimer Argumentations-Kategoriensystem (MAKS). Mannheimer Kategoriensystem für die Auswertung von Argumentationen in Gesprächen zwischen Müttern und jugendlichen Töchtern. Oktober 1990.

Nr. 23 Wagner, F., Huerkamp, M., Jockisch, H. & Graumann, C. F.: Sprachlich realisierte soziale Diskriminierungen: empirische Überprüfung eines Modells expliziter Diskriminierung. Oktober 1990.

Nr. 24 Rettig, H., Kiefer, L., Sommer, C. M. & Graumann, C. F.: Perspektivität und soziales Urteil: Wenn Versuchspersonen ihre Bezugsskalen selbst konstruieren. November 1990.

Nr. 25 Kiefer, L., Sommer, C. M. & Graumann, C. F.: Perspektivität und soziales Urteil: Klassische Urteilseffekte bei individueller Skalenkonstruktion. November 1990.

Nr. 26 Hofer, M., Pikowsky, B., Fleischmann, Th. & Spranz-Fogasy, Th.: Argumentationssequenzen in Konfliktgesprächen zwischen Müttern und Töchtern. November 1990.

Nr. 27 Funk-Müldner, K., Dorn-Mahler, H. & Winterhoff-Spurk, P.: Kategoriensystem zur Situationsabhängigkeit von Aufforderungen im betrieblichen Kontext. Dezember 1990.

Nr. 28 Groeben, N., Schreier, M. & Christmann, U.: Argumentationsintegrität (I): Herleitung, Explikation und Binnenstrukturierung des Konstrukts. Dezember 1990.

Nr. 29 Blickle, G. & Groeben, N.: Argumentationsintegrität (II): Zur psychologischen Realität des subjektiven Wertkonzepts – ein experimenteller Überprüfungsansatz am Beispiel ausgewählter Standards. Dezember 1990.

Nr. 30 Schreier, M. & Groeben, N.: Argumentationsintegrität (III): Rhetorische Strategien und Integritätsstandards. Dezember 1990.

Nr. 31 Sachtleber, S. & Schreier, M.: Argumentationsintegrität (IV): Sprachliche Manifestationen argumentativer Unintegrität – ein pragmalinguistisches Beschreibungsmodell und seine Anwendung. Dezember 1990.

Nr. 32 Dietrich, R., Egel, H., Maier-Schicht, B. & Neubauer, M.: ORACLE und die Analyse des Äußerungsaufbaus. Februar 1991.

Nr. 33 Nüse, R., Groeben, N. & Gauler, E.: Argumentationsintegrität (V): Diagnose argumentativer Unintegrität – (Wechsel-)wirkungen von Komponenten subjektiver Werturteile über argumentative Sprechhandlungen. März 1991.

Nr. 34 Christmann, U. & Groeben, N.: Argumentationsintegrität (VI): Subjektive Theorien über Argumentieren und Argumentationsintegrität – Erhebungsverfahren, inhaltsanalytische und heuristische Ergebnisse. März 1991.

Nr. 35 Graf, R., Dittrich, S., Kilian, E. & Herrmann, Th.: Lokalisationssequenzen: Sprecherziele, Partnermerkmale und Objektkonstellationen (Teil II). Drei Erkundungsexperimente. März 1991.

Nr. 36 Hofer, M., Pikowsky, B., &. Fleischmann, Th.: Jugendliche unterschiedlichen Alters im argumentativen Konfliktgespräch mit ihrer Mutter. März 1991.

Nr. 37 Herrmann, Th., Graf, R. & Helmecke, E.: „Rechts" und „Links" unter variablen Betrachtungswinkeln: Nicht-Shepardsche Rotationen. April 1991.

Nr. 38 Herrmann, Th. & Grabowski, J.: Mündlichkeit, Schriftlichkeit und die nicht-terminalen Prozeßstufen der Sprachproduktion. Februar 1992.

Nr. 39 Thimm, C. & Kruse, L.: Dominanz, Macht und Status als Elemente sprachlicher Interaktion. Mai 1991.

Nr. 40 Thimm, C. & Kruse, L.: Sprachliche Effekte von Partnerhypothesen in dyadischen Situationen. September 1993.

Nr. 41 Thimm, C., Maier, S. & Kruse, L.: Statusrelationen in dyadischen Kommunikationssituationen: Zur Rolle von Partnerhypothesen. April 1994.

Nr. 42 Funk-Müldner, K., Dorn-Mahler, H. & Winterhoff-Spurk, P.: Nonverbales Verhalten beim Auffordern – ein Rollenspielexperiment. Dezember 1991.

Nr. 43 Dorn-Mahler, H., Funk-Müldner, K. & Winterhoff-Spurk, P.: AUFF_{KO} – Ein inhaltsanalytisches Kodiersystem zur Analyse von komplexen Aufforderungen. Oktober 1991.

Nr. 44 Herrmann, Th.: Sprachproduktion und erschwerte Wortfindung. Mai 1992.

Nr. 45 Grabowski, J., Herrmann, Th. & Weiß, P.: Wenn „vor" gleich „hinter" ist – zur multiplen Determination des Verstehens von Richtungspräpositionen. Juni 1992.

Nr. 46 Barattelli, St., Koelbing, H.G. & Kohlmann, U.: Ein Klassifikationssystem für komplexe Objektreferenzen. September 1992.

Nr. 47 Haury, Ch., Engelbert, H. M., Graf, R. & Herrmann, Th.: Lokalisationssequenzen auf der Basis von Karten– und Straßenwissen: Erste Erprobung einer Experimentalanordnung. August 1992.

Nr. 48 Schreier, M. & Czemmel, J.: Argumentationsintegrität (VII): Wie stabil sind die Standards der Argumentationsintegrität ? August 1992.

Nr. 49 Engelbert, H. M., Herrmann, Th. & Haury, Ch.: Ankereffekte bei der sprachlichen Linearisierung. Oktober 1992.

Nr. 50 Spranz-Fogasy, Th.: Bezugspunkte der Kontextualisierung sprachlicher Ausdrücke in Interaktionen. Ein Konzept zur analytischen Konstitution von Schlüsselwörtern. November 1992.

Nr. 51 Kiefer, M., Barattelli, St. & Mangold-Allwinn, R.: Kognition und Kommunikation: Ein integrativer Ansatz zur multiplen Determination der lexikalischen Spezifität der Objektklassenbezeichnung. Februar 1993.

Nr. 52 Spranz-Fogasy, Th.: Beteiligungsrollen und interaktive Bedeutungskonstitution. Februar 1993.

Nr. 53 Schreier, M. & Groeben, N.: Argumentationsintegrität (VIII): Zur psychologischen Realität des subjektiven Wertkonzepts. Eine experimentelle Überprüfung für die 11 Standards integeren Argumentierens. Dezember 1992.

Nr. 54 Sommer, C. M., Freitag, B. & Graumann, C. F.: Aggressive Interaction in Perspectival Discourse. März 1993.

Nr. 55 Huerkamp, M., Jockisch, H., Wagner, F. & Graumann, C. F.: Facetten expliziter sprachlicher Diskriminierung: Untersuchungen von Ausländer-Diskriminierungen anhand einer deutschen und einer ausländischen Stichprobe. Februar 1993.

Nr. 56 Rummer, R., Grabowski, J., Hauschildt, A. & Vorwerg, C.: Reden über Ereignisse: Der Einfluß von Sprecherzielen, sozialer Nähe und Institutionalisiertheitsgrad auf Sprachproduktionsprozesse. April 1993.

Nr. 57 Blickle, G.: Argumentationsintegrität (IX): Personale Antezedensbedingungen der Diagnose argumentativer Unintegrität. Juli 1993.

Nr. 58 Herrmann, Th., Buhl, H.M., Schweizer, K. & Janzen, G.: Zur repräsentationalen Basis des Ankereffekts. Kognitionspsychologische Untersuchungen zur sprachlichen Linearisierung. September 1993.

Nr. 59 Carroll, M.: Keeping spatial concepts on track in text production. A comparative analysis of the use of the concept path in descriptions and instructions in German. Oktober 1993.

Nr. 60 Speck, A.: Instruieren im Dialog. Oktober 1993.

Nr. 61 Herrmann, Th. & Grabowski, J.: Das Merkmalsproblem und das Identitätsproblem in der Theorie dualer, multimodaler und flexibler Repräsentationen von Konzepten und Wörtern (DMF-Theorie). November 1993.

Nr. 62 Rummer, R., Grabowski, J. & Vorwerg, C.: Zur situationsspezifischen Flexibilität zentraler Voreinstellungen bei ereignisbezogenen Sprachproduktionsprozessen. November 1993.

Nr. 63 Christmann, U. & Groeben, N.: Argumentationsintegrität (X): Realisierung argumentativer Redlichkeit und Reaktionen auf Unredlichkeit. November 1993.

Nr. 64 Christmann, U. & Groeben, N.: Argumentationsintegrität (XI): Retrognostische Überprüfung der Handlungsleitung subjektiver Theorien über Argumentationsintegrität bei Kommunalpolitikern/innen. November 1993.

Nr. 65 Schreier, M.: Argumentationsintegrität (XII): Sprachliche Manifestationsformen argumentativer Unintegrität in Konfliktgesprächen. Dezember 1993.

Nr. 66 Christmann, U., Groeben, N. & Küppers, A.: Argumentationsintegrität (XIII): Subjektive Theorien über Erkennen und Ansprechen von Unintegritäten im Argumentationsverlauf. Dezember 1993.

Nr. 67 Christmann, U. & Groeben, N.: Argumentationsintegrität (XIV): Der Einfluß von Valenz und Sequenzstruktur argumentativer Unintegrität auf kognitive und emotionale Komponenten von Diagnose- und Bewertungsreaktionen. Dezember 1993.

Nr. 68 Schreier, M., Groeben, N. & Mlynski, G.: Argumentationsintegrität (XV): Der Einfluß von Bewußtheitsindikatoren und (Un-)Höflichkeit auf die Rezeption argumentativer Unintegrität. Februar 1994.

Nr. 69 Thimm, C., Rademacher, U. & Augenstein, S.: "Power-Related Talk (PRT)": Ein Auswertungsmodell. Januar 1994.

Nr. 70 Kiefer, L., Rettig, H., Sommer, C. M. & Graumann, C. F.: Perspektivität und soziales Urteil: Vier Sichtweisen zum Thema "Ausländerstop". Januar 1994.

Nr. 71 Graumann, C. F.: Discriminatory Discourse. Conceptual and methodological problems. 1994.

Nr. 72 Huerkamp, M.: SAS-Makros zur Analyse und Darstellung mehrdimensionaler Punktekonfigurationen. 1994.

Nr. 73 Galliker, M., Huerkamp, M., Wagner, F. & Graumann, C. F.: Funktionen expliziter sprachlicher Diskriminierung: Validierung der Kernfacetten des Modells sprachlicher Diskriminierung. 1994.

Nr. 74 Buhl, H.M., Schweizer, K. & Herrmann, Th.: Weitere Untersuchungen zum Ankereffekt. April 1994.

Nr. 75 Herrmann, Th.: Psychologie ohne 'Bedeutung'? Zur Wort-Konzept-Relation in der Psychologie. Mai 1994.

Nr. 76 Neubauer, M., Hub, I. & Thimm, C.: Transkribieren mit LaTeX: Transkriptionsregeln, Eingabeverfahren und Auswertungsmöglichkeiten. Mai 1994.

Nr. 77 Thimm, C. & Augenstein, S.: Sprachliche Effekte in hypothesengeleiteter Interaktion: Durchsetzungsstrategien in Aushandlungsgesprächen. Mai 1994.

Nr. 78 Sommer, C. M., Rettig, H., Kiefer, L. & Frankenhauser, D.: "Germany will be one single concrete block ...". Point of View and Reference to Topic Aspects in Adversial Discussions on Immigration. September 1994.

Nr. 79 Maier, S. & Kruse, L.: Ein Design zur Erfassung einer dialogischen Kommunikationssituation: Das Experiment "Terminabsprache". November 1994.

Nr. 80 Grabowski, J.: Schreiben als Systemregulation – Ansätze einer psychologischen Theorie der schriftlichen Sprachproduktion. Oktober 1994.

Nr. 81 Hermanns, F.: Schlüssel-, Schlag- und Fahnenwörter. Zu Begrifflichkeit und Theorie der lexikalischen «politischen Semantik». Dezember 1994.

Nr. 82 Kiefer, L., Rettig, H., Frankenhauser, D., Sommer, C. M. & Graumann, C. F.: Perspektivität und Persuasion: Effektivität perspektivenrelevanter Persuasionsstrategien. Dezember 1994.

Nr. 83 Liebert, W.-A.: Das analytische Konzept "Schlüsselwort" in der linguistischen Tradition. Dezember 1994.

Nr. 84 Buhl, H. M., Schweizer, K. & Herrmann, Th.: Der Einfluß von Räumlichkeit und Reizmodalität auf den Ankereffekt. Dezember 1994.

Nr. 85 Koelbing, H.G., Mangold-Allwinn, R., Barattelli, St., Kohlmann, U. & Stutterheim, C. v.: Welchen Einfluß hat der Ausführende auf den Instruierenden ? Dezember 1994.

Nr. 86 Held, Th. & Maier-Schicht, B.: Benutzerhandbuch und Dokumentation eines Experimentalsystems auf der Basis der Expertensystemschale knoX. Dezember 1994.

Nr. 87 Maier-Schicht, B., Theiss, G. & Held, Th.: Ein Expertensystem als Experimentalsystem. Februar 1995.