Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Document
D-92-11

Möglichkeiten der Wissensmodellierung für technische Diagnose-Expertensysteme

Kerstin Becker

Juni 1992

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
D-6750 Kaiserslautern, FRG
Tel.: (+49 631) 205-3211/13
Fax: (+49 631) 205-3210

Stuhlsatzenhausweg 3
D-6600 Saarbrücken 11, FRG
Tel.: (+49 681) 302-5252
Fax: (+49 681) 302-5341

Deutsches Forschungszentrum für Künstliche Intelligenz

The German Research Center for Artificial Intelligence (Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI) with sites in Kaiserslautern and Saarbrücken is a non-profit organization which was founded in 1988. The shareholder companies are Atlas Elektronik, Daimler Benz, Fraunhofer Gesellschaft, GMD, IBM, Insiders, Mannesmann-Kienzle, Philips, SEMA Group Systems, Siemens and Siemens-Nixdorf. Research projects conducted at the DFKI are funded by the German Ministry for Research and Technology, by the shareholder companies, or by other industrial contracts.

The DFKI conducts application-oriented basic research in the field of artificial intelligence and other related subfields of computer science. The overall goal is to construct *systems with technical knowledge and common sense* which - by using AI methods - implement a problem solution for a selected application area. Currently, there are the following research areas at the DFKI:

- Intelligent Engineering Systems
- Intelligent User Interfaces
- Intelligent Communication Networks
- Intelligent Cooperative Systems.

The DFKI strives at making its research results available to the scientific community. There exist many contacts to domestic and foreign research institutions, both in academy and industry. The DFKI hosts technology transfer workshops for shareholders and other interested groups in order to inform about the current state of research.

From its beginning, the DFKI has provided an attractive working environment for AI researchers from Germany and from all over the world. The goal is to have a staff of about 100 researchers at the end of the building-up phase.

Prof. Dr. Gerhard Barth
Director

Feature based Integration of CAD and CAPP

Kerstin Becker

DFKI-D-92-11

Diese Arbeit wurde von Prof. Michael M. Richter und Herrn Dipl.-Inform. Christoph Klauck betreut

Diese Arbeit wurde finanziell unterstützt durch das Bundesministerium für Forschung und Technologie (FKZ ITW-8902 C4).

Inhaltsverzeichnis

Kapitel 1

Einleitung

Ein wichtiges Gebiet der Künstlichen Intelligenz, das vor allem stark anwendungsorientiert ist, stellen Expertensysteme dar. Mit diesen Systemen können verschiedene Aufgaben, wie die Diagnose, Planung und Konfiguration, in den unterschiedlichsten Gebieten, wie dem Maschinenbau oder der Medizin, gelöst werden. Die verschiedenen Aufgaben können klar voneinander unterschieden werden. Aber auch bei den Einsatzgebieten treten deutliche Unterschiede auf. Diese Unterschiede beruhen zum einen auf dem Einsatzgebiet selbst (es ist einfach ein Unterschied, ob man einen Menschen, oder ein Auto untersucht, unabhängig von der Aufgabe), zum anderen beruhen sie jedoch darauf, in welcher Form das Wissen des jeweiligen Einsatzgebietes vorliegt und wie es modelliert werden kann.

Im Rahmen des ARC-TEC-Projektes am DFKI Kaiserslautern, unter Leitung von Prof. Dr. M. M. Richter, das sich mit der Akquisition, Repräsentation und Compilation von Wissen im Zusammenhang mit Planung, Konfiguration und Diagnose technischer Systeme und Abläufe beschäftigt, setzt sich diese Arbeit speziell mit technischem Diagnosewissen auseinander.

Folgende Fragen sollen im Verlauf der Arbeit geklärt werden:

- Welche Möglichkeiten der Wissenmodellierung existieren für technisches Diagnosewissen?
- Inwieweit wurden die verschiedenen Modellierungskonzepte bislang eingesetzt?
- In welchen Bereichen eignet sich welche Modellierungsart?
- Welche Verbesserungsvorschläge können gemacht werden?

Um diese Fragen zu beantworten, befaßt sich diese Arbeit in Kapitel 2 zunächst mit den verschiedenen Wissensmodellierungsarten — ihren Eigenschaften, Unterschieden und eventuellen Überschneidungen.

Dabei werden folgende fünf Modellierungskonzepte unterschieden:

- *heuristisch*
- *modellbasiert*
- *fallbasiert*
- *statistisch*
- *Entscheidungsbaum* oder *Entscheidungstabelle*

Anschließend werden in Kapitel 3 zu den unterschiedlichen Wissensmodellierungsarten bereits existierende Diagnose–Expertensysteme vorgestellt.

Im einzelnen sind dies die folgenden Systeme: MED2, MegaFilEx, ein System zur Diagnose an CNC–Maschinen, Davis' System, QUASIMODIS, ein System zur Diagnose an flexiblen Fertigungssystemen, PATDEX, PROSPECTOR und DIWA.

- MED2, eine Shell zur Diagnose sowohl im technischen als auch im medizinischen Bereich
- MegaFilEx, ein System zur Diagnose spezieller Plattenspeicher
- ein System zur Diagnose an CNC–Maschinen
- Davis' System, ein System zur Diagnose einfacher digitaler Hardware
- QUASIMODIS, ein System zur Diagnose in technischen Anwendungsbereichen
- ein System zur Diagnose an flexiblen Fertigungssystemen
- PATDEX, ein System zur Diagnose in technischen Anwendungsbereichen
- PROSPECTOR, ein System zur Erkennung geologischer Formationen, die Bodenschätze vermuten lassen
- DIWA, ein System zur Diagnose in technischen Anwendungsbereichen

In Kapitel 4 wird eine Bewertung der verschiedenen Wissensmodellierungsarten vor dem Hintergrund der technischen Diagnose nach unterschiedlichen Gesichtspunkten vorgenommen. Zu jeder Modellierungsart werden die Vor– und Nachteile diskutiert, die auftreten, wenn man sie als alleiniges Modellierungskonzept für eine Domäne verwendet. Außerdem werden Vorschläge gemacht, wann man sie sinnvoll und wann man sie weniger sinnvoll einsetzen kann. Wegen der Probleme, die bei der Vorgehensweise, nur eine Modellierungsart bei einem System zugrundezulegen, auftreten, wird als Alternative die Kombination mehrerer Wissensarten in einem System, ein sogenannter Hybrid, vorgeschlagen. Durch diese Kombination werden die Vorteile der einzelnen Wissensformen akkumuliert und die Nachteile kompensiert. Abschließend wird das System MOLTKE 3 als Beispiel eines Hybrides vorgestellt.

Kapitel 2

Wissensmodellierung in technischen Diagnose–Expertensystemen

Diagnose–Expertensysteme sind wissensbasierte Systeme, für die sich im Zusammenhang mit der Wissensmodellierung folgende wichtige Fragen stellen:

1.) Wie kann Wissen aussehen, und wie kann es modelliert werden?

2.) Wie wird dieses Wissen repräsentiert?

3.) Wie wird dieses Wissen verarbeitet?

In dieser Arbeit liegt das Hauptaugenmerk auf dem ersten Punkt. Auf den zweiten Punkt wird teilweise bei der Vorstellung der unterschiedlichen Diagnose–Expertensysteme in Kapitel 3 eingegangen. Punkt drei ist nicht Thema dieser Arbeit, sondern wird in [Bec92] genau behandelt.

Um die erste Frage beantworten zu können, muß zunächst geklärt werden, über welches Wissen ein menschlicher Experte für die Diagnoseerstellung verfügt.
Prinzipiell kann man das beim Experten vorliegende Wissen in zwei Arten unterscheiden:

- *Tiefenwissen* und
- *Erfahrungswissen*[1]

Unter Tiefenwissen wird das Wissen verstanden, über das beispielsweise ein Servicetechniker, der gerade mit seiner Ausbildung fertig ist, verfügt. Es handelt sich um das Wissen, das z. B. den Aufbau einer Maschine ausmacht, erklärt, wie deren einzelne Komponenten zusammenhängen und wie die Maschine insgesamt funktioniert. Der Servicetechniker kennt zu Beginn sicherlich schon ein paar Fehler, die an der Maschine auftreten können, da er einige Fehlerbeschreibungen schon in seiner Ausbildung vermittelt bekommt. Im

[1] Erfahrungswissen wird in der Literatur auch häufig als *flaches* Wissen bezeichnet [Coh85].

großen und ganzen jedoch verfügt er zu Beginn seiner Tätigkeit lediglich über Wissen, mit dem er sich die Ursachen für vorliegende Fehler herleiten kann.

Unter Erfahrungswissen versteht man das Wissen, das der Servicetechniker, um bei diesem Beispiel zu bleiben, im Laufe seiner Berufspraxis durch Anwendung seines bisherigen Wissens hinzugewinnt, und somit schließlich zu einem Experten auf seinem Gebiet wird. Dieses Wissen liegt einerseits in Form von bereits gelösten Fällen vor, andererseits jedoch auch einfach in Form von dem, was man schlichtweg als Erfahrung bezeichnet und das es ihm ermöglicht, intuitive Entscheidungen zu treffen, die, obgleich nicht unbedingt fundiert, zu einer richtigen Diagnose führen. Das bedeutet, der Experte ist in der Lage, Beziehungen zwischen Symptomen und Zwischen- und Enddiagnosen herzustellen, die aus dem fundierten Tiefenwissen nicht so ohne weiteres herauszuziehen sind. Natürlich ergibt sich das Tiefenwissen auch aus den bereits gelösten Fällen, wird aber nicht so explizit in Form eines Falles im Gedächtnis behalten.

Der Servicetechniker gewinnt im Laufe der Zeit nicht nur an Erfahrung, sondern auch sein Tiefenwissen gewinnt durch die Berufspraxis an Umfang. So ist sicherlich festzustellen, daß der Servicetechniker im Verlaufe seiner Ausbildung nicht sämtliche Kausalitäten einer Maschine kennenlernt, sondern daß er mit zunehmender Praxis auch zusätzliche, fundierte Kenntnisse in seinen Wissensschatz aufnimmt. Alles in allem macht ihn dies nach einigen Jahren schließlich zu einem Experten.

Die Unterschiede zwischen den Wissensformen sind theoretisch deutlich (man kann genau sagen, was ein Fall ist, was Assoziationen sind, etc.). In der Realität, also beim Menschen, ist es nicht so einfach, zu sagen, dieses Wissen liegt bei ihm in Form eines Falls vor, jenes Wissen in Form einer Assoziation, etc. . Die Grenzen sind beim Menschen recht verschwommen. Noch schwieriger wird es, wenn man bei der Vorgehensweise eines Technikers oder Experten versucht nachzuvollziehen, wann er auf welche Wissensformen zugreift, da auch diese Abgrenzungen sehr verschwommen sind. Aber auch dies wird in dieser Arbeit nicht abgehandelt, sondern sollte in Arbeiten, die sich mit kognitiver Psychologie befassen, nachgelesen werden.

Die Benutzung der verschiedenen Wissensformen durch einen Experten kann an dem folgenden Beispiel nachvollzogen werden:

Kommt man mit einem Auto in eine Autowerkstatt, so schildert man zunächst die bereits beobachteten Symptome (zum Beispiel, daß der Motor nicht anspringt und die Ladekontrolleuchte brennt). Der Experte zieht aus diesen, und vielleicht weiteren von ihm durch Tests in Erfahrung gebrachten Symptomen Schlüsse, die ihn einer Diagnose näher bringen. Hierbei greift er meistens auf Erfahrungswissen zurück, da er so meist schneller zu Zwischendiagnosen gelangt, als es bei der Verwendung von kausalen Schlußketten der Fall wäre. Trifft er auf Schwierigkeiten, beispielsweise auf unerwartete Symptome, oder wird das Problem sehr komplex, so überlegt er, ob er einen ähnlichen Fall schon einmal behandelt hat. Ist dies der Fall, so versucht er die Diagnose dieses früheren Falles auf den gerade zu lösenden Fall zu übertragen. Ist dies nicht der Fall, und er weiß sich mit intuitiven Schlüssen nicht mehr weiterzuhelfen, so greift er auf sein Tiefenwissen zu und fängt an, systematisch alle Ursachen, die in Betracht kommen, zu untersuchen. Insgesamt

wechselt der Experte häufig zwischen systematischer und intuitiver Vorgehensweise ab, greift mal auf Erfahrungswissen und mal auf Tiefenwissen zu, was die vorher beschriebene Schwierigkeit der Grenzfindung zwischen den Wissensmodellierungsarten beim Experten (wann er auf welches Wissen zugreift) gerade ausmacht.

Insgesamt sollte zu diesem Abschnitt gesagt werden, daß sich die hier vorgestellten Erkenntnisse nicht auf bestimmte Literaturquellen stützen (leider wurde zu diesem Thema keine Literatur gefunden, die einen Überblick über menschliche Wissensformen aus Sicht von Psychologen vermittelt; die meisten Wissenschaftler befassen sich mit einem ganz bestimmten Gebiet, z. B. fallbasiertem Schließen [RS89]), sondern auf Erfahrungen beruhen, die auf mit einem Psychologen durchgeführten Gesprächen basieren, und auf Erfahrungen, die ich selbst bei bereits durchgeführten Wissensakquisitionen sammeln konnte (in den Bereichen Autodiagnose und Drehmaschinen). Informationen über allgemeine kognitive Psychologie können in [And83] eingeholt werden.

In den folgenden Abschnitten und Kapiteln werden häufig die Begriffe *Vollständigkeit* und *Korrektheit* verwendet. Unter Vollständigkeit wird hier vorwiegend Vollständigkeit in Bezug auf die Realität (z. B. ein Modell ist vollständig, wenn alle Komponenten, Verbindungen und Funktionen des realen Gerätes erfaßt wurden, oder eine Diagnosenmenge ist vollständig, wenn alle in der Realität vorkommenden Diagnosen erfaßt wurden) verstanden (andernfalls wird die Bedeutung explizit hinzugefügt). Unter Korrektheit wird ebenfalls Korrektheit in Bezug auf die Realität (eine Diagnose ist korrekt, wenn der richtige Fehler gefunden wurde) verstanden (auch hier wird die Bedeutung andernfalls hinzugefügt).

2.1 Heuristische Wissensmodellierung

Erfahrungswissen läßt sich auf verschiedene Weisen modellieren. Eine der Möglichkeiten, wie Erfahrungswissen vorliegen kann, sind Assoziationen [Pup90][2]. Hier liegt das diagnostische Expertenwissen so vor, daß zwischen Problemmerkmalen (Symptomen) und Problemlösungen (Diagnosen) Beziehungen gebildet werden, die auf der Erfahrung des einzelnen Experten beruhen. Bei diesem, in Form von Assoziationen vorliegenden Wissen handelt es sich meistens um unsicheres Wissen. Die Unsicherheit liegt in der Bewertung der Beziehungen zwischen Problemmerkmalen und Problemlösungen, basiert aber nicht auf statistischen Auswertungen, sondern auf der Intuition und Erfahrung eines oder mehrerer Experten. In Abbildung 2.1 ist dieser Sachverhalt graphisch dargestellt. Wie man nun die Beziehungen zwischen Symptomen und Diagnosen bewertet, ob man also Sicherheitsfaktoren, Wahrscheinlichkeiten, Punktzahlen oder irgendeine Form von qualitativer Bewertung wählt, hängt vom Einsatzgebiet und von dem jeweiligen Experten und dem Wissensingenieur ab. Die Assoziationen liegen meistens in Form von Regeln vor. Die unterschiedlichen Bewertungsmöglichkeiten zwischen Symptomen und Diagnosen sollen an dem folgenden Beispiel verdeutlicht werden:

[2] Die Erkenntnisse dieses Abschnittes stützen sich vorwiegend auf [Pup90].

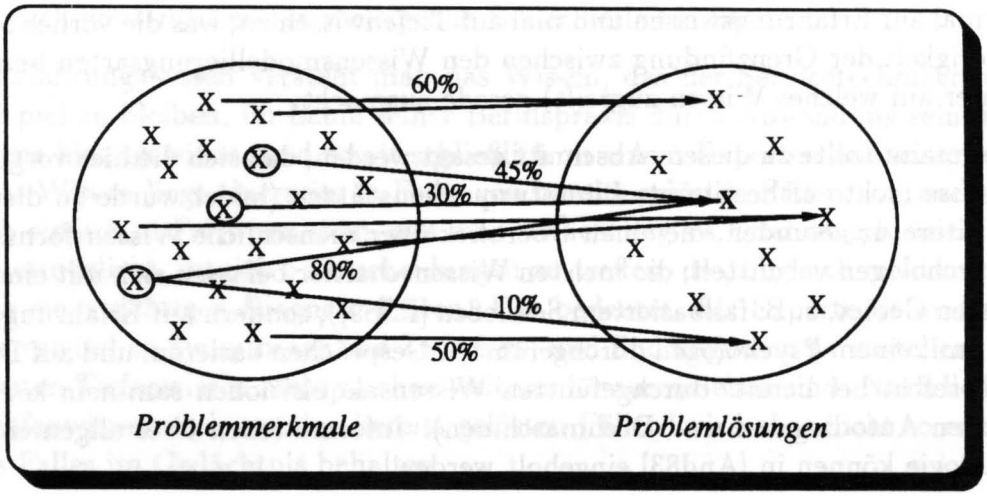

Abbildung 2.1: Grundstruktur heuristischer Modellierung

Beispiel 2.1.1 Eine Regel mit verschiedenen Bewertungsformen:

wenn Kühlwasserverlust

dann

	<Diagnose>	<Angabe in Form von %>	<Angabe in Form von qual. Werten>	<Angabe in Form von Punkten>
	Schläuche undicht	(50%)	(sehr starker Verdacht)	(10 Punkte)
oder	Kühler undicht	(12%)	(mittlerer Verdacht)	(6 Punkte)
oder	Ausgleichsbehälter undicht	(6%)	(leichter Verdacht)	(4 Punkte)
oder	Wasserpumpe undicht	(22%)	(starker Verdacht)	(4 Punkte)
oder	Zylinderkopfdichtung defekt	(8%)	(leichter Verdacht)	(4 Punkte)
oder	Riß im Motorblock	(2%)	(sehr leichter Verdacht)	(2 Punkte)

Das zur Diagnose benötigte Wissen setzt sich aus folgenden Teilelementen zusammen:

- *Symptome*: Merkmale einer Maschine mit ihren zugehörigen Ausprägungen. Die für die Diagnoseerstellung benötigten Merkmale werden von einem Experten vorgegeben
- *Zwischendiagnosen*
- *Enddiagnosen*
- *Assoziationen zwischen Symptomen und Zwischendiagnosen und zwischen Symptomen und Enddiagnosen, die durch Regeln erfaßt werden*
- *Tests*: Verfahren mit denen man Symptomerhebungen durchführen kann

- *Strategiewissen*: Wissen, das nötig ist, die erstellte Regelbasis abarbeiten zu können[3]
- *Verfahren zur Verarbeitung der unsicheren Werte*: Wissen, das für die Verrechnung der unterschiedlichen Unsicherheiten, die den vom Experten angegebenen Assoziationen und Symptomen anheften, benötigt wird[4].

Sowohl Symptome, als auch Diagnosen können zu Symptom–, bzw. Diagnoseklassen zusammengefaßt werden. Zu Symptomklassen werden dabei die Symptome zusammengefaßt, die sich bei einer bestimmten Diagnose, also Fehlerursache, zeigen, und zu Diagnoseklassen werden alle Diagnosen zusammengefaßt, die eine mögliche Erklärung für das Vorliegen eines Symptoms oder einer Reihe von Symptomen (also einer Symptomklasse) liefern.

Die Struktur des heuristischen Diagnosewissens und sein Umfang wird durch die folgenden beiden Vorgehensweisen bei der Wissensakquisition beeinflußt:

- lösungsorientiert
- merkmalsorientiert

Bei der lösungsorientierten Vorgehensweise erfaßt und strukturiert der Experte zunächst die Lösungen, die von dem Diagnose–Expertensystem erkannt werden sollen, listet die zu jeder Lösung gehörenden Merkmale auf, die für oder gegen deren Gültigkeit sprechen (es werden also Regeln der Form Symptom $\longrightarrow$ Enddiagnose erstellt), verfaßt dann einen Fragenkatalog zur Erfassung der benötigten Merkmale, und strukturiert diesen nach Merkmalsgruppen. Die lösungsorientierte Vorgehensweise eignet sich besonders für die schnelle Erstellung eines Prototyps, da man sich ja auf bestimmte Fehler beschränkt, die das System erkennen soll. Nachteilig wirkt sich diese Vorgehensweise auf die Änderbarkeit aus. Da nur Merkmale für bestimmte Lösungen erfragt werden, müssen bei einer Ergänzung des Systems meistens aufwendige Umstrukturierungen, sowohl in den Merkmalsgruppen, als auch in dem Fragenkatalog, erfolgen.
Bei der merkmalsorientierten Vorgehensweise erfaßt und strukturiert man alle Merkmale, die für die Diagnoseerstellung von Bedeutung sind. Voraussetzung für diese Vorgehensweise ist, daß man einen guten Überblick über die Gesamtdomäne hat. Anschließend erfaßt man die Lösungen zu den verschiedensten Merkmalskombinationen, eventuell über Zwischenlösungen laufend. Man erhält somit Regeln der Form

- Symptom $\longrightarrow$ Enddiagnose
- Symptom $\longrightarrow$ Zwischendiagnose
- Symptom & Zwischendiagnose $\longrightarrow$ Enddiagnose

[3] In der Literatur werden hierfür einige Strategien angegeben [Pup88, Bec92], z. B. Vorwärtsverkettung, Rückwärtsverkettung, Hypothesize–and–Test–Strategie und Establish–Refine–Strategie, um die bekanntesten zu nennen.

[4] Auch hierfür können in der Literatur einige Verfahren nachgelesen werden; bekannt sind z. B. Theorem von Bayes, Theorie von Dempster–Shafer, und weitere [Pup88, Ric89].

Außerdem benötigt man auch hier wieder einen Fragenkatalog zur Erfassung der Merkmalsausprägungen. Der Vorteil dieser Vorgehensweise ist, daß man hierbei im Gegensatz zur lösungsorientierten Vorgehensweise besser Erweiterungen an dem System vornehmen kann.

Insgesamt sind heuristische Diagnose–Expertensysteme sehr verbreitet. Ein Grund hierfür liegt in den jeweiligen Anwendungsgebieten selbst. So liegt ein großer Teil medizinischen Wissens in Form von Assoziationen zwischen Symptomen und Krankheiten vor, und besteht zu einem großen Teil aus Erfahrungswissen. Bei technischen Geräten liegt zwar auch viel Wissen in Form von Symptom–Diagnose–Assoziationen vor, aber hier kann man auch alle Vorgänge der Maschine und ihren Aufbau durch Tiefenwissen erfassen, was in der Medizin zur Zeit nicht möglich ist.

Ein weiterer Grund für die Verbreitung heuristischer Diagnose–Expertensysteme ist ihre relativ einfache und wenig aufwendige Erstellung im Vergleich zu Diagnose–Expertensystemen, bei denen auf andere Wissensmodellierungsarten (wie etwa auf modellbasiertes Wissen) zugegriffen wird. Im Prinzip benötigt man lediglich einen Experten, der über möglichst gute Assoziationen zwischen Symptomen und Diagnosen verfügt. Verrechnet man nun die Erfahrungswerte mit einem guten Verfahren, so können mit diesen Systemen schnell gute Resultate erzielt werden.

Man kann mit einem heuristischen Diagnose–Expertensystem Teile der Vorgehensweise eines menschlichen Experten (nämlich die hypothetisch–deduktive Vorgehensweise) mitsamt seinem Wissen sehr gut nachmodellieren. Die anderen Bereiche des menschlichen Expertenwissens werden in den folgenden Abschnitten beschrieben.

2.2 Modellbasierte Wissensmodellierung

Eine andere Möglichkeit der Wissensmodellierung für technische Diagnose–Expertensysteme ist die *modellbasierte Wissensmodellierung*, auch unter dem Begriff *kausale Wissensmodellierung* bekannt. Bei der heuristischen Wissensmodellierung werden nur Assoziationen zwischen Fehlermerkmalen und Fehlerursachen modelliert, die auf der Erfahrung eines Experten beruhen. Konkretes Wissen über das zu diagnostizierende Objekt, hier das technische Gerät, wird nicht direkt verwendet. Dieses objektbezogene Wissen bildet den Kern kausaler Diagnose–Expertensysteme. Das objektbezogene Wissen wird dazu benutzt, ein Modell des zu diagnostizierenden Objektes zu erstellen. Dieses Modell setzt sich aus zwei Wissensarten zusammen (die Erkenntnisse in diesem Abschnitt stammen vorwiegend aus [Pup90, Reh91]):

- *Aufbauwissen*
- *Funktionswissen*

Das Aufbauwissen selbst besteht wiederum aus den Wissensarten

- *Strukturwissen*
- *Konnektivitätswissen*
- *Topologiewissen*

Durch das Strukturwissen werden die einzelnen Komponenten und Unterkomponenten, aus denen sich das Diagnoseobjekt zusammensetzt, erfaßt. Der Aufbau jeder Komponente wird hierbei genau beschrieben. Die Komponentenzerlegung einer Auto–Zündanlage ist in Abbildung 2.2 dargestellt.
Mit dem Konnektivitätswissen werden die Verbindungen zwischen den einzelnen Kom-

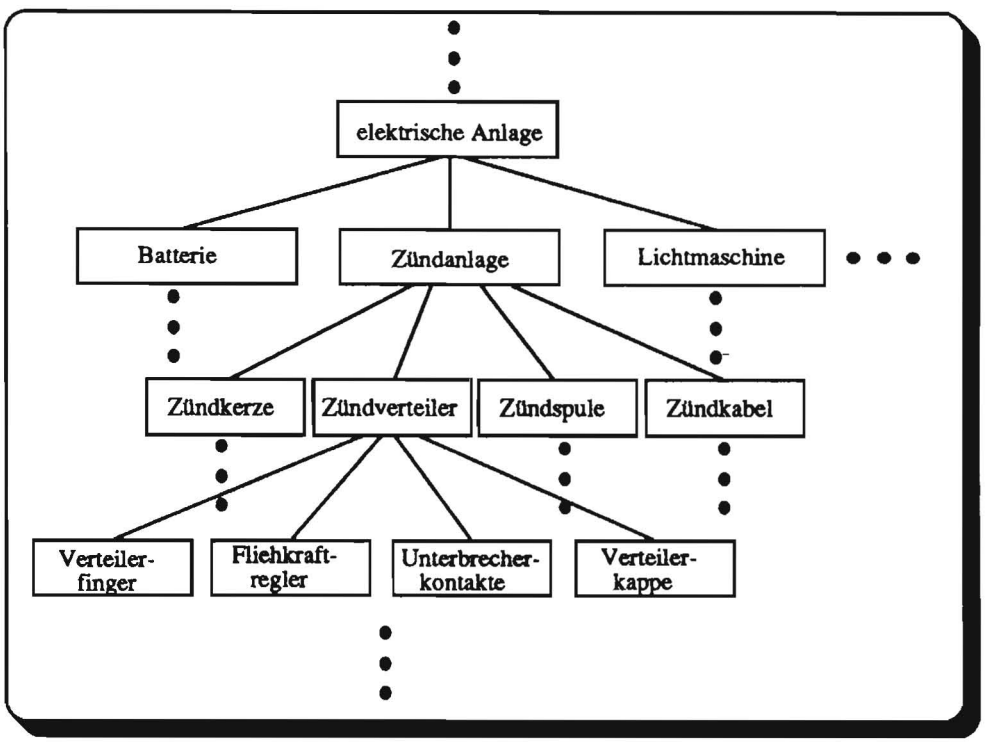

Abbildung 2.2: Komponentenzerlegung einer Zündanlage

ponenten des Strukturwissens beschrieben. In Abbildung 2.3 wird die um das Konnektivitätswissen erweiterte Beschreibung einer Auto–Zündanlage dargestellt.
Die räumliche Anordnung der im Strukturwissen beschriebenen Komponenten wird durch das Topologiewissen erfaßt. Ein Beipiel hierfür ist in Beispiel 2.2.1 nachfolgend zu finden.

Diese Wissensart wird zur Diagnose jedoch meistens nicht verwendet[5]. Der Grund hierfür ist einmal darin zu finden, daß die Erfassung der Topologie und ihre Repräsentation und Verarbeitung sehr schwierig und mit erheblichem Aufwand verbunden ist. Zum

[5]Für die Konstruktion eines Gerätes ist dieses Wissen jedoch von erheblicher Relevanz.

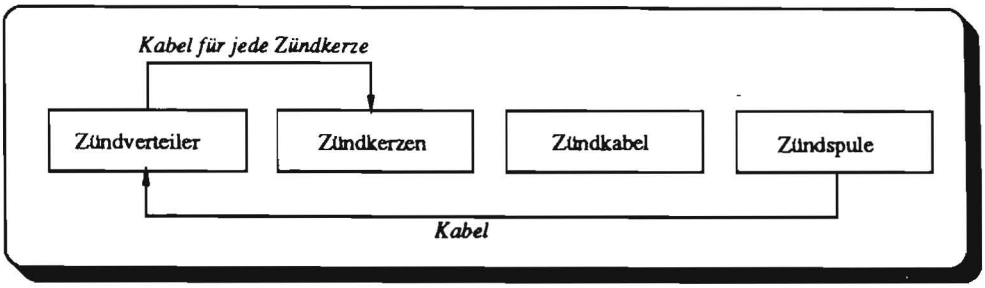

Abbildung 2.3: Konnektivität einiger Komponenten einer Zündanlage

anderen treten mit der Topologie verbundene Fehler in den meisten Anwendungsgebieten nicht auf.

Beispiel 2.2.1 Bei folgendem Fehler ist die räumliche Anordnung der Komponenten von großer Bedeutung:
Liegt die Abgasanlage zu dicht an der Benzinanlage, so kann dies zu Dampfbläschen im Benzin führen, was wiederum zu Ruckeln bis Ausfallen des Motors führen kann.

Prinzipiell kann man das Funktionswissen des zu diagnostizierenden Objektes aus zwei verschiedenen Blickwinkeln heraus betrachten, und damit auch auf unterschiedliche Art und Weise modellieren [Pup88].
Auf der einen Seite hat man die Möglichkeit, das Verhalten des funktionierenden Gerätes zu beschreiben. Diese Art der Wissensmodellierung wird als *funktionale Wissensmodellierung* bezeichnet. Für das KFZ–Beispiel wäre eine funktionale Beschreibung:

wenn die Umdrehungszahl der Antriebswellen steigt

dann steigt die Geschwindigkeit.

Auf der anderen Seite kann man das Verhalten des defekten Gerätes beschreiben. Hier spricht man von *pathophysiologischer Wissensmodellierung.* Auch hierfür soll ein Beispiel gegeben werden:

wenn das Kabel gebrochen ist

dann ist die Stromversorgung unterbrochen.

Will man in medizinischen Anwendungsbereichen kausales Wissen modellieren, so kann man dies meistens nur durch pathophysiologische Wissensmodellierung erreichen, da die Funktionsweise des Menschen nicht hinreichend gut bekannt, zu komplex und von Mensch zu Mensch verschieden ist. Es läßt sich nicht *ein* Modell für den Menschen erstellen, genau genommen müßte man wahrscheinlich jeden Menschen extra modellieren (nicht jeder bekommt bei gleicher Raumtemperatur einen Schnupfen).

Auf technische Geräte trifft dieser Sachverhalt nicht so extrem zu. Vom Menschen geschaffen kann man sie immer durch Funktions- und Konstruktionspläne vollständig beschreiben. Außerdem verhalten sich verschiedene Geräte eines Typs meistens gleich (vorausgesetzt, die allgemeinen Rahmenbedingungen (z. B. Luftfeuchtigkeit) sind die gleichen).

Sowohl funktionalem, als auch pathophysiologischem Wissen können verschiedene Modellierungskonzepte zugrundeliegen [Reh91].

Zunächst soll zwischen *quantitativer* und *qualitativer* Modellierung unterschieden werden.
Technische Verhaltensweisen lassen sich durch die ihnen zugrundeliegenden physikalischen Gesetze mittels Differentialgleichungen beschreiben. Das Verhalten drückt sich damit in Form von exakten Zahlen (quantitativen Werten) aus. Das hierbei auftretende Problem ist der erhebliche Aufwand, den die Berechnungen der Differentialgleichungen verursachen. Dieser Aufwand steigt mit der Genauigkeit der Lösung der Differentialgleichungen stark an, da diese anhand numerischer Verfahren approximiert werden.
Diese Problematik ist ein Grund dafür, warum man häufig dazu übergeht, die quantitativen Werte auf einen diskreten Wertebereich (qualitative Werte) abzubilden. Ein anderer Grund ist, daß man das zu bewältigende Problem durch qualitative Modellierung besser und übersichtlicher lösen kann. Ein Beispiel für den Unterschied zwischen quantitativer und qualitativer Modellierung ist in Beispiel 2.2.2 dargestellt.

Beispiel 2.2.2 Man kann die Kühlertemperatur auf zwei Arten beschreiben:

quantitativ : Kühlertemperatur beträgt 101 Grad Celsius

qualitativ : Kühlertemperatur ist zu hoch

Der Nachteil (in manchen Fällen jedoch auch gewollte Vorteil, weil einfacher zu verarbeiten) dieser Abstrahierung ist der damit zusammenhängende Informationsverlust. Die Arbeit auf der qualitativen Ebene erfordert somit neue Methoden zur Verknüpfung qualitativer Werte. Bei diesen neuen Methoden ist darauf zu achten, daß die qualitative Verhaltensbeschreibung der ursprünglichen quantitativen Verhaltensbeschreibung ähnelt. Der Vorteil der qualitativen Modellierung liegt neben der besseren Verständlichkeit darin, daß die Eingaben für das Diagnose–Expertensystem nicht als exakte Werte vorliegen müssen (der Automechaniker kann z. B. direkt angeben: Kühlertemperatur zu hoch). Außerdem liegt das Wissen auch beim Experten selbst häufig in qualitativer Form vor, und somit ist man seinem Modell dann recht nahe.

Zusammenfassend ist die qualitative Modellierung der quantitativen Modellierung vorzuziehen, wenn es gelingt, quantitative Vorgänge qualitativ zu beschreiben, so daß das intendierte Verhalten erhalten bleibt.

Ein weiteres Modellierungskonzept besteht darin, Wissen *hierarchisch* oder *flach* zu modellieren. Unter hierarchischer Modellierung versteht man die Beschreibung des Diagnoseobjektes auf verschiedenen Ebenen. Jede Ebene stellt ein Modell des Diagnoseobjektes dar. Dabei wird auf einer oberen Ebene von Details der darunterliegenden Ebene

abstrahiert. Unter flacher Modellierung versteht man die Beschreibung des zu diagnostizierenden Gerätes auf nur einer Detaillierungsebene. So ist die in Abbildung 2.2 gezeigte Modellierung hierarchisch. Der Vorteil der hierarchischen Modellierung liegt in ihrer Flexibilität. Je nach Aufgabenstellung kann die Detaillierungsebene verwendet werden, mit der effizient zu einer Lösung zu kommen ist. Die Nachteile der hierarchischen Modellierung liegen in dem erhöhten Speicherplatzbedarf, dem erhöhten Aufwand bei der Erstellung der Wissensbasis und dem erhöhten Verwaltungsaufwand. Die Nachteile der hierarchischen Modellierung sind gleichzeitig die Vorteile der flachen Wissensmodellierung. Der Nachteil der flachen Modellierung ist ihr starrer Charakter gegenüber den Benutzeranforderungen (der Detaillierungsgrad der Diagnose kann vom Benutzer nicht geändert werden).

Der Unterschied der Abstraktionskonzepte quantitative $\longrightarrow$ qualitative Modellierung und tiefere Ebene $\longrightarrow$ höhere Ebene bei der hierarchischen Modellierung besteht darin, daß bei der Abstraktion von quantitativ auf qualitativ auf einer physikalischen Detaillierungsebene von der exakten Funktionsweise auf eine vereinfachte Funktionsweise abstrahiert wird, während bei der hierarchischen Modellierung von Strukturdetails abstrahiert wird.

Wieder eine andere Unterscheidung kann man zwischen *statischer* und *dynamischer* Wissensmodellierung treffen. Prinzipiell kann jeder interessante Vorgang in einem technischen Gerät als dynamisch betrachtet werden. Dennoch wird der Begriff dynamische Modellierung nach [Reh91] strenger definiert.

Ein Verhalten wird dynamisch *genannt, wenn zu seiner Modellierung eine explizite Betrachtung der Zeit notwendig ist, oder einfacher, wenn es nicht statisch ist. Dies ist zum Beispiel dann der Fall, wenn sich im Verlauf des modellierten Vorganges die Werte interessierender Meßpunkte mehr als einmal ändern.*

Ein typisches Beispiel für dynamisches Verhalten sind Systeme mit Rückkopplungen, wie in Abbildung 2.4 zu sehen ist.

Das statische Verhalten eines technischen Geräts ist in [Reh91] wie folgt definiert:

Ist zur Modellierung eines Verhaltens keine explizite Betrachtung der Zeit notwendig, so bezeichnet man es als statisch. *Für derartiges Verhalten läßt sich der Ablauf der Zeit implizit über die Reihenfolge der Ursache–Wirkung–Schritte darstellen. Anders formuliert ist statisches Verhalten dadurch gekennzeichnet, daß für gegebene Eingaben eine Situation entsteht, die Bestand hat und sich nicht mehr ändert, so daß eine Momentaufnahme ausreicht, um die Auswirkungen der Eingaben zu beschreiben.*

Das folgende Verhalten ist zum Beispiel statisch: „Einschalten des Lichts bewirkt Lampe brennt". Durch die notwendige Repräsentation des Zeitaspektes ergibt sich bei der dynamischen Modellierung gegenüber der statischen Modellierung ein erheblicher Mehraufwand. Deshalb sollte die statische Modellierung der dynamischen Modellierung soweit als möglich vorgezogen werden. Kann das dynamische Verhalten des Diagnoseobjektes nicht völlig statisch modelliert werden, so sollte versucht werden, die dynamischen Aspekte soweit einzukapseln, daß sie nach außen hin statisch wirken. Sollte auch dies nicht möglich sein, so muß der Mehraufwand durch die dynamische Modellierung in Kauf genommen

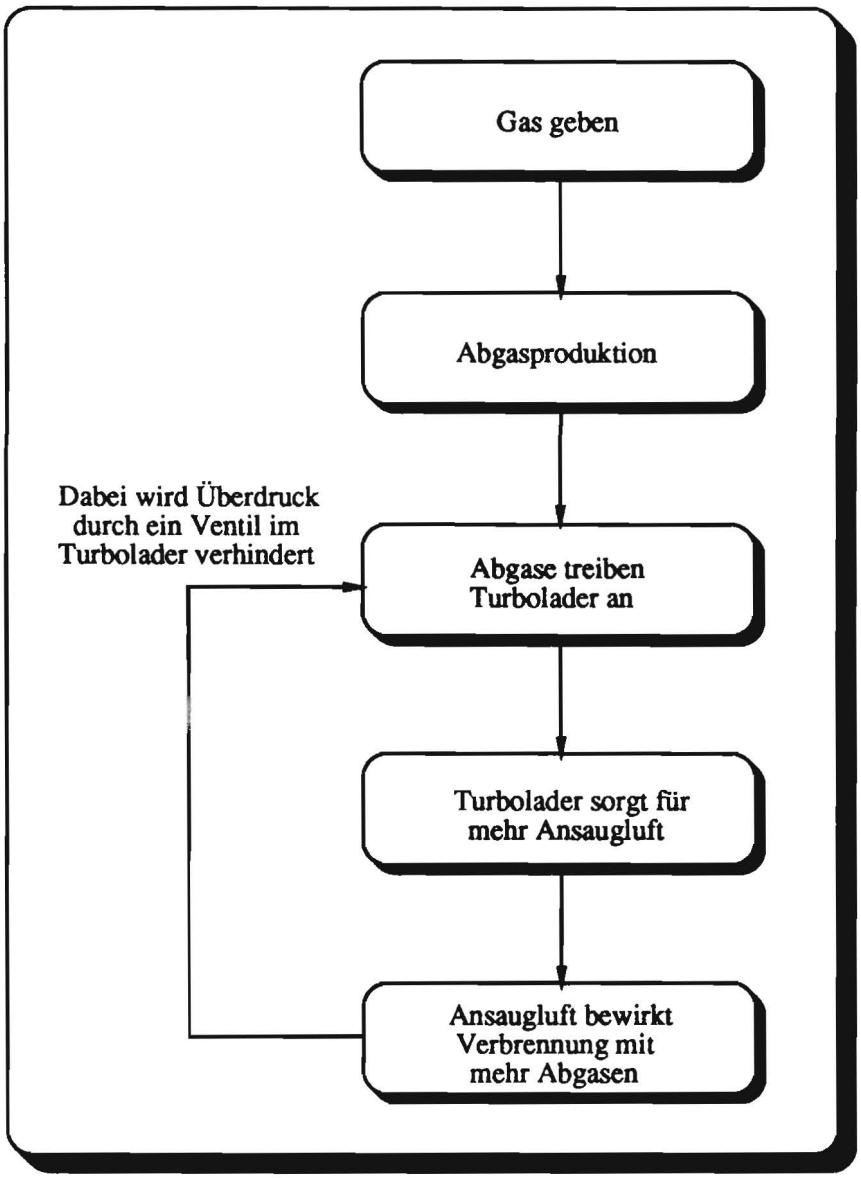

Abbildung 2.4: Der Abgasturbolader mit seinem dynamischen Verhalten

werden.

Eine letzte Unterscheidung bei der Wissensmodellierung ist durch die Sichtweise auf das Diagnoseobjekt gegeben. Hier kann zwischen *komponentenorientierter* und *prozeßorientierter* Modellierung unterschieden werden.
Bei der prozeßorientierten Modellierung stehen die in dem zu diagnostizierenden Objekt ablaufenden technischen Prozesse im Vordergrund. Diese Prozesse kommunizieren über ihre Parameter miteinander. Die physikalischen Komponenten interessieren nur dann, wenn sie an den technischen Prozessen beteiligt sind.
Bei der komponentenorientierten Modellierung hingegen werden die einzelnen Bauteile des Diagnoseobjektes beschrieben. Unter dieser Beschreibung versteht man ihren Aufbau und ihre Funktionsweise. Die Komponenten kommunizieren über Schnittstellen, die *Ports* genannt werden.
Bei der prozeßorientierten Modellierung wird das Gesamtverhalten des technischen Gerätes explizit beschrieben, während dieses Wissen bei der komponentenorientierten Modellierung implizit enthalten ist und umgekehrt.
Da das Ziel der Diagnostik meistens die Feststellung eines fehlerhaften Bauteils ist, eignet sich der komponentenorientierte Ansatz eher zur Modellierung des Diagnosewissens, als der prozeßorientierte Ansatz.

Zusammenfassend seien die verschiedenen Modellierungkonzepte hier noch einmal gegenübergestellt, wobei angemerkt werden soll, daß sich ihre Verwendung nicht gegenseitig ausschließt (man kann ein Diagnoseobjekt beispielsweise gleichzeitig qualitativ, hierarchisch, statisch und komponentenorientiert modellieren):

quantitative Modellierung — qualitative Modellierung

hierarchische Modellierung — flache Modellierung

dynamische Modellierung — statische Modellierung

prozeßorientierte Modellierung — komponentenorientierte Modellierung

Um das Wissen einer Domäne überhaupt kausal modellieren zu können, sollten folgende Voraussetzungen erfüllt sein:

- Das Wissen muß für die Problemstellung vollständig zu erfassen sein, d. h. in dem zu diagnostizierenden Gebiet, beispielsweise einem technischen Gerät, müssen sowohl der Aufbau, als auch die Funktionsweise auf jedem benötigten Abstraktionslevel und in der gewünschten Tiefe vollständig bekannt sein.

- Das Wissen sollte effizient zur Simulation nutzbar sein, d. h. es muß bei komplexen Modellen eine Möglichkeit geben, auf höheren Abstraktionsebenen zu diagnostizieren. Oder es muß die Möglichkeit bestehen, mittels einer Fokussierungskomponente die Diagnose auf verdächtige Teilbereiche einzuschränken.

Um die Korrektheit einer Diagnose zu gewährleisten, ist meistens auch noch zusätzliches, d. h. über die eigentliche Modellbeschreibung hinausgehendes, kausales common–sense–Wissen notwendig. Das bedeutet insbesondere, daß auch manche Fehlerursachen miterfaßt werden sollten, die beispielsweise auf Umwelteinflüssen oder ähnlichem beruhen (z. B. Fehlerursache: Sand im Getriebe).

2.3 Fallbasierte Wissensmodellierung

Eine weitere Form der Modellierung diagnostischen Wissens ist die *fallbasierte Wissensmodellierung*. Hier liegt das Wissen einer Domäne in Form von konkreten, bereits gelösten *Diagnosefällen* vor. Anhand dieser Diagnosefälle wird eine möglichst umfangreiche und damit auch repräsentative Falldatenbasis aufgebaut.

Ein Diagnosefall setzt sich aus einer Menge für die Diagnose relevanter Merkmale und einer zugehörigen Diagnose zusammen[6] [Pup90]. Ein kleiner Ausschnitt aus einer Falldatenbasis ist in Abbildung 2.5 gezeigt.

	neuer Fall	Vergleichsfall-1	Vergleichsfall-2
Autotyp	Audi-80	Mercedes 190	Audi-80
km-Stand	124.000	95.000	66.000
Motor ruckelt	ja	ja	nein
Auto springt nicht an	meistens	manchmal	immer
Lösung	?	Zündkerzen verbraucht	Batterie leer

Abbildung 2.5: Ein kleiner Ausschnitt aus einer Falldatenbasis zur Diagnose an Kraftfahrzeugen

[6] Welche Merkmale dabei relevant sind, wird von einem Experten bestimmt. In dieser Festlegung liegt auch ein guter Teil heuristisches Wissen.

Man kann verschiedene Arten von Merkmalen unterscheiden:

- *Ja-Nein-Merkmale*: Merkmale, deren Ausprägungen sich auf die Werte „vorhanden" oder „nicht vorhanden" beschränken, z. B. „Ladekontrolleuchte brennt" oder „Ladekontrolleuchte brennt nicht".
- *numerische Merkmale*: Merkmale, deren Ausprägungen in Form von konkreten Zahlen vorliegen, z. B. „Kilometerstand = 35.312 km" (es handelt sich im Prinzip um quantitative Werte).
- *symbolische Merkmale*: Merkmale, deren Ausprägungen zunächst nicht in Form von konkreten Zahlen vorliegen, z. B. (immer, häufig, manchmal, selten, nie). Symbolische Wertebereiche lassen sich jedoch häufig auf numerische Wertebereiche abbilden, z. B. immer=1, häufig=2, manchmal=3, selten=4, nie=5 (hier kann man von qualitativen Werten sprechen, die wieder auf qualitative Werte abgebildet werden).

Die grundlegende Vorgehensweise bei der fallbasierten Diagnose liegt darin, einen zum gerade zu diagnostizierenden Fall ähnlichsten Fall aus der Falldatenbasis herauszusuchen. Um den Grad der Übereinstimmung zweier Fälle bewerten zu können, ist ein *Ähnlichkeitsmaß* erforderlich [Weß90]. Dieses Maß ist eine Vorschrift, mit der die Ähnlichkeiten der einzelnen Merkmale zweier Fälle zu einer Gesamtähnlichkeit der beiden Fälle zusammengezogen werden. Zur Unterstützung des Ähnlichkeitsmaßes werden die Fälle häufig in Klassen geordnet[7].
Für die einzelnen Merkmalstypen muß es eigene Vorschriften zur Feststellung ihrer Ähnlichkeiten geben (die Ähnlichkeit zweier Merkmalsausprägungen wird in einem Wertebereich von [0, 1] angegeben) [Weß90].

- *Ja-Nein-Merkmale*: Stimmen die Ausprägungen überein, so ist die Ähnlichkeit gleich 1, tritt eine Diskrepanz auf, so ist die Ähnlichkeit gleich 0.
- *numerische Merkmale*: Zur Feststellung der Ähnlichkeit zweier Merkmalsausprägungen wird meistens eine mathematische Formel verwendet, z. B.

  $$\ddot{A}hnlichkeit = kleinererWert/größererWert$$

  beim Benzinverbrauch. Die Funktion zur Formel muß den Wertebereich [0, 1] haben.
- *symbolische Merkmale*: Hier müssen Ähnlichkeitswerte zwischen Ausprägungspaaren direkt angegeben werden[8], z. B.

  Ähnlichkeit(immer, nie) = 0;

  Ähnlichkeit(immer, selten) = 0.25;

  Ähnlichkeit(immer, manchmal) = 0.5;

[7] Hierfür gibt es spezielle Klassifikationsmechanismen [Weß90, Bar88, BPW88].

[8] Es sei denn, man hat die symbolischen Merkmalswerte auf einen numerischen Wertebereich abgebildet. In diesem Fall kann man wie bei numerischen Merkmalen vorgehen.

Ähnlichkeit(immer, häufig) = 0.75;

Ähnlichkeit(immer, immer) = 1;

etc.

Für die einzelnen Merkmale müssen Gewichtungen angegeben werden, die ihre Relevanz innerhalb der Domäne oder eines gerade zu diagnostizierenden Falles widerspiegeln. So ist die Farbe eines Autos sicherlich völlig uninteressant für die Motordiagnose, während der Autotyp durchaus wichtig sein kann[9]. In der Gewichtung und der Form des Ähnlichkeitsmaßes steckt entweder heuristisches oder statistisches Wissen, je nachdem, ob sie auf der Erfahrung eines Experten beruhen, oder durch ein statistisches System erstellt werden[10].

Prinzipiell kann man den herausgesuchten ähnlichsten Fall zu einem der folgenden Zwecke nutzen [CMM86]:

- Übertragung des Lösungsweges (generativer Ansatz)
  - direkte Übernahme des Lösungsweges
  - modifizierte, angepaßte Übernahme des Lösungsweges
- Übertragung der Lösung (Variantenansatz)
  - direkte Übernahme der Lösung
  - modifizierte, angepaßte Übernahme der Lösung

Voraussetzung für die Anwendung der fallbasierten Diagnose in einer Domäne ist, daß man eine möglichst umfangreiche Falldatenbasis zur Verfügung haben sollte. Dies ist jedoch keine Forderung, die sich auf die Korrektheit einer gestellten Diagnose bezieht. Sie ist eher eine Empfehlung, da man mit mehr verschiedenen Fällen auch mehr Fehler finden kann und bestimmte Fehler auch sicherer finden kann.

Die Idee der fallbasierten Diagnose ist noch sehr jung, so daß bisher wenige Erfahrungen mit derartigen Systemen gemacht worden sind.

2.4 Statistische Wissensmodellierung

Die vierte Möglichkeit Diagnosewissen zu modellieren besteht darin, Statistiken zu verwenden [Pup90, DR83]. Im Prinzip handelt es sich bei der *statistischen Wissensmodellierung* um einen Spezialfall der heuristischen Wissensmodellierung. Der Unterschied liegt in der Herkunft und Verrechnung der Unsicherheiten. Während bei heuristischen Diagnose–Expertensystemen auftretende Unsicherheiten von einem Experten subjektiv

[9] Bei einem System zur Feststellung von Unfallursachen wäre auch die Farbe eines Autos von Bedeutung (z. B. dunkle Autos sind nachts häufiger in Unfälle verwickelt als helle Autos).

[10] Ein Stichwort in diesem Zusammenhang ist das fallbasierte Lernen [RS89, PS89, Alt91].

geschätzt werden, werden sie bei statistischen Diagnose–Expertensystemen durch Auswertung von Fällen objektiv errechnet. Die Verrechnung der Unsicherheiten geschieht bei heuristischen Systemen durch die Verwendung eines intuitiven Verrechnungsschemas. Ein solches Schema ist in der Regel im mathematischen Sinne nicht korrekt, liefert jedoch meistens zufriedenstellende Resultate. Statistische Diagnose–Expertensysteme verwenden stattdessen eine *mathematische Evidenztheorie.* Diese erzeugt mathematisch korrekte Ergebnisse innerhalb des verwendeten Modells. Der Unterschied zwischen statistischer und heuristischer Wissensmodellierung liegt also darin, daß bei der statistischen Wissensmodellierung ein zugehöriges Wahrscheinlichkeitsmodell vorliegt, mit dem mathematisch korrekte Resultate erzielt werden (d. h. wenn ein statistisches System eine Diagnose mit der Wahrscheinlichkeit 0.8 liefert, so ist diese Bewertung absolut korrekt (korrekt bezüglich des Modells)), während bei der heuristischen Wissensmodellierung kein Modell zugrundegelegt wird, wodurch auch die erzielten Ergebnisse nicht richtig sein müssen (liefert ein heuristisches System eine Diagnose mit der Wahrscheinlichkeit 0.8, so muß dies kein im mathematischen Sinne korrekter Wert sein).

Um Wissen statistisch modellieren zu können, müssen sehr strenge Voraussetzungen erfüllt sein. Die meisten statistischen Systeme verwenden in irgendeiner Form das *Theorem von Bayes* [Pup88, Ric89, CM85], so daß sich Teile der folgenden Voraussetzungen auf diese Evidenztheorie beziehen:

- Die einzelnen Symptome müssen voneinander unabhängig sein, d. h. aus dem Vorkommen des einen Symptoms darf man nicht auf das Vorkommen eines anderen Symptoms schließen können.
- Die Diagnosemenge muß vollständig erfaßt sein, d. h. für jede mögliche Diagnose müssen in der Falldatenbasis Fälle existieren.
- Die einzelnen Diagnosen müssen sich gegenseitig ausschließen (Stichwort „single–fault–assumption“).
- Die Falldatenbasis muß repräsentativ sein.
- Jede Diagnosenalternative muß durch ausreichend viele Fälle in der Falldatenbasis vertreten sein.
- Man benötigt folgende Wahrscheinlichkeiten:
  - a̅ priori Wahrscheinlichkeiten der Diagnosen;
  - bedingte Wahrscheinlichkeiten, die angeben, wie häufig ein bestimmtes Symptom bei Vorliegen einer bestimmten Diagnose vorkommt.

Bei Verletzung dieser Voraussetzungen wird aus dem statistischen Wissen heuristisches Wissen, weil das dahinterstehende Modell verloren geht.

Prinzipiell kann man die Falldatenbasis zur statistischen Auswertung verwenden, die auch schon bei der fallbasierten Diagnose benutzt wird, d. h. auch hier liegen die Fälle in Form von Symptomen mit zugehörigen Diagnosen vor.

2.5 Wissen in Form von Entscheidungsbäumen und Entscheidungstabellen

Die fünfte Möglichkeit, Diagnosewissen zu modellieren, ist, es in Form von Entscheidungsbäumen oder Entscheidungstabellen zu modellieren [Pup90].

2.5.1 Entscheidungsbäume

Ein Entscheidungsbaum ist ein Baum, bei dem die inneren Knoten Fragen darstellen, die Kanten mögliche Antworten zu den in den inneren Knoten enthaltenen Fragen und die Blätter die Lösung des zu bearbeitenden Problemgebietes. Ein innerer Knoten kann jedoch auch selbst wieder einen Entscheidungsbaum beinhalten [Pup90, Sch91].
In Abbildung 2.6 ist ein allgemeines Beispiel für einen Entscheidungsbaum gezeigt [Pup90]. Durch einen Entscheidungsbaum ist die Abarbeitungsreihenfolge der Fragen, durch deren

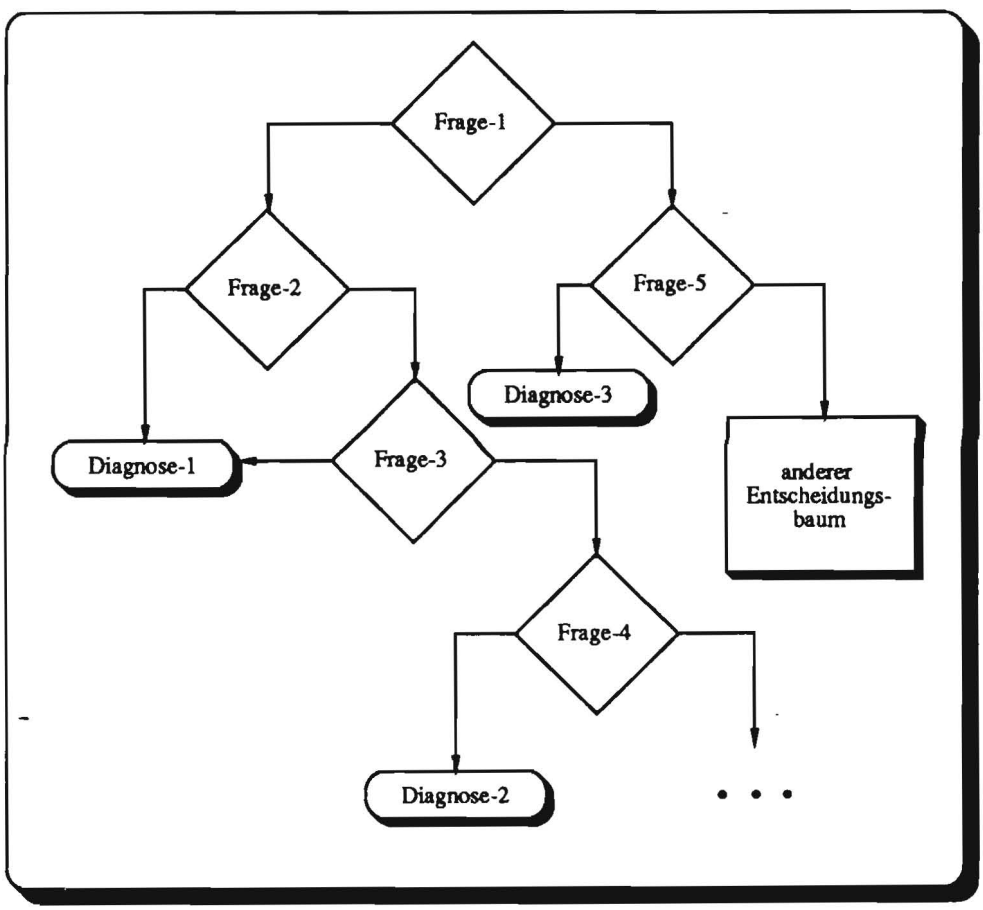

Abbildung 2.6: Ein allgemeiner Entscheidungsbaum

Beantwortung das Problem gelöst werden soll, von vornherein festgelegt (deterministisch),

d. h. bei dem so modellierten Wissen handelt es sich um Domänewissen, das untrennbar mit dem dazugehörigen Abarbeitungswissen verknüpft ist. Da die Trennung von Domänewissen und Abarbeitungswissen jedoch ein wichtiges Merkmal für Expertensysteme ist, wird häufig die Frage aufgeworfen, ob man für ein Expertensystem, das lediglich mit in dieser Form modelliertem Wissen arbeitet, überhaupt den Begriff Expertensystem verwenden soll, oder ob es sich dann nicht nur um ein „gewöhnliches" Programm handelt. Diese Frage wird in Kapitel 4, im Anschluß an die Vorstellung diverser Diagnose–Expertensysteme, näher beleuchtet.
Die vorgegebene Reihenfolge selbst entsteht zum einen aus Erfahrungswerten, d. h. man läßt die Tests in einer Reihenfolge durchführen, die eine Ursachenwahrscheinlichkeit repräsentiert (das Teil, das erfahrungsgemäß am häufigsten kaputt geht, wird als erstes getestet). Zum anderen sind es aber auch die durchzuführenden Tests selbst, die eine gewisse Reihenfolge sinnvoll erscheinen lassen, d. h. hier geht man nach dem Prinzip „einfache Tests zuerst" vor. Findet man hier eine gesunde Mischung, so ist auch die Akzeptanz des Benutzers, der die ganzen Tests durchführen soll, gewährleistet. In Abbildung 2.7 ist ein Entscheidungsbaum aus dem Bereich Kraftfahrzeug gezeigt.

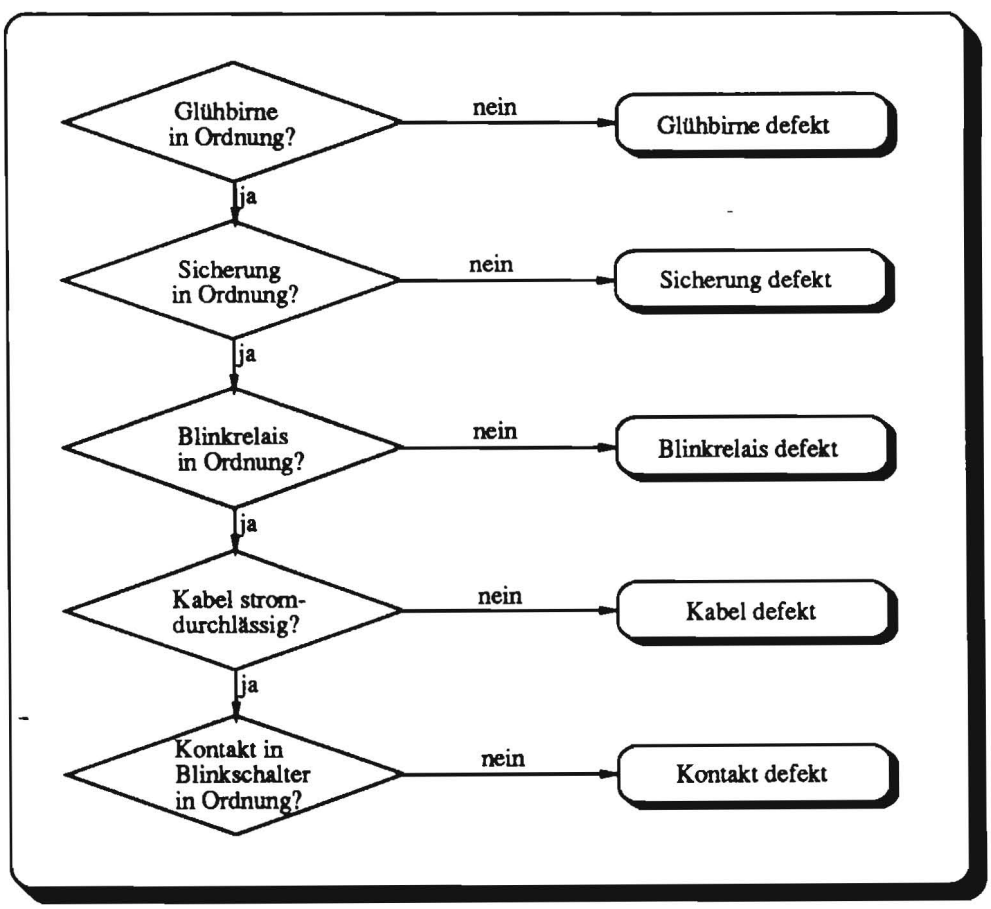

Abbildung 2.7: Ein Entscheidungsbaum bei Auftreten des Fehlers „Blinker funktioniert nicht"

Die Vorteile eines Entscheidungsbaums liegen darin, daß er kleine Probleme sehr übersichtlich und verständlich erfaßt. Außerdem kann man durch ihn in gewissen Problembereichen in sehr schneller Zeit zu einer Problemlösung kommen. Für große Probleme kann er jedoch das genaue Gegenteil bedeuten, d. h. sie erweisen sich in dieser Form als sehr unübersichtlich. Ein weiterer Kritikpunkt an Entscheidungsbäumen ist, daß sie sehr änderungsfeindlich sind. Nimmt man mitten in dem Entscheidungsbaum eine lokale Änderung vor, so kann sich dies durchaus global auswirken, d. h. es kann passieren, daß man wegen dieser lokalen Änderung den ganzen Entscheidungsbaum abändern muß [Pup90].

2.5.2 Entscheidungstabellen

Bei Entscheidungstabellen handelt es sich im Prinzip um entsequentialisierte Entscheidungsbäume [Pup90, GSJM87]. Die Abarbeitungsreihenfolge ist nicht mehr von vornherein festgelegt, wodurch auch eine Dialogführung mit dem Benutzer nicht mehr fest vorgegeben ist. Der allgemeine Entscheidungsbaum aus Abbildung 2.6 ist in Abbildung 2.8 als Entscheidungstabelle graphisch wiedergegeben. Bei einer Entscheidungstabelle

Entscheidungs-tabelle-1	Regel-1	Regel-2	Regel-3	Regel-4	Regel-5
Frage-1	X	X	X	-	-
Frage-2	X	-	-		
Frage-3		X	-		
Frage-4			X		
Frage-5				X	-
Diagnose-1	X	X			
Diagnose-2			X		
Diagnose-3				X	
Entschei-dungstab.-2					X

X = ja
- = nein
leer = nicht relevant

Dabei ist diese Tabelle folgendermaßen zu lesen:

Regel-1: Antwort(Frage-1)=ja und Antwort(Frage-2)=ja ---> Diagnose-1
Regel-2: Antwort(Frage-1)=ja und Antwort(Frage-2)=nein und Antwort(Frage-3)=ja ---> Diagnose-1
...
Regel-5: Antwort(Frage-1)=nein und Antwort(Frage-5)=nein ---> Aufruf(Entscheidungstab.-2)
...

Abbildung 2.8: Eine Entscheidungstabelle zu dem obigen allgemeinen Entscheidungsbaum

sind die Regeln voneinander unabhängig. Ihre Wissensrepräsentation und Wissensmanipulation entsprechen denen eines kommutativen Regelinterpreters, wobei die Regeln Implikationen darstellen. Der Entscheidungsbaum aus Abbildung 2.7 ist in Abbildung 2.9 als Entscheidungstabelle abgebildet.

Der Vorteil bei Entscheidungstabellen gegenüber Entscheidungsbäumen, ist, daß sie

	Regel-1	Regel-2	Regel-3	Regel-4	Regel-5
Glühbirne okay?	-	X	X	X	X
Sicherung okay?		-	X	X	X
Blinkrelais okay?			-	X	X
Kabel okay?				-	X
Kontakt okay?					-
Glühbirne defekt	X				
Sicherung defekt		X			
Blinkrelais defekt			X		
Kabel defekt				X	
Kontakt defekt					X

Abbildung 2.9: Eine Entscheidungstabelle zu dem obigen Beispiel aus dem Bereich Kraftfahrzeug

leichter abzuändern sind. Aber sie sind relativ unübersichtlich [Pup90].

Insgesamt gehören Entscheidungsbäume und Entscheidungstabellen zu den Standard–Software–Engineering–Spezifikationstechniken [Pup90] und werden u. a. in Expertensystem–Werkzeugen (beispielweise im VAX–Decision Expert) angeboten.

2.6 Abgrenzungen

Bei der Aufteilung und Definition der fünf Wissensmodellierungsarten traten verschiedentlich Diskussionen darüber auf, ob nicht die ein oder andere Modellierungsart eigentlich einer anderen unterzuordnen ist. Im einzelnen traten diese Diskussionen an folgenden Stellen auf:

- heuristisch — Entscheidungsbäume
- heuristisch — Entscheidungstabellen

Der Unterschied zwischen heuristischer Wissensmodellierung und Entscheidungsbäumen liegt darin, daß bei Entscheidungsbäumen die Abarbeitungsreihenfolge deterministisch vorgegeben ist, d. h. die Reihenfolge, in der Symptome erhoben werden ist von vornherein fest vorgegeben. Bei der heuristischen Wissensmodellierung ist die Abarbeitungsreihenfolge der Regeln nicht vorgegeben und richtet sich nach den Eingaben des Benutzers und nach dem verwendeten Verrechnungsschema.

Daraufhin ergab sich die Frage, was eine Entscheidungstabelle, als entsequentialisierter Baum, noch von heuristischem Wissen trennt. Die Antwort darauf lautet, daß die Regeln in einer Entscheidungstabelle immer einen Pfad in dem dazugehörigen Entscheidungsbaum darstellen. Dies bedeutet insbesondere, daß alle Symptome, die auf einem solchen Pfad liegen, erhoben werden müssen, um eine Diagnose stellen zu können. Bei der heuristischen Modellierung müssen viele Symptome gar nicht erst erhoben werden, um eine Diagnose stellen zu können. Außerdem benötigt man bei Entscheidungstabellen kein Verrechnungsschema, da explizit vorgegeben ist, welche Diagnose bei welchen Symptomkombinationen zu stellen ist.

Kapitel 3

Verschiedene existierende Diagnose–Expertensysteme

In diesem Kapitel sollen nun einige Diagnose–Expertensysteme (oder Diagnoseshells[1]) vorgestellt werden, die bereits modelliert und/oder realisiert wurden und teilweise auch schon in der Industrie oder der Forschung im Einsatz sind.

Zunächst werden heuristische Diagnose–Expertensysteme vorgestellt. Hierbei handelt es sich um MED2, MegaFilEx und ein Expertensystem zur Diagnose an CNC–Maschinen. Anschließend werden drei modellbasierte Diagnose–Expertensysteme beschrieben, nämlich das System von Davis, QUASIMODIS und ein Expertensystem zur wissensbasierten Diagnose für flexible Fertigungssysteme. Im dritten Abschnitt dieses Kapitels wird das fallbasierte System PATDEX beschrieben. Im darauffolgenden Teil wird das statistische Expertensystem PROSPECTOR vorgestellt, und im letzten Abschnitt dieses Kapitels wird auf das Diagnose–Expertensystem DIWA eingegangen, das mit Wissen in Form von Entscheidungsbäumen arbeitet.

Die ungleiche Verteilung der beschriebenen Diagnose–Expertensysteme auf die Bereiche heuristische, modellbasierte, fallbasierte, statistische und auf Entscheidungsbäumen oder Entscheidungstabellen basierende Wissensmodellierung ergibt sich vorwiegend daraus, daß auf einigen Gebieten sehr viele Systeme existieren, auf anderen Gebieten jedoch umso weniger. So existieren beispielsweise relativ viele heuristische und modellbasierte Expertensysteme, da dies auch die ersten Ansätze waren [Pup90]. Auf einem neueren Gebiet, wie den fallbasierten Expertensystemen, gibt es noch relativ wenig (wie sich bei der Literaturrecherche gezeigt hat). Auch statistische Diagnose–Expertensysteme liegen nicht häufig vor, wahrscheinlich, da hierfür umfangreiche Falldatenbasen zur Auswertung erforderlich sind, um repräsentative Ergebnisse zu erhalten [Pup90]. Ebenso selten scheinen Diagnose–Expertensysteme zu sein, die Wissen in Form von Entscheidungsbäumen oder Entscheidungstabellen zur Diagnoseerstellung heranziehen. Dies rührt wahrscheinlich daher, daß hier die Ähnlichkeit zu „normalen“ Programmen zu groß ist (die Trennung

[1] Shells (Hüllen) sind Expertensysteme ohne ausgefüllte Wissensbasis. Sie beinhalten: Inferenzkomponente, Erklärungskomponente, Benutzeroberfläche, Wissensakquisitionskomponente und fest vorgegebene Formalismen zur Wissensrepräsentation [Sch89].

zwischen Wissensbasis und Verarbeitung des Wissens ist nicht unbedingt so deutlich, wie für Expertensysteme typisch), und man somit vielleicht nicht immer gleich von einem Expertensystem spricht. Es wäre deshalb durchaus denkbar, daß zwar noch einige Expertensysteme dieser Art vorliegen, jedoch nicht als solche bezeichnet werden.

In der folgenden Tabelle wird ein Überblick über die hier vorgestellten Systeme gegeben:

System	*Wissensmodellierung*	*Jahr*
PROSPECTOR	statistisch	1979
MODIS	heuristisch	1983
Davis	modellbasiert	1983
MED2	heuristisch	1986
CNC-Maschine	heuristisch	1986
DAX	heuristisch	1988
System mit KEE	modellbasiert	1988
MegaFilEx	heuristisch	1988
PATDEX	fallbasiert	1990
QUASIMODIS	modellbasiert	1991
DIWA	Entscheidungsbaum	1991

Tabelle 3.1: Überblick über die vorgestellten Systeme

Die Beschreibungen der einzelnen Systeme sehen so aus, daß zunächst die Gesamtarchitektur jedes Systems vorgestellt wird. Da das Hauptaugenmerk dieser Arbeit auf der Wissensmodellierung liegt, wird auch hauptsächlich auf die Wissensbasen der einzelnen Systeme eingegangen, d. h. die anderen Komponenten, wie die Dialogschnittstellen und die Problemlösungskomponenten, werden nur kurz umrissen. Teilweise fallen die Beschreibungen zu diesen Komponenten so aus, daß nur auf besonders interessante Aspekte und Realisierungen hingewiesen wird, die sich auf die Struktur der Wissensbasen auswirken.

Bei der Beschreibung einer Wissensbasis wird zum einen darauf eingegangen, welches Wissen modelliert wird und wie die Modellierung dieses Wissens aussieht, zum anderen jedoch auch darauf, in welche Bereiche die Wissensbasis aufgeteilt ist und zu welchem Zweck eine solche Unterteilung stattfindet.

3.1 Heuristische Diagnose–Expertensysteme

In diesem Abschnitt werden zunächst die heuristischen Diagnose–Expertensysteme MED2, MegaFilEx und ein Expertensystem zur Diagnose an CNC-Maschinen vorgestellt.

Die meisten heuristischen Diagnose–Expertensysteme gehören zur ersten Generation der Diagnose–Expertensysteme. Im Prinzip konnte man hier mit relativ einfachen Mitteln erste Erfolge erzielen. Im Laufe der Zeit kristallisierten sich jedoch auch andere Möglichkeiten der Wissensmodellierung heraus, die sich zum einen daraus ergaben, daß es einfach nicht genügte, das Wissen heuristisch zu modellieren (die Qualität der Lösungen war teilweise durchaus verbesserungsbedürftig, aber mit heuristischer Modellierung nicht mehr machbar), zum anderen aber auch aus der interdisziplinären Arbeit mit Psychologen, durch die man eine genauere Modellierung des menschlichen Experten erzielen wollte.

3.1.1 MED2

Eine der bekannten Expertensystemshells, bei der heuristisch modelliertes Wissen zur Diagnoseerstellung herangezogen wird, ist MED2 (Meta–Ebenen–Diagnosesystem 2) [Pup88, Pup87, Pup90, Pup86]. Diese Expertensystemshell wurde in unterschiedlichen Gebieten eingesetzt. Unter anderem ist mit ihrer Hilfe ein medizinisches Diagnostiksystem erstellt worden, das jedoch keinen eigenen Namen erhielt [Pup86]. Außerdem wurde diese Shell auch für die Erstellung des Systems DAX, das unter anderem von Thomas Legleitner entwickelt wurde, verwendet, mit Hilfe dessen die technische Diagnose an Getrieben vorgenommen wird [LME88]. Ein weiteres heuristisches Diagnosesystem, daß noch mit MED1 erstellt wurde, ist MODIS, ein Expertensystem zur Erstellung von Diagnosen für den Ottomotor und seine Aggregate, das von Horst Peter Borrmann entwickelt wurde [Bor83].

In Abbildung 3.1 ist der Aufbau der Expertensystem–Shell graphisch dargestellt. Die Shell setzt sich aus drei Hauptteilen zusammen, die in sich wieder aufgeteilt sind:

- Wissensbank
- Benutzerschnittstellen
- Problemlösungskomponente

Der Bereich Benutzerschnittstellen soll hier nur kurz umrissen werden, der Bereich Problemlösungskomponente wird in [Bec92] genau behandelt.

Wird ein Expertensystem mit Hilfe der Expertensystemshell MED2 entwickelt, so wird das Wissen des Bereichs, den es zu diagnostizieren gilt, heuristisch modelliert.

Dies bietet sich für das Einsatzgebiet der Medizin besonders an, da die prinzipielle Vorgehensweise eines Arztes eigentlich immer hypothetisch–deduktiv ist. Aber auch in technischen Bereichen läßt sich diese Art der Wissensmodellierung gut einsetzen, da auch hier, wie bereits erwähnt, in gewissem Maße hypothetisch–deduktiv gearbeitet wird.

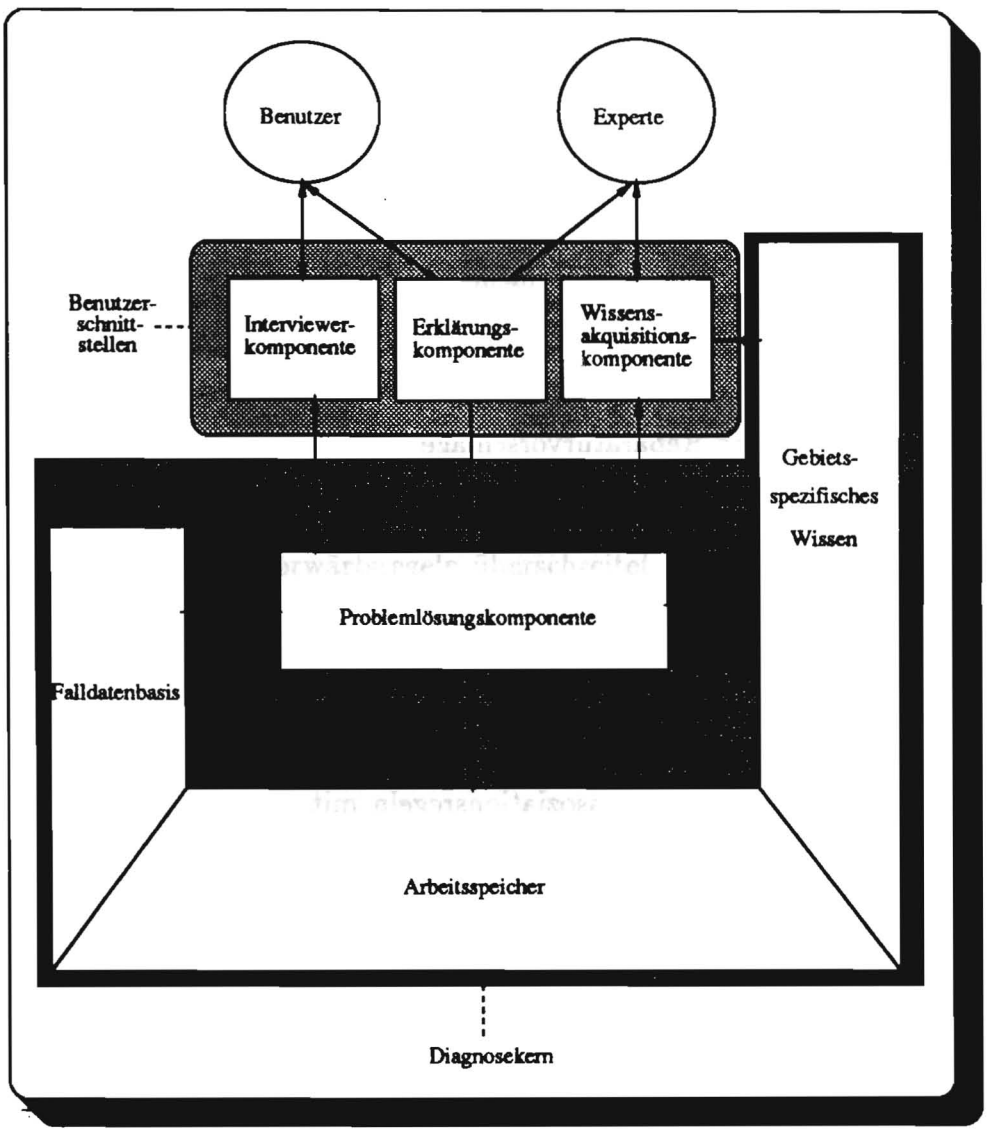

Abbildung 3.1: Allgemeiner Aufbau von MED2

Prinzipiell untergliedert sich die Wissensbank von MED2 in drei Teile:

- Arbeitsspeicher
- Gebietsspezifische Wissensbank
- Falldatenbasis

Das gebietsspezifische Wissen wird, wie bereits erwähnt, in einer separaten Wissensbank abgelegt, wobei Frames die gewählte Wissensrepräsentation darstellen. Insgesamt gibt es sechs verschiedene Frame–Typen, in denen das benötigte Wissen untergebracht wird:

1. *Pathokonzepte*: hierbei handelt es sich um Kontext–, Grob– und Feindiagnosen;
2. *Manifestationen*: Fragen und einfache Symptominterpretationen;
3. *Questionsets*: Gruppenbildung zusammengehöriger Manifestationen;
4. *Varianten*: therapeutische Varianten der Pathokonzepte, d. h. die mit einem Pathokonzept assoziierten Reparaturvorschläge;
5. *Explanationsets*: Liste von Pathokonzepten, die die momentan vorliegenden Symptome erklären;
6. *Regeln*: 16 Regel–Typen, die u. a. die Assoziationen zwischen Symptomen und Diagnosen ausdrücken, teilweise Abarbeitungsregeln sind, teilweise aber auch den nächsten Questionset bestimmen, etc.

Zu den *Regeln* sollte noch gesagt werden, daß hiermit das eigentliche assoziative Wissen modelliert wird, indem die Assoziationsregeln mit durch den Experten geschätzten Wahrscheinlichkeiten versehen werden. Dabei sind die wichtigsten Regel–Typen für die Abarbeitung:

- *Vorwärtsregeln*: Dienen ausschließlich zur Verdachtsgenerierung;
- *Rückwärtsregeln*: Dienen zur
  i) Verdachtsgenerierung
  ii) Diagnosebewertung
- *Vor– und Rückwärtsregeln*: Falls ein bestimmter Schwellwert bei der Bewertung durch Vorwärtsregeln überschritten wird, werden Rückwärtsregeln[2] zur vollständigen Bewertung verwendet[3];

[2]Die Reihenfolge: Vorwärtsregeln–Rückwärtsregeln ergibt sich aus der von MED2 verwendeten Hypothesize–and–Test–Strategie [Pup86].

[3]Jede Assoziationsregel ist mit Unsicherheiten in Form von Punktbewertungen versehen. Diese Punkte werden durch bestimmte Verfahren verrechnet [Pup86], wobei dann ein vorher festgelegter Schwellwert überschritten werden kann.

Aktive Regeln sind nun zum einen die Vorwärtsregeln, deren Kontext etabliert ist, und zum anderen die Rückwärtsregeln, deren Hypothese im Arbeitsspeicher ist. Das Konzept, einen Arbeitsspeicher getrennt von der eigentlichen Wissensbank einzurichten, dient zur Realisierung der Hypothesize–and–Test–Strategie, die zur Problemlösung verwendet wird. Dabei enthält der Arbeitsspeicher immer alle verdächtigen, jedoch noch nicht etablierten Diagnosen.

Durch das Arbeitsspeicherkonzept erhält man

- die Möglichkeit, Differentialdiagnosen[4] direkt vom System vergleichen zu lassen, bevor diese etabliert werden;
- eine Indikation von Questionsets[5] zur gleichzeitigen Aufklärung mehrerer Diagnosen im Arbeitsspeicher, d. h. es werden nicht nur Fragen nach der momentan am stärksten verdächtigten Diagnose gestellt, sondern unter Umständen auch nach der schwächsten.
- eine Erklärung des Systemvorgehens, da alle neuen Verdachtsdiagnosen in einem Fenster[6] geführt werden.

Eine Diagnose wird unter den folgenden Umständen in den Arbeitsspeicher aufgenommen:

- die Punktsumme der Vorwärtsregeln überschreitet einen Schwellwert und ist nach der Aktivierung der Rückwärtsregeln immer noch überschritten;
- eine Diagnose D1 ist in der Diagnoseheterarchie[7] Vorgänger einer Diagnose D2, welche im Arbeitsspeicher ist;
- eine Diagnose ist Differentialdiagnose einer auf Etablierung geprüften Diagnose;
- bei Interesse an einer bestimmten Diagnose kann diese vom Benutzer zur Mit–Untersuchung im Arbeitsspeicher verankert werden.

Ein weiterer Bereich der Datenbasis ist die Falldatenbasis. Bei MED2 handelt es sich hierbei jedoch nur im entfernten Sinne um eine Falldatenbasis. Hier werden die Fälle nur zu dem Zweck abgespeichert, bei eventuellen Folgesitzungen nicht erneut alle Abfragen neustarten zu müssen. Die Problemlösungskomponente kann einfach auf die bereits bearbeiteten Fälle zugreifen. Es sind keine Problemlösungen durch Ziehen analoger Schlüsse möglich.

[4] Differentialdiagnosen sind hierbei alle Diagnosen, die eine mögliche Erklärung für das Auftreten der vorliegenden Symptome darstellen.

[5] Questionsets stellen eine Sammlung von Fragen und einfachen Symptominterpretationen dar, die zusammengehören.

[6] Wenn man mit MED2 arbeitet, so wird der Bildschirm in verschiedene Fenster untergliedert: In einem Fenster erscheinen die zu markierenden Fragemenüs, in einem anderen Fenster die etablierten Diagnosen, in einem weiteren Fenster die Explanationsets (werden später erklärt) und in einem vierten Fenster der Inhalt des Arbeitsspeichers.

[7] Diese Diagnoseheterarchie ergibt sich aus den Pathokonzepten.

Als nächstes soll kurz auf die Benutzerschnittstellen von MED2 eingegangen werden. Insgesamt gibt es in MED2 drei Arten von Benutzerschnittstellen:

- Wissensakquisitionskomponente
- Interviewerkomponente
- Erklärungskomponente

Über die Wissensakquisitionskomponente kann der Experte sein Wissen in die Expertensystemshell eingeben. Dies geschieht einerseits durch Texteingabe, wobei zwischen den Sprachen Englisch und Deutsch gewählt werden kann, zudem können aber auch Graphiken eingelesen werden. Der Experte hat die Möglichkeit, das System über die Interviewerkomponente zu testen (d. h. Probeläufe zu starten), sich dabei die Erklärungen für die Vorgehensweise des Systems durch die Erklärungskomponente liefern zu lassen und schließlich, wenn nötig, noch in der aktuellen Sitzung Änderungen in der Wissensbank über die Wissensakquisitionskomponente vorzunehmen.

Der Benutzer eines mit MED2 entwickelten Systems wird über die Interviewerkomponente von dem System befragt. Dabei kann er zwischen zwei Modi wählen:

- *aktiv*: hier wählt der Benutzer die Questionsets aus, die seine Beobachtungen umfassen;
- *passiv*: das System selbst übernimmt die Initiative:
  - i) *erfahrungsgesteuert*, d. h. das System wählt anhand von Abarbeitungsregeln die Questionsets aus; eine Menge von Questionsets, die routinemäßig zur Abklärung des genaueren Zustandes führt, kann indiziert werden, wenn ein Pathokonzept oder eine Manifestation etabliert worden ist.
  - ii) *zielgerichtet*, d. h. falls mit der obigen Strategie kein Ergebnis erzielt wurde, wird der Questionset ausgewählt, der zur Überprüfung der Diagnosen im Arbeitsspeicher am nützlichsten ist, d. h. durch dessen Beantwortung beispielsweise einige Diagnosen ausgeschlossen werden können; damit wird geklärt, ob eine verdächtige Diagnose zutrifft, und Differentialdiagnosen werden unterschieden;
  - iii) *systematisch*, d. h. bei sehr allgemeiner Symptomatik kann es passieren, daß überhaupt keine brauchbaren Verdachtshypothesen herleitbar sind; zur Klärung solcher Situationen gibt es in MED2 ganz allgemeine Kontextdiagnosen, die dann die Dialogsteuerung übernehmen, um eine systematische, durch einen Experten festgelegte Erfassung der Symptomatik zu ermöglichen.

Auch auf die Problemlösungskomponente von MED2 soll in dieser Arbeit nur kurz eingegangen werden. Näheres ist unter [Pup87] und [Pup86] zu finden. Prinzipiell ist festzustellen, daß zwischen der Datenvorverarbeitung (Database Reasoning) und der eigentlichen diagnostischen Auswertung der Daten (Diagnostic Reasoning) unterschieden wird.

Bei der Datenvorverarbeitung handelt es sich um einfache Symptominterpretationen, was bedeutet, daß beispielsweise quantitative Daten in qualitative Daten umgewandelt werden, daß die Dauer eines Symptoms aus Anfang und Ende des Gültigkeitsintervalles berechnet wird, oder daß einige nützliche Indizes berechnet werden.

Bei der eigentlichen diagnostischen Auswertung handelt es sich um die Verdachtsgenerierung und um die Hypothesenüberprüfung. Dabei bedient sich das System der Hypothesize–and–test–Strategie, wodurch eine Konzentration auf die wichtigsten Verdachtsdiagnosen erreicht wird.

3.1.2 MegaFilEx

Ein weiteres Expertensystem, das mit heuristisch modelliertem Wissen arbeitet, ist MegaFilEx (MegaFile–Expertensystem), das in der Endprüfung der Plattenspeicher des Typs MegaFile von SIEMENS eingesetzt wird. MegaFile ist ein 5 1/4–Zoll Festplattenspeicher mit einer Kapazität von 300 MByte [Kar89].

Zunächst soll mit Abbildung 3.2 der allgemeine Aufbau von MegaFilEx gezeigt werden.

MegaFilEx befragt den Benutzer über die Dialogschnittstelle zu dem vorliegenden Fehler in Form von Menüs, in denen Zutreffendes vom Benutzer angekreuzt werden soll. Dabei soll sich das System dem Wissensstand des jeweiligen Benutzers anpassen, indem es einem erfahrenen Benutzer z. B. das Einbringen eigener Fehlerhypothesen ermöglicht. Weitere Eingaben für MegaFilEx sind Tests (u. a. Funktionstest (Einschalttest), Streßtest (50 Stunden Dauertest), Schreib– Lesetest, ... , Endtest), anhand derer ein Fehler festgestellt und ein Fehlerprotokoll erstellt wird, das dann weiter ausgewertet wird. Der Benutzer kann jedoch auch explizit von MegaFilEx aufgefordert werden, weitere Tests durchzuführen, wenn weitere Daten benötigt werden.

Die Wissensbasis wurde mit Hilfe eines Experten erstellt und umfaßt etwa tausend Regeln.

Hier soll noch erwähnt werden, daß eine spezielle Wissensakquisitionskomponente nicht vorliegt, sondern daß das gebietsspezifische Wissen von einem Wissensingenieur durch Befragung eines Experten akquiriert wurde. Der Wissensingenieur erstellte dann auch die benötigte Wissensbasis.

Die Wissensbasis von MegaFilEx wird in drei Bereiche untergliedert:

- *Fehlerauswertung*: Hierbei handelt es sich um Auswahlregeln, die es ermöglichen sollen, eine geeignete anfängliche Hypothesenliste zu erstellen, die dann nach und nach abgearbeitet werden soll. Ein Beispiel für eine solche Auswahlregel ist in (Abbildung 3.3) zu finden. Wie in dem Beispiel zu erkennen ist, sind die Regeln mit Bewertungen versehen, die der Experte aufgrund seines Erfahrungsschatzes vorgenommen hat.

- *Testhierarchien*: Bei den Testhierarchien handelt es sich um Diagnose– und Fortsetzungsregeln. Durch die Auswahlregeln, die die Fehlerbeschreibung zu dem aktuell aufgetretenen Fehler analysieren, erhält man eine Liste von Verdachtshypothesen

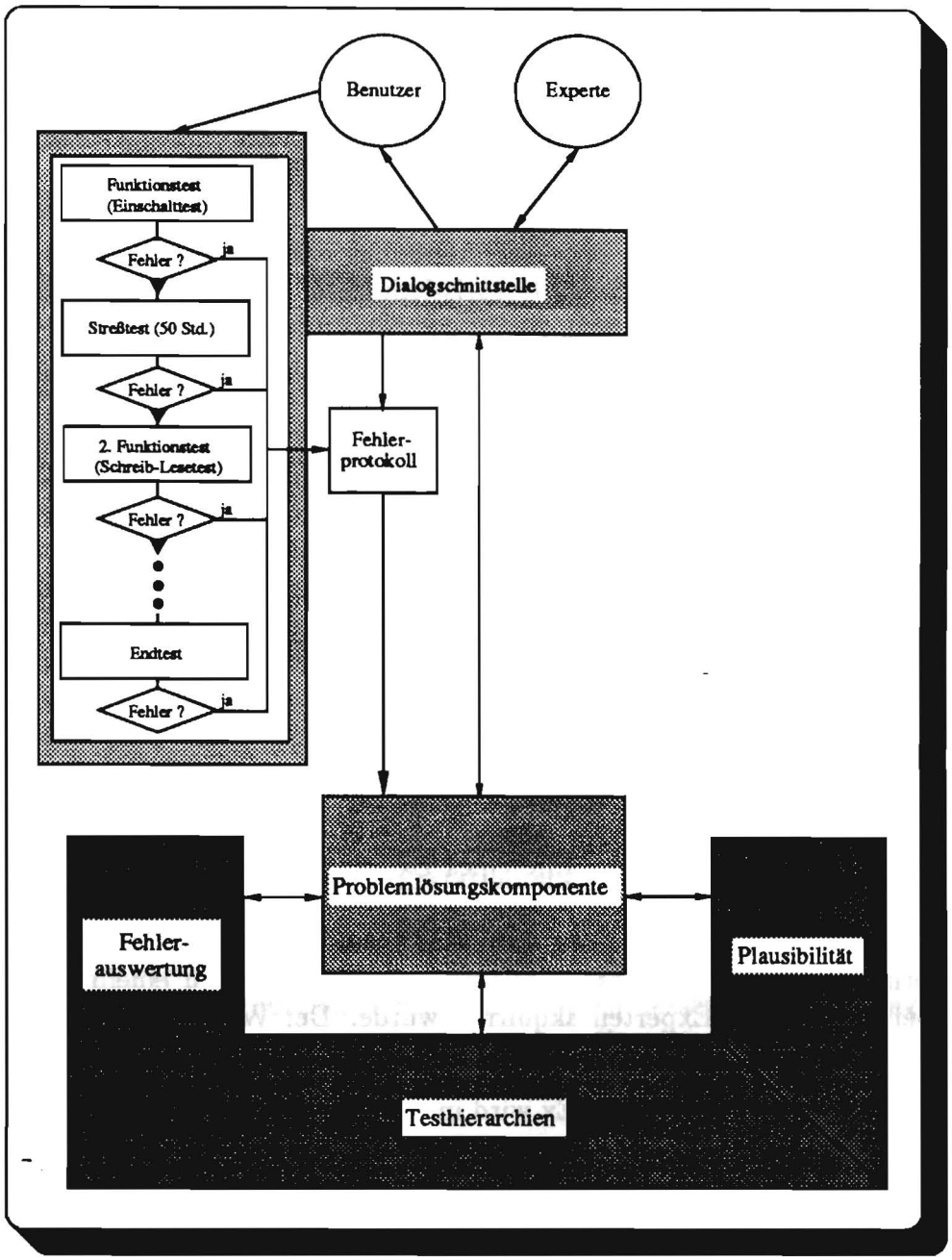

Abbildung 3.2: Aufbau von MegaFilEx

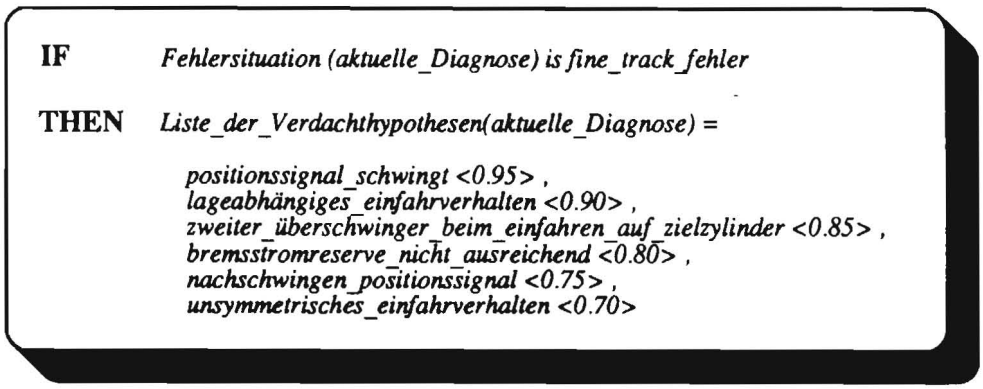

Abbildung 3.3: Eine Auswahlregel aus der Wissensbasis von MegaFilEx

für die vorliegende Fehlersituation. Dabei bezieht sich jede Verdachtshypothese auf eine mögliche Fehlerursache. Jeder Fehlerursache ist nun eine Testhierarchie zugeordnet, die den Zusammenhang zwischen den zur Bestimmung der Fehlerursache notwendigen Tests und den aus der Testauswertung resultierenden Annahmen über die Fehlerursache beschreibt. Ein Beispiel für eine solche Testhierarchie wird in Abbildung 3.4 graphisch dargestellt.

- *Plausibilität*: Ist die Liste der Verdachtshypothesen abgearbeitet, so hat das System einen oder mehrere Reparaturvorschläge bestimmt. Da bei der Auswertung der Hypothesen bereits abgeleitete Ergebnisse nicht berücksichtigt werden, muß im Falle von Mehrfachfehlern noch eine Plausibilitätskontrolle durchgeführt werden. Durch diese Kontrolle werden dann überflüssige Reparaturvorschläge ausgesiebt.

Alles in allem handelt es sich bei MegaFilEx um eine relativ typische Anwendung von heuristisch modelliertem Wissen (durch die Testhierarchien sind in gewissem Maße auch Entscheidungsbäume involviert). Es gilt noch als Expertensystem der ersten Generation, bei dem man sich ausschließlich auf die Verwendung heuristischen Wissens bei der Diagnoseerstellung beschränkt hat.

Einige Teile von MegaFilEx sind noch in der Weiterentwicklung. Vor allem eine Erweiterung der Wissensbasis auf kausal modelliertes Wissen halten die Entwickler für unbedingt erforderlich, d. h. sie beabsichtigen beispielsweise, den Aufbau des zu diagnostizierenden Plattenspeichers explizit in die Wissensbasis aufzunehmen.

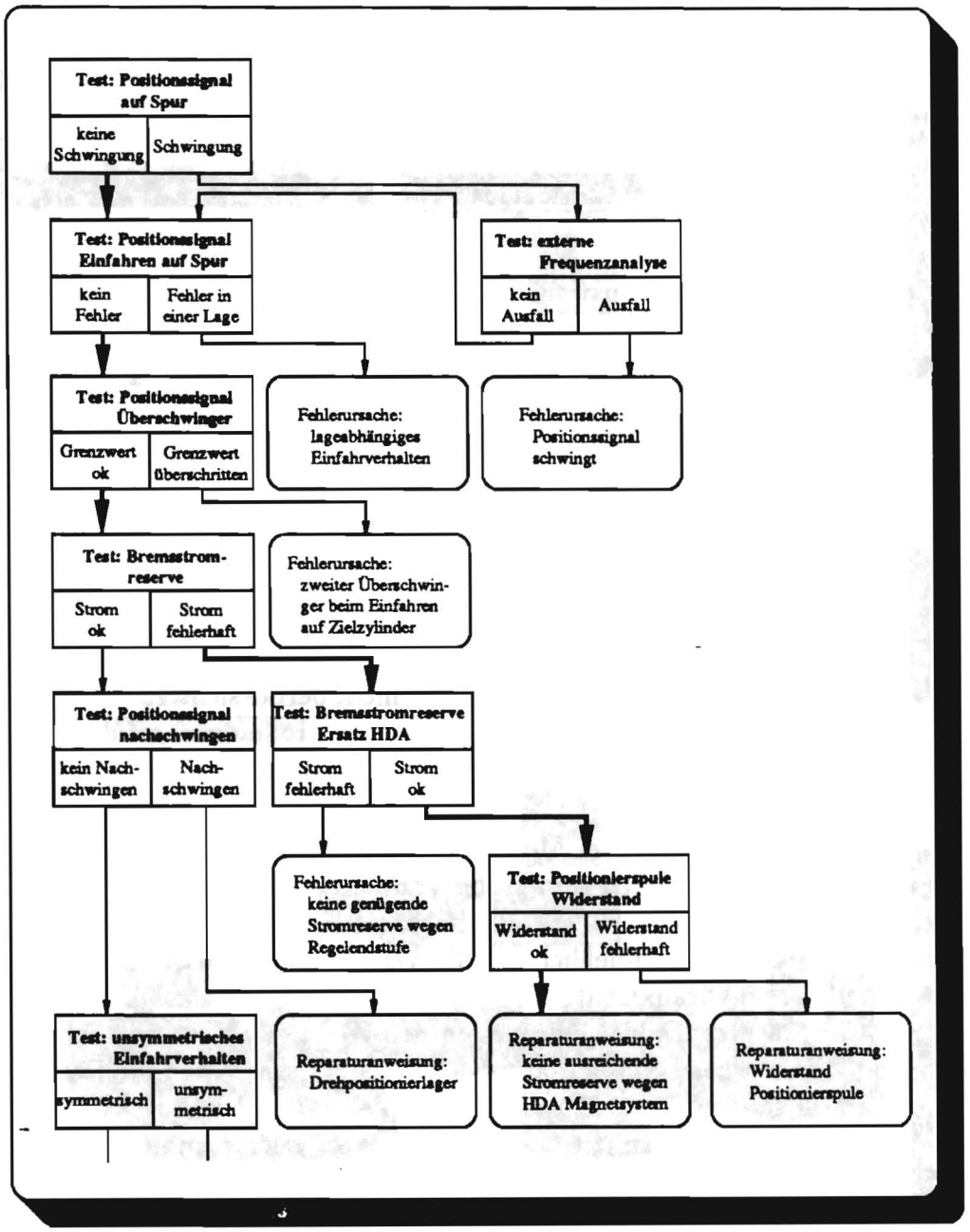

Abbildung 3.4: Testhierarchie *„Fine–Track–Fehler“*

3.1.3 Ein Expertensystemwerkzeug zur Diagnose an CNC–Maschinen

Ein Expertensystem, das vor allem für das ARCTEC–Projekt interessant ist, ist ein Expertensystemwerkzeug zur Diagnose an CNC–Maschinen, das am Fraunhofer Institut (IAO) in Stuttgart entwickelt wurde [EPZ87]. Es handelt sich um ein weiteres heuristisches System.

Auch bei diesem Expertensystemwerkzeug findet sich die typische Architektur eines Expertensystems wieder, wie in Abbildung 3.5 gezeigt.

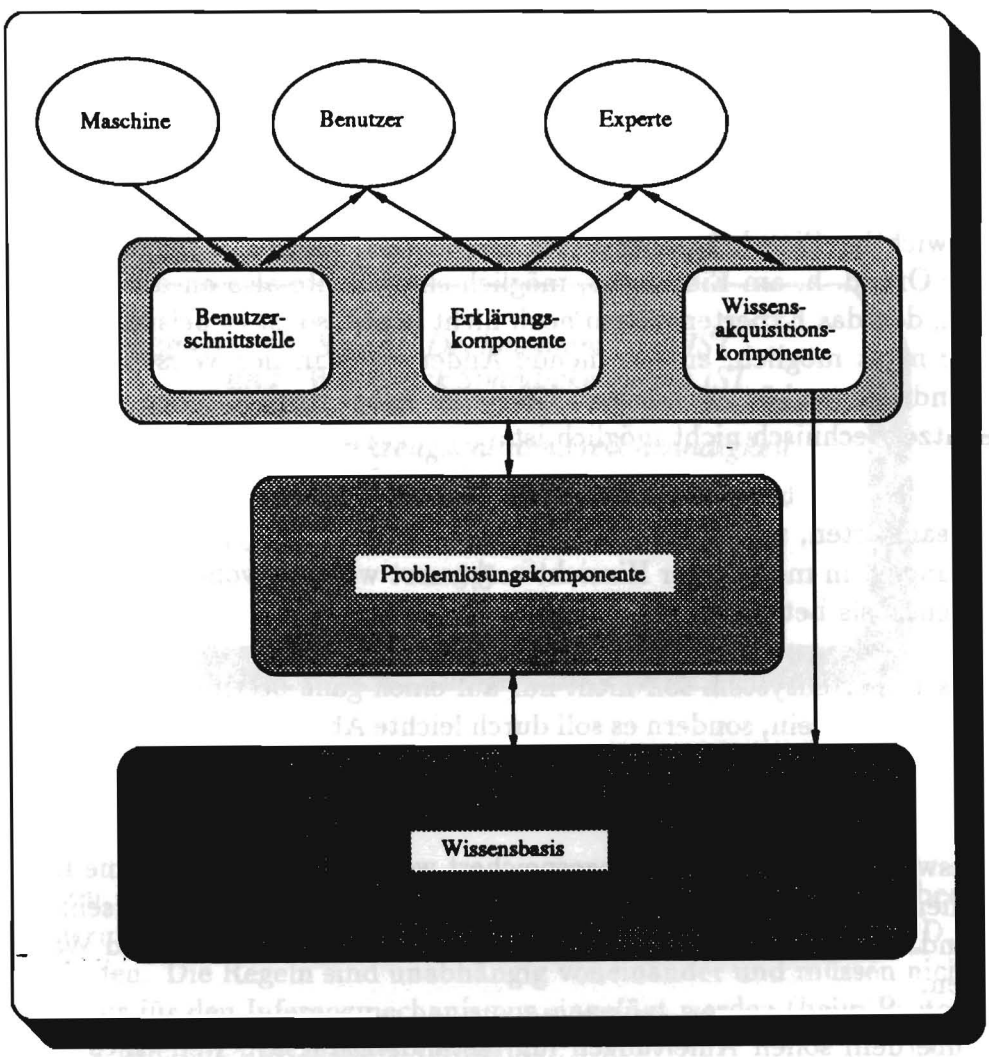

Abbildung 3.5: Aufbau des Expertensystemwerkzeugs zur Diagnose an CNC–Maschinen

Interessant bei diesem System ist, daß es in zwei voneinander getrennten Versionen existiert:

- Die Version, die in *LISP* implementiert wurde, dient dem Experten zur Erstellung des Expertensystems. Er kann mit seiner Hilfe einerseits das Wissen in die Wissensbasis eingeben, andererseits kann er das System auch zum Testen der von ihm erstellten Wissensbasis verwenden. In letzterem Fall fungiert der Experte als Benutzer des Systems.
- Die Version des Systems, die in *C* implementiert wurde, stellt das eigentliche Expertensystem dar, d. h. das System, das dem Maschinenbediener oder dem Servicetechniker als Diagnosewerkzeug dienen soll.

Das zur Nutzung vorgesehene Expertensystem ist in die zu diagnostizierende Maschine integriert, was bedeutet, daß es direkt Daten von der Maschine erhalten kann. Wie auch in Abbildung 3.5 zu sehen ist, muß aber auch der Benutzer des Expertensystems noch von ihm erfragte Daten eingeben.

Eine wichtige Einschränkung dieses Systems ist, daß Änderungen an der Wissensbasis nicht vor Ort, d. h. am Einsatzort, möglich sind. Sollte also an der Maschine ein Fehler auftreten, den das Expertensystem noch nicht kennt, so ist es beispielsweise dem Servicetechniker nicht möglich, entsprechende Änderungen in der Wissensbasis vorzunehmen. Solche Änderungen können nur vom Hersteller selbst vorgenommen werden, da es für den Endbenutzer technisch nicht möglich ist.

Es handelt sich bei diesem Expertensystemwerkzeug um die Weiterentwicklung eines bereits realisierten, in der Anwendung befindlichen Prototypen [EPZ87]. Diese Weiterentwicklung ist in mehrfacher Hinsicht verbessert worden, wobei die meisten Änderungen die Wissensbasis betreffen:

- Das Expertensystem soll nicht nur auf einen ganz bestimmten Maschinentypen zugeschnitten sein, sondern es soll durch leichte Abwandlungen von Fakten und Regeln auf verschiedene Maschinentyp übertragbar sein, wobei die Maschinen jedoch aus den gleichen Maschinenkomponenten bestehen.
- Desweiteren sollen Daten abgespeichert werden, die über die reine Diagnose hinausgehen, für den Diagnoseverlauf aber durchaus von Bedeutung sein können. Dabei handelt es sich um Daten wie Standort, Alter der Maschine und Wartungsmaßnahmen.
- Außerdem sollen Anleitungen für Instandsetzungsarbeiten eines Servicetechnikers festgehalten werden, d. h. Tests durch Veränderungen bzw. Ausbauen von Maschinenteilen sollen möglich sein.
- Schließlich soll die Änderbarkeit der Wissensbasis flexibler gestaltet werden.
- Die wohl bedeutendste Änderung ist jedoch, daß in der Weiterentwicklung des Prototypen eine Problemlösungskomponente existiert.

Die Wissensbasis ist folgendermaßen aufgebaut:

- *Fakten:* Die Beschreibung eines Fakts erfolgt durch eine Faktdefinition, die bei der Beschreibung einer Kompnente zwischen *Maschinenteilname* und *Eigenschaftsname des Maschinenteils* unterscheidet. Durch den Maschinenteilnamen erhält man die Stelle, an der das Maschinenteil in der Maschine eingebaut ist, und durch den Eigenschaftsnamen erhält man das Symptom, das beobachtet oder gemessen werden soll. Dabei kann ein Maschinenteil mehrere Eigenschaften besitzen. Sinn dieser Zweiteilung ist es, eine klare semantische Unterscheidung zu erhalten und somit eine bessere Übersicht innerhalb der Wissensbasis.

- Einer Faktdefinition werden nun Attribute hinzugefügt. In den Attributen sind Informationen enthalten, die teilweise neu sind, teilweise jedoch auch in den Regeln enthalten waren, wie sie in dem Prototypen vorlagen. Durch die Attribute können detailliertere Definitionen und Ableitungen gemacht werden. Außerdem werden durch die Faktattributierung Informationen aus der dynamischen Regelstruktur in die statische Faktdefinition transformiert.

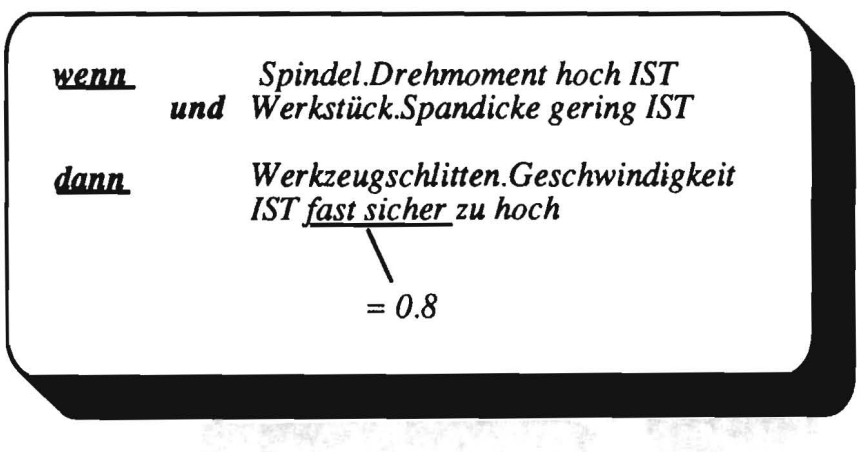

Abbildung 3.6: Beispielregel aus der Wissensbasis

- Die Regeln des Expertensystemwerkzeugs sehen so aus, daß die Vorbedingungen und Folgerungen im Gegensatz zu dem Prototypen nur noch mit UND verknüpft werden dürfen. Die Regeln sind unabhängig voneinander und müssen nicht mehr in eine Struktur für den Inferenzmechanismus eingefügt werden (beim Prototyp waren Wissen und Inferenz miteinander verknüpft).
  Weiterhin handelt es sich jedoch um heuristische Regeln, d. h. um Regeln, die auf den Erfahrungen eines Experten beruhen, und dementsprechend mit Erfahrungswerten versehen sind. Ein Beispiel für eine solche Regel ist in Abbildung 3.6 zu finden.

3.2 Modellbasierte Diagnose–Expertensysteme

In diesem Kapitel sollen nun einige Diagnose–Expertensysteme vorgestellt werden, die zur Diagnoseerstellung modellbasiertes Wissen heranziehen.

Auf diesem Gebiet wird in den letzten Jahren immer mehr geforscht, so daß hier schon einige Systeme existieren, die beschrieben werden können. Stellvertretend sind dies in dieser Arbeit das System von Davis, QUASIMODIS und die Expertensystemshell KEE.

3.2.1 Davis' System

Eines der klassischen Systeme, die mit kausal modelliertem Wissen arbeiten, ist das System von Davis, das am Massachusetts Institute of Technology in Cambridge (USA) unter Leitung von Randall Davis entwickelt wurde [Dav84]. Davis' System soll die Diagnose an elektronischen Schaltkreisen durchführen, und wurde bereits an mehreren Beispielen getestet.

Zunächst wird in Abbildung 3.7 der allgemeine Aufbau von Davis' System dargestellt. Erwähnenswert ist die relativ komfortable Dialogschnittstelle, bei der der Experte die

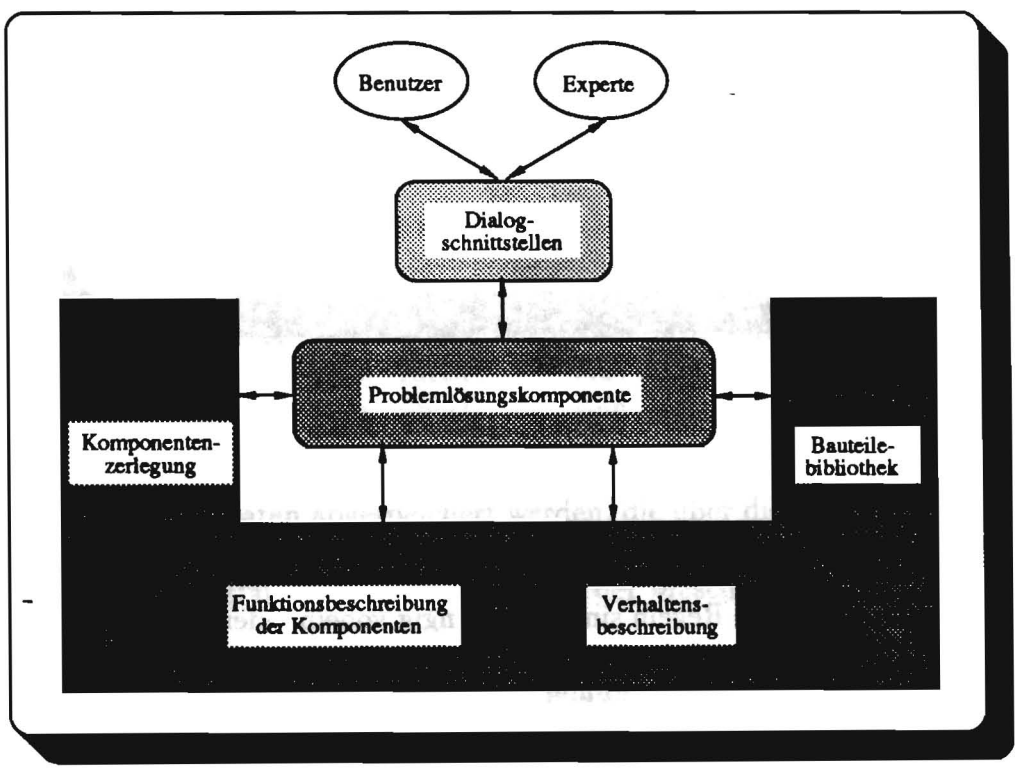

Abbildung 3.7: Prinzipieller Aufbau von Davis' System

Möglichkeit haben soll, Wissen in Form von Schaltplänen in das System einzugeben, das dann durch einen Parser für die Wissensbasis entsprechend aufgearbeitet wird [Dav84].

Insgesamt wird das Wissen über das zu diagnostizierende technische Gerät funktional modelliert, d. h. es wird mit modellbasiertem Wissen gearbeitet. Dies spiegelt sich auch in dem Aufbau der Wissensbasis wider, die in die Bereiche

- *Strukturbeschreibung des technischen Geräts*
  - *physikalische Organisation des technischen Geräts*
  - *funktionale Organisation des technischen Geräts*
- *Verhaltensbeschreibung des technischen Geräts*
- *Bauteilebibliothek*

untergliedert ist.

Das Wichtigste bei dieser Untergliederung ist die Aufteilung in Strukturwissen und Verhaltenswissen.

Die Beschreibung der Struktur des technischen Geräts beinhaltet dabei zum einen die Beschreibung des physikalischen Aufbaus des Geräts und zum anderen die Beschreibung der Funktion der einzelnen Komponenten des Geräts, und ist somit komponentenorientiert. Damit erhält man zwei verschiedene, jedoch miteinander verknüpfte Beschreibungen des zu diagnostizierenden Geräts.

Die Strukturbeschreibung findet auf verschiedenen Ebenen mit unterschiedlichem Detaillierungsgrad statt; sie ist also hierarchisch.

Die Basis–Ebene der Strukturbeschreibung wird, wie in Abbildung 3.8 an dem Beispiel eines Addierers gezeigt, aus drei Konzepten aufgebaut: Module, Ports und Terminale.

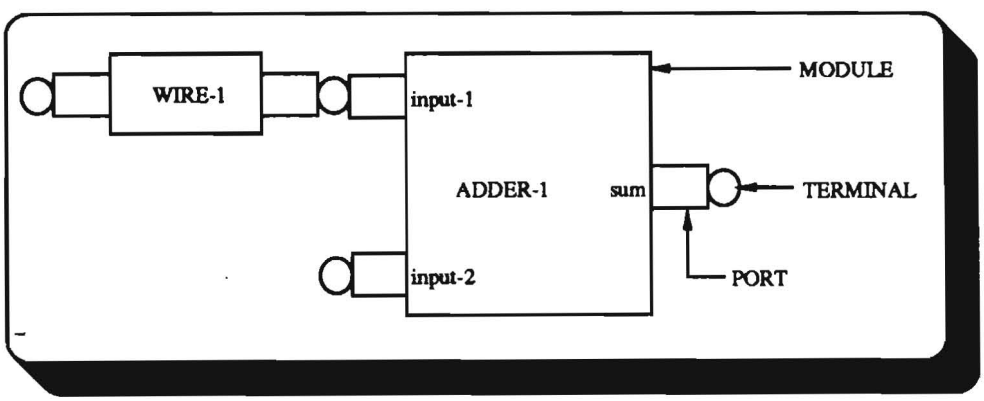

Abbildung 3.8: Strukturbeschreibung auf oberster Ebene

Dabei kann man sich Module wie Black–Boxen vorstellen. Ports sind die Stellen, an denen Informationen in die Module hinein- oder aus den Modulen herausfließt, wobei jeder Port wiederum mindestens zwei Terminale hat: ein Terminal an der Außenseite und mindestens ein Terminal an der Innenseite des Ports. Die Terminale sind primitive

Elemente, an denen man auf der einen Seite den genauen Informationsfluß untersuchen kann, die auf der anderen Seite jedoch selbst keine interessante Unterstruktur haben.

Zwei Module können nun verbunden werden, indem ihre Terminale überlagert werden. In Abbildung 3.8 sieht man dies zwischen WIRE-1 und ADDER-1.

Wie bereits erwähnt wurde, wird die Struktur eines technischen Geräts in Davis' System auf verschiedenen Ebenen verschiedenen Detaillierungsgrades beschrieben. Die Beschreibung des physikalischen Aufbaus wird bis zu den einzelnen Chips aufgeschlüsselt; die Beschreibung der Funktion der einzelnen Komponenten endet mit den einzelnen Gattern. Dabei sind die Chips einerseits und die Gatter andererseits selbst wieder Black-Boxen, von denen nur ihr Funktionieren oder Nichtfunktionieren interessiert. Sie liegen also auf der Ebene mit dem höchsten Detaillierungsgrad. In Abbildung 3.9 ist das Gerät aus Abbildung 3.8 detaillierter beschrieben. Auf dieser Beschreibungsebene sieht man auch,

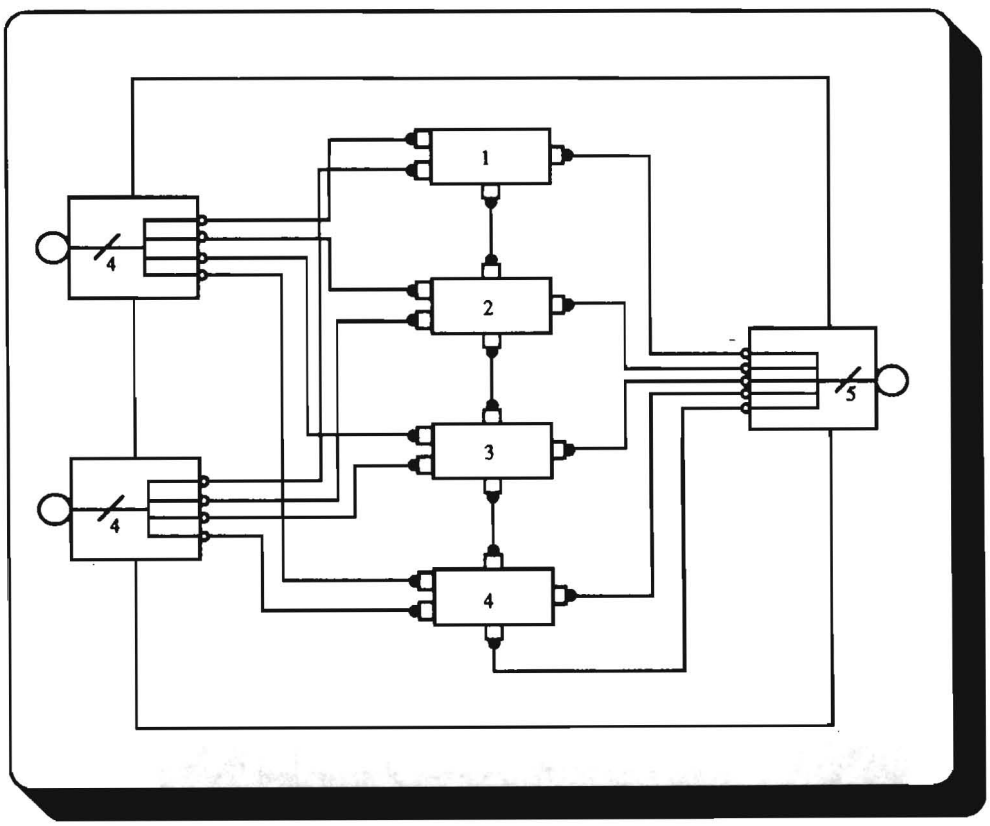

Abbildung 3.9: Strukturbeschreibung auf tieferer Ebene

warum ein Port mehrere Terminale auf der Innenseite haben kann: Ports liefern die wichtige Information zum Wechseln von Abstraktionsleveln. In dem ausgewählten Beispiel bedeutet dies, daß es durchaus nützlich sein kann, den Informationsfluß über WIRE-1 als Integer–Zahl zwischen 0 und 15 zu betrachten, daß es aber auch von Nöten sein kann die vier Bits, die aus dem Addierer in den Port kommen einzeln zu untersuchen.

In der Wissensbasis wird die Strukturbeschreibung beispielsweise eines Moduls in Form von Kommandos abgelegt. Ein Beispiel für eine solche Beschreibung ist in Abbildung 3.10 zu finden. In Abbildung 3.10 ist ein Prototyp eines Ripple–Carry–Addierers[8] definiert.

```
(definemodule adder NBitsWide
  (repeat NBitsWide i
    (part slice-i adder-slice)
    (run-wire (input-1 adder) (input-1 slice-i))
    (run-wire (input-2 adder) (input-2 slice-i))
    (run-wire (ouput slice-i) (sum adder))
  (repeat (-NBitsWide 1) i (run-wire (carry-out slice-i)
                                     (carry-in slice-[i+1])))
  (run-wire (carry-out slice-[NBitsWide-1]) (sum adder)))
```

Abbildung 3.10: Strukturbeschreibung eines Moduls

Belegt man nun den Breiten–Parameter in diesem Prototyp, so erhält man eine Instanz eines Ripple–Carry–Addierers.

Sowohl der Prototyp, als auch die Instanzen eines Prototyps werden in der Wissensbasis abgelegt. Damit erreicht man, daß Informationen, die allen Instanzen eines Prototyps gemeinsam sind, einmal in der Wissensbasis gespeichert werden. Die Definition eines Prototypen wird schließlich in der Bauteilebibliothek abgelegt. Mit diesem Prototypen ist es nun möglich, eine Instanz erst dann weiter zu detaillieren, wenn die entsprechende Information benötigt wird. Damit kann man natürlich sehr viel Speicherplatz einsparen. In dem oben gewählten Beispiel bedeutet dies, daß der Addierer eigentlich aus zwei Halbaddierern und einem ODER–Gatter besteht und daß jeder Halbaddierer wiederum aus einem XOR– und einem UND–Gatter besteht. Die Initialisierung sieht jedoch so aus, daß man einen Addierer mit zwei Eingaben und einer Ausgabe hat, anstatt vier Beschreibungen von unterschiedlichem Detaillierungsgrad und mit zwanzig Gattern.

Die Beschreibung der physikalischen Organisation eines Moduls besteht aus Begriffen wie Gehäuse, Schaltplatte, Chip, etc. ; die Beschreibung der funktionalen Organisation besteht aus Begriffen wie Adder, Slice, Half–Adder, etc. . Da jede Komponente eine physikalische und eine funktionale Beschreibung hat, erhält man somit zwei miteinander verknüpfte Beschreibungen. In Abbildung 3.11 ist die Verflechtung der beiden Beschreibungen für den Addierer graphisch dargestellt.

Neben der Strukturbeschreibung ist in Davis' System auch eine Verhaltensbeschreibung realisiert worden. Mit Verhaltensbeschreibung ist hier die Black–Box–Beschreibung einer Komponente gemeint: wie verläßt die Information die Komponente im Verhältnis

[8]Ein Ripple–Carry–Addierer ist ein Addierer, bei dem der Übertrag mit jedem Takt weiter nach vorne gereicht wird.

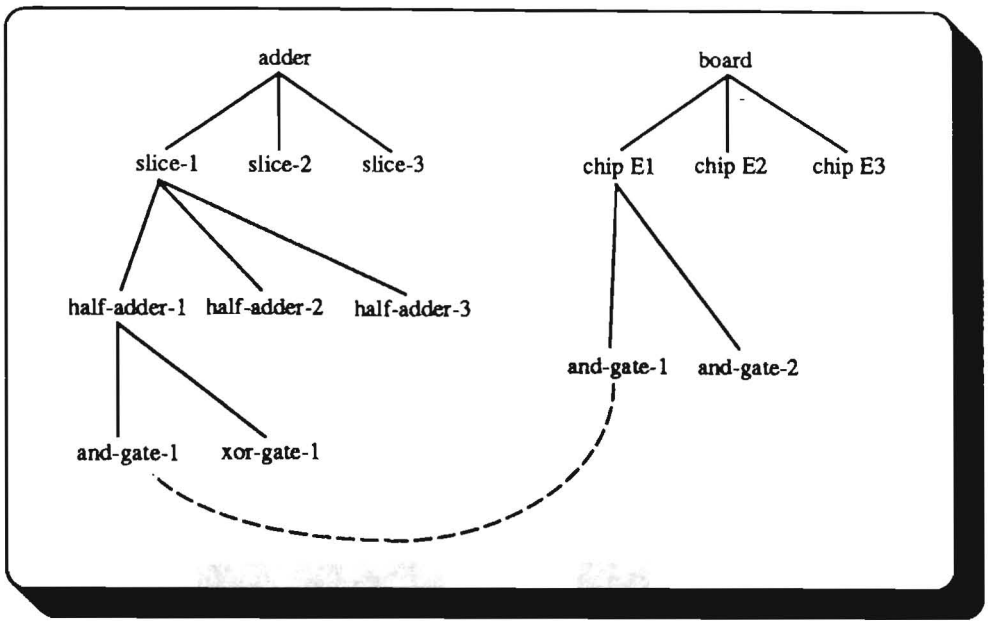

Abbildung 3.11: Verflechtung von physikalischer und funktionaler Beschreibung

dazu, wie sie sie erreicht hat? Dabei ist das Verhalten des Gerätes und seiner Komponenten statisch modelliert worden.

In Davis' System wurde zur Verhaltensbeschreibung ein Constraint–artiger Ansatz gewählt. Das Verhalten des obigen Addierers wird beispielsweise durch die Relationen der logischen Pegel der Terminale auf den Ports *input-1*, *input-2* und *sum* ausgedrückt. Diese Umsetzung ist in Abbildung 3.12 zu finden. Diese Regeln sind jedoch nicht gerichtet,

```
to get sum from (input-1 input-2) do (+ input-1 input-2)
to get input-1 from (sum input-2) do (- sum input-2)
to get input-2 from (sum input-1) do (- sum input-1)
```

Abbildung 3.12: Regeln zur Verhaltensbeschreibung eines Addierers

und eignen sich somit lediglich für eine Verhaltensbeschreibung eines analogen Schaltkreises. Die letzten beiden Regeln sind nicht gerade eine gute Verhaltensbeschreibung eines Addierers: das Gerät führt keine Subtraktion durch; legt man einen logischen Pegel am Ausgang an, so verursacht das Erscheinen eines logischen Pegels an dem einen Eingang nicht das Erscheinen eines logischen Pegels an dem anderen Eingang. Somit beschreiben die beiden letzten Regeln mehr die Inferenzen, die aus dem Verhalten des technischen Gerätes gezogen werden.

Dies hat dazu geführt, daß bei Davis' System eine Unterscheidung getroffen wird zwischen *Simulationsregeln*, die das digitale Verhalten, d. h. den elektrischen Fluß in dem Gerät beschreiben, und *Inferenzregeln*, die die Schlüsse, die man aus dem Geräteverhalten zieht, beschreiben. Diese Unterscheidung wurde jedoch nur aus Gründen der Übersicht getroffen, damit man eben genau erkennen kann, was Simulation und was Inferenz ist.

Die Simulations- und Inferenzregeln werden in zwei parallelen, aber getrennten Netzwerken festgehalten. Um die Trennung zu erhalten, erhält jedes Terminal zwei Slots: in einem Slot werden die durch die Simulationsregeln berechneten Werte abgespeichert, in dem anderen Slot die durch die Inferenzregeln berechneten Werte. In Abbildung 3.13 wird auch dies noch einmal graphisch dargestellt. Sinn einer solchen Realisierung der

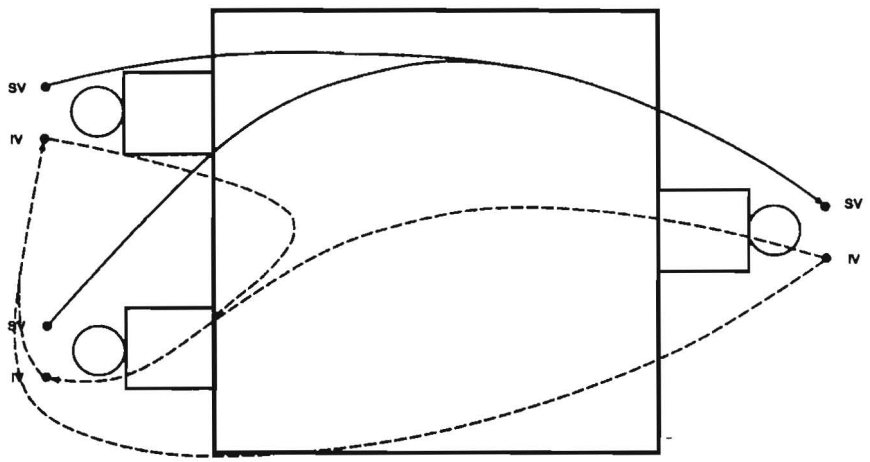

Abbildung 3.13: Addierer mit einer Simulations- und zwei Inferenzregeln

Netzwerke ist es auch, daß man einem Terminal einen Wert zuordnen kann und dann beobachten kann, was bei der Simulation passiert.

Insgesamt läßt sich feststellen, daß es sich bei Davis' System um funktional, hierarchisch, komponentenorientiert und statisch modelliertes Wissen über das zu diagnostizierende technische Gerät handelt.

Wie bereits erwähnt stellt Davis' System eines der klassischen Systeme auf dem Gebiet der mit modellbasiertem Wissen arbeitenden Expertensysteme dar. Seine Arbeit lieferte die Grundlage für eine ganze Reihe anderer Arbeiten auf diesem Gebiet.

3.2.2 QUASIMODIS

Ein weiteres System, das mit kausal modelliertem Wissen arbeitet, ist QUASIMODIS (QUAlitative SImulation zur MOdellbasierten Diagnose dynamischer Systeme), das von Robert Rehbold in seiner Dissertation vorgestellt wird [Reh91] und in der Arbeitsgruppe von Herrn Prof. Dr. M. M. Richter an der Universität Kaiserslautern entwickelt wurde [Ric91].

Wie bereits in dem Abschnitt über modellbasiertes Wissen ausgeführt wurde (Kapitel 2), kann man die kausale Wissensmodellierung in funktionale und pathophysiologische Wissensmodellierung, und diese wiederum in dynamische und statische Wissensmodellierung unterteilen. Bei QUASIMODIS handelt es sich um ein System, bei dem funktionales, dynamisches Wissen verarbeitet wird. Für den Gesamtaufbau ist es wichtig zu erwähnen, daß QUASIMODIS ein übergeordnetes Expertensystem benötgt (beispielsweise ein heuristisches), welches ihm die Fehlerhypothesen, die überprüft werden sollen, liefert. Auf die Gründe hierfür wird später näher eingehen. Zuerst jedoch soll der Aufbau von QUASIMODIS in Abbildung 3.14 gezeigt werden.

QUASIMODIS kann, wie Abbildung 3.14 zu entnehmen ist, als Untermodul von einem anderen Expertensystem verwendet werden. Die Kommunikation mit dem Benutzer, dem Experten und gegebenenfalls der zu diagnostizierenden Maschine wird von einem übergeordneten, hier heuristischen System übernommen. Dieses übergeordnete Expertensystem liefert nun Fehlerhypothesen an QUASIMODIS, die von diesem überprüft werden sollen. Nötigenfalls werden weitere Symptome erfragt. Da die Benutzerschnittstellen von dem übergeordneten System realisiert werden sollen, wird darauf nicht weiter eingegangen[9], sondern direkt zur Wissensmodellierung in QUASIMODIS kommen, und damit zu der Begründung dafür, warum QUASIMODIS ein übergeordnetes Expertensystem zur Generierung der Fehlerhypothesen benötigt. Der Grund ist einfach in der Art des Wissens zu finden, mit dem QUASIMODIS arbeitet.

Bisher arbeiten die meisten modellbasierten Expertensysteme mit statischem Wissen. Diesen Expertensystemen gegenüber steht unter anderem QUASIMODIS, mit dem ein integriertes Konzept zur Modellierung und Simulation einerseits und deren Nutzung für die Diagnoseerstellung an dem modellierten Gerät andererseits entwickelt wurde. QUASIMODIS modelliert dabei auch die Dynamik des zu diagnostizierenden technischen Geräts. Dabei wird zum einen das fehlerfreie, funktionstüchtige Gerät modelliert, zum anderen werden, nach Lieferung einer oder mehrerer Fehlerhypothesen, die zugehörigen fehlerhaften Modelle erstellt, die es ermöglichen, den genauen Fehler festzustellen. Da die Modelle zur Simulation der einzelnen Fehler nur nach Bedarf erstellt werden, weil bei der Berücksichtigung dynamischer Vorgänge (z. B. Regelkreise, Rückkopplungen) in dem zu diagnostizierenden komplexen Gerät eine Erstellung sämtlicher Fehlermodelle fast unmöglich und nur mit erheblichem Mehraufwand verbunden ist, spricht man bei QUASIMODIS von einem interpretierenden Ansatz. Die hinter QUASIMODIS steckende Idee soll in Abbildung 3.15 verdeutlicht und anschließend erklärt werden.

Entspricht das tatsächlich beobachtete Verhalten eines technischen Systems S nicht

[9] Als Beispiel für eine solche Realisierung kann MED2 betrachtet werden.

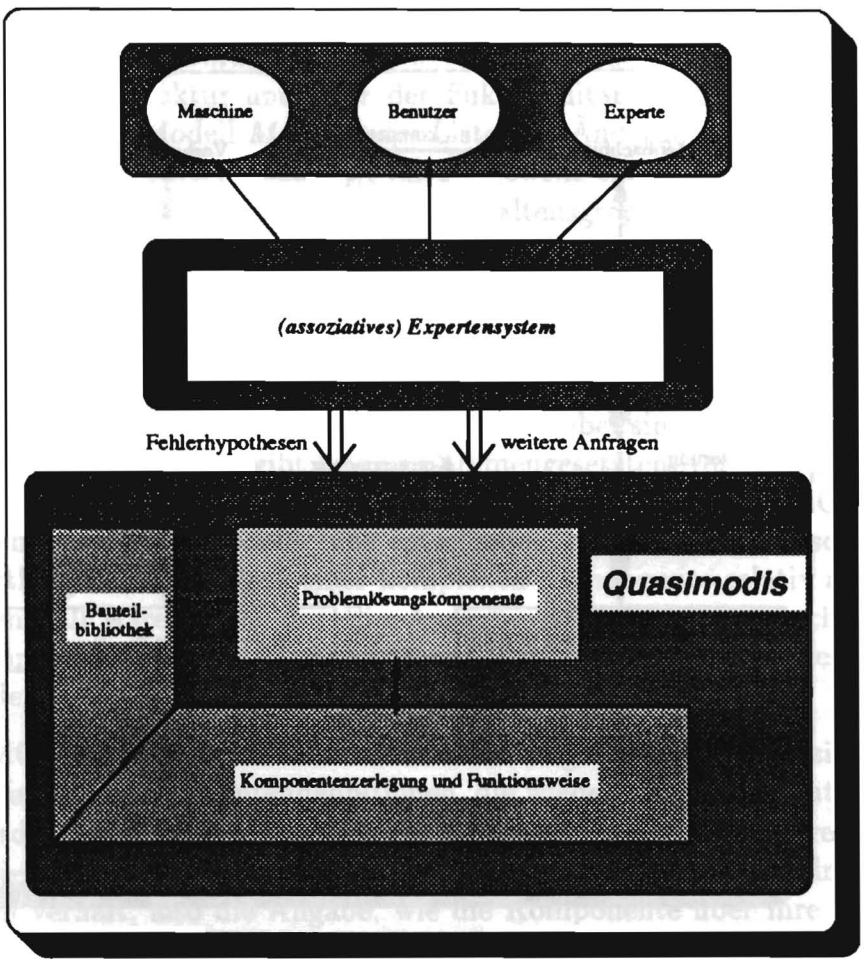

Abbildung 3.14: Allgemeiner Aufbau von QUASIMODIS

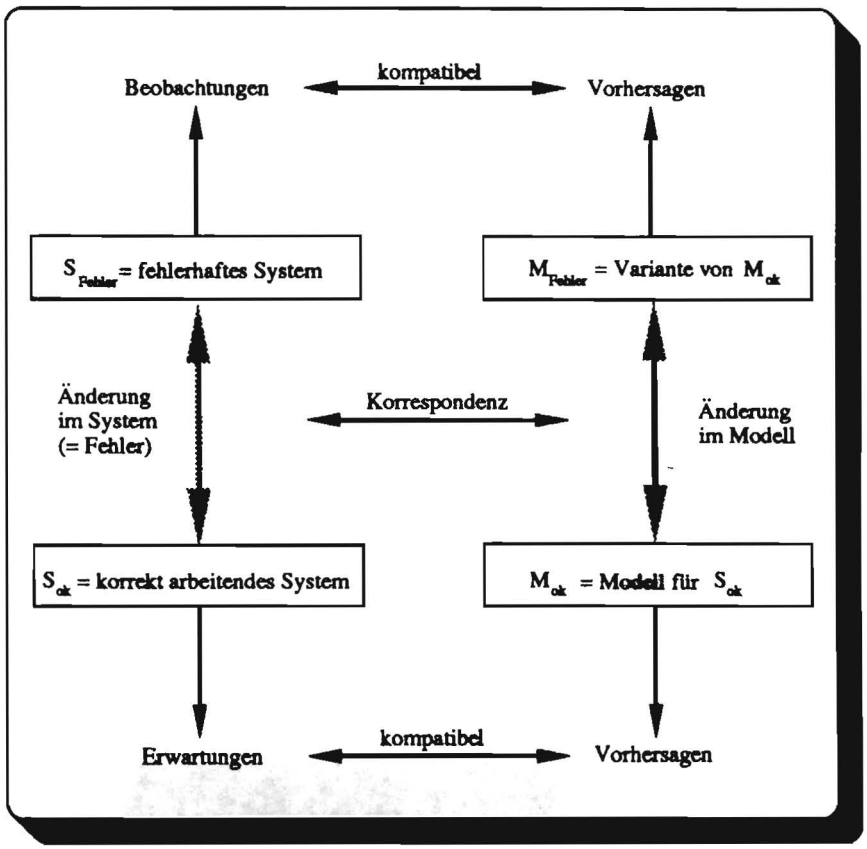

Abbildung 3.15: Die Idee hinter QUASIMODIS

dem vom Simulator prognostizierten Verhalten für das zugehörige Modell M_{ok}, so ist durch das Modell nicht das tatsächlich vorliegende Gerät repräsentiert. Geht man davon aus, daß das Modell korrekt ist, d. h. , daß M_{ok} das Verhalten des korrekten Systems S_{ok} richtig vorhersagt, dann funktioniert das technische System S nicht wie S_{ok}, sondern ist ein anderes Gerät S_{Fehler}. Bei der Fehlersuche wird nun so vorgegangen, daß ein verändertes Modell M_{Fehler} gesucht wird, für das das prognostizierte Verhalten den Beobachtungen am fehlerhaften System S_{Fehler} entspricht. Aus den Unterschieden zwischen M_{ok} und M_{Fehler} hofft man, auf die Fehlerquelle(n) im tatsächlichen Gerät Rückschlüsse ziehen zu können, indem man auf das Vorliegen von Korrespondenzen zwischen der Modelländerung und der Systemänderung baut.

Zur Realisierung dieser Vorgehensweise werden Hypothesen F_i in Form von Änderungen gegenüber der Struktur und/oder der Fuktionalität des Modells M_{ok} generiert. Aus dem ursprünglichen Modell M_{ok} und der Liste von Änderungen erhält man für jede Fehlerhypothese F_i ein $FehlermodellM_{F_i}$, für das jeweils ein Verhaltensgraph berechnet werden kann. Ein Fehlermodell M_{F_i}, dessen Verhaltensgraph das beobachtete Systemverhalten enthält, stellt ein potentielles Modell des vorliegenden Systems dar, ist also ein Kandidat für M_{Fehler}.

Beim Aufbau der Wissensbasis, d. h. bei der Modellerstellung werden zunächst Komponenten mit ihren Schnittstellen und Eigenschaften definiert, von denen dann Instanzen zu einem flachen Modell[10] zusammengesetzt werden. Dabei sind bei QUASIMODIS alle Komponenten primitiv, d. h. es gibt keine zusammengesetzten Komponenten. Lediglich das Gesamtmodell bildet eine übergeordnete Hierarchiestufe. In QUASIMODIS hat man sich gegen eine hierarchische Modellierung entschieden, da schon bei statischer Modellierung die Ermittlung des Verhaltens einer komplexen Komponente relativ aufwendig ist; im dynamischen Fall wären nicht nur Ein–Ausgabe–Relationen zu berechnen, sondern neue Verhaltensregeln samt zugehöriger Constraints automatisch zu generieren, ein selbst für triviale Fälle fast unlösbares Problem.

In QUASIMODIS erfolgt die Verhaltensbeschreibung des zu diagnostizierenden technischen Geräts in den Komponentendefinitionen, die wiederum in eine Bauteilebibliothek eingetragen werden. Zur Definition einer solchen Komponentenklasse gehören die Schnittstellen zur Außenwelt (*Ports*),deren Typen (*Wertebereich*) und die Regeln, nach denen sich das Bauteil verhält, also die Angabe, wie die Komponente über ihre Schnittstellen wirkt. Außerdem legen die einzelnen Bauteile den an ihren Schnittstellen anliegenden Werten bestimmte Constraints auf, die als *Komponenten–Constraints* (im folgenden: *K–Constraints*) bezeichnet werden. Diese *K–Constraints* sind unmittelbar den Komponenten zugeordnet und wirken ausschließlich auf die Schnittstellen einer einzelnen Komponente.

Hat man für alle benötigten Bauteile Komponentenklassen definiert, so werden die entsprechenden Instanzen für das Modell erzeugt. Dabei werden die *Ports* der Komponenteninstanzen an Variable gebunden, über die die Komponenteninstanzen aufeinander wirken, wobei überprüft werden muß, ob die Variable vom gleichen Typ wie der *Port* ist. Man erhält somit ein Netz, dessen Knoten aus Variablen und Komponenteninstanzen

[10]Das Gerät wird hier flach modelliert, da ein hierarchisches dynamisches Modell große Anforderungen an den Modellierer stellen würde: er müßte auf jeder Hierarchiestufe das Verhalten beschreiben und zusätzlich noch für die Verträglichkeit der Verhalten auf den verschiedenen Stufen garantieren.

bestehen. Dieses Netz ist außerdem so gestaltet, daß eine Variable jeweils nur mit Komponenteninstanzen verbunden ist und umgekehrt. Es repräsentiert die Modellstruktur, und eine Belegung aller in dem Modell enthaltenen Variablen stellt einen Zustand des modellierten Geräts dar. Ein Beispiel für ein solches Netz ist in Abbildung 3.16 zu finden.

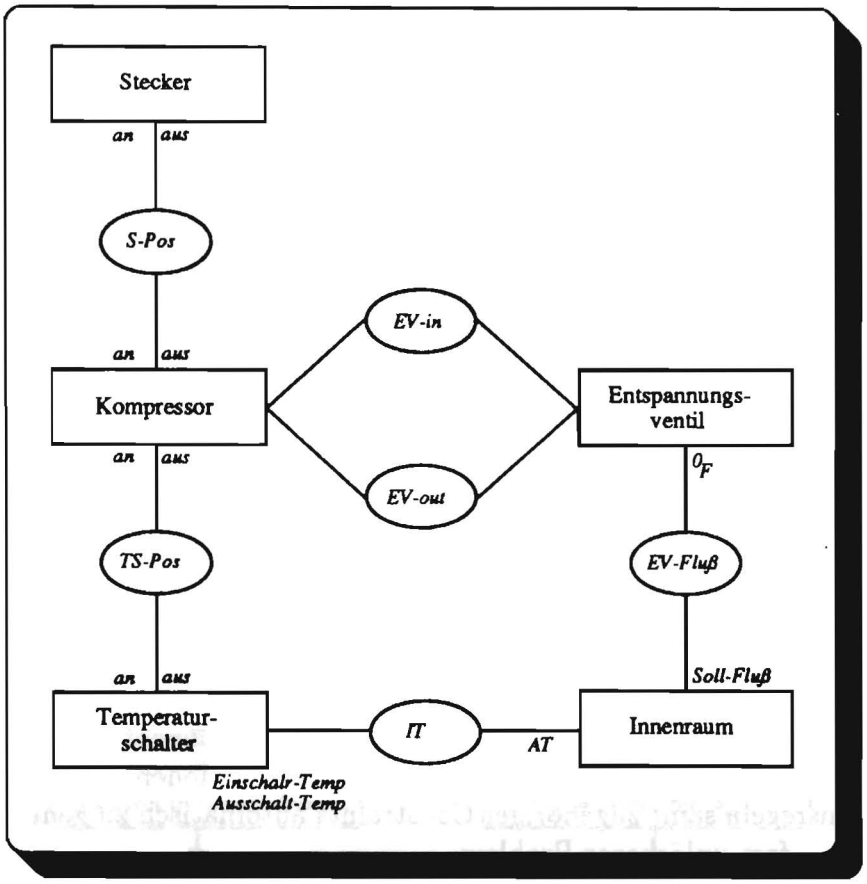

Abbildung 3.16: Beispielnetz

Um zu überprüfen, ob das Verhalten des Modells bereits aufgrund der Verhaltensregeln und *K–Constraints* eindeutig ist und dem Verhalten des modellierten Geräts entspricht, werden ein oder mehrere typische Startzustände ausgewählt, mit denen eine Simulation durchgeführt wird. Das Ergebnis einer Simulation ist ein gerichteter Verhaltensgraph, dessen Knoten Zustände und dessen Kanten Zustandsübergänge darstellen.

Im allgemeinen ist dieses Verhalten noch nicht eindeutig, d. h. ein Zustand hat mehrere Nachfolger. Dies liegt daran, daß mit qualitativen Werten gearbeitet wird und Relationen zwischen nicht direkt verbundenen Komponenten noch nicht repräsentiert sind. Solche Relationen werden in QUASIMODIS durch *Designer–Constraints* (im folgenden: *D–Constraints*) dargestellt. Im Gegensatz zu den *K–Constraints* können die *D–Constraints* jede beliebige Variable miteinander verknüpfen. Der Begriff *Designer–Constraint* kommt daher, daß man mit diesen Constraints Intensionen darstellen möchte, die der Entwickler bei der Konstruktion des zu modellierenden Geräts hatte, z. B. bei der Dimensionierung der verschiedenen Komponenten im Hinblick aufeinander. Über die *D-Constraints* werden

zusätzliche Verbindungen zwischen Variablen gegeben, die die Funktion des Gesamtmodells spezifizieren.

Der gerade geschilderte Sachverhalt wird mit folgendem Beispiel verdeutlicht:

Beispiel 3.2.1 Für ein gegebenes Rohr und eine gegebene Pumpe einer Anlage ist a priori nicht klar, daß das Rohr den von der Pumpe erzeugten Druck aushält. Der Designer einer Anlage wird aber natürlich auf die entsprechend abgestimmte Dimensionierung achten, so daß der ohne diese Zusatzinformation denkbare Fall des Rohrbruchs nicht auftreten kann. Ein derartiger Zusammenhang wird in QUASIMODIS durch *D–Constraints* repräsentiert.

Die Modellierung selbst ist dann beendet, wenn das Modell bei der Simulation für alle ausgewählten Startzustände nur noch das tatsächliche Verhalten liefert, d. h. wenn es einen Verhaltensgraphen liefert, in dem jeder Knoten nur noch genau einen Nachfolger hat. Fehlt die Eindeutigkeit, so ist die Aufgabe der Diagnoseerstellung nur unter erschwerten Bedingungen möglich, jedoch immer noch erfüllbar.

3.2.3 Ein Expertensystem zur wissensbasierten Diagnose für Flexible Fertigungssysteme

Ein Diagnose–Expertensystem, das modellbasiertes, pathophysiologisches Wissen zur Lösung eines Diagnoseproblems verwendet, ist das Expertensystem zur wissensbasierten Diagnose für Flexible Fertigungssysteme, das in Aachen von Prof. Dr. M. Weck und Gisela Kiratli entwickelt wurde [WK89, Kir]. Dieses Expertensystem wurde auf der LISP–Maschine EXPLORER mit der Expertensystem–Shell KEE 3.0 (Knowledge Engeneering Environment) entwickelt [kee86].

Zunächst wird in Abbildung 3.17 der Prinzipielle Aufbau dieses Diagnose–Expertensystems dargestellt.

Die Wissensbasis untergliedert sich in zwei Hauptbereiche:

- *Agenda*
- *gebietsspezifische Wissensbasis*

Die Agenda ist wie eine Art Arbeitsspeicher zu verstehen: in ihr werden alle Zwischenergebnisse des Diagnoseprozesses, die nicht verfolgten Ziele, die verworfenen Ziele und das aktuell verfolgte Ziel abgelegt. Die Agenda dient also als Zwischenspeicher für die Problemlösungskomponente.

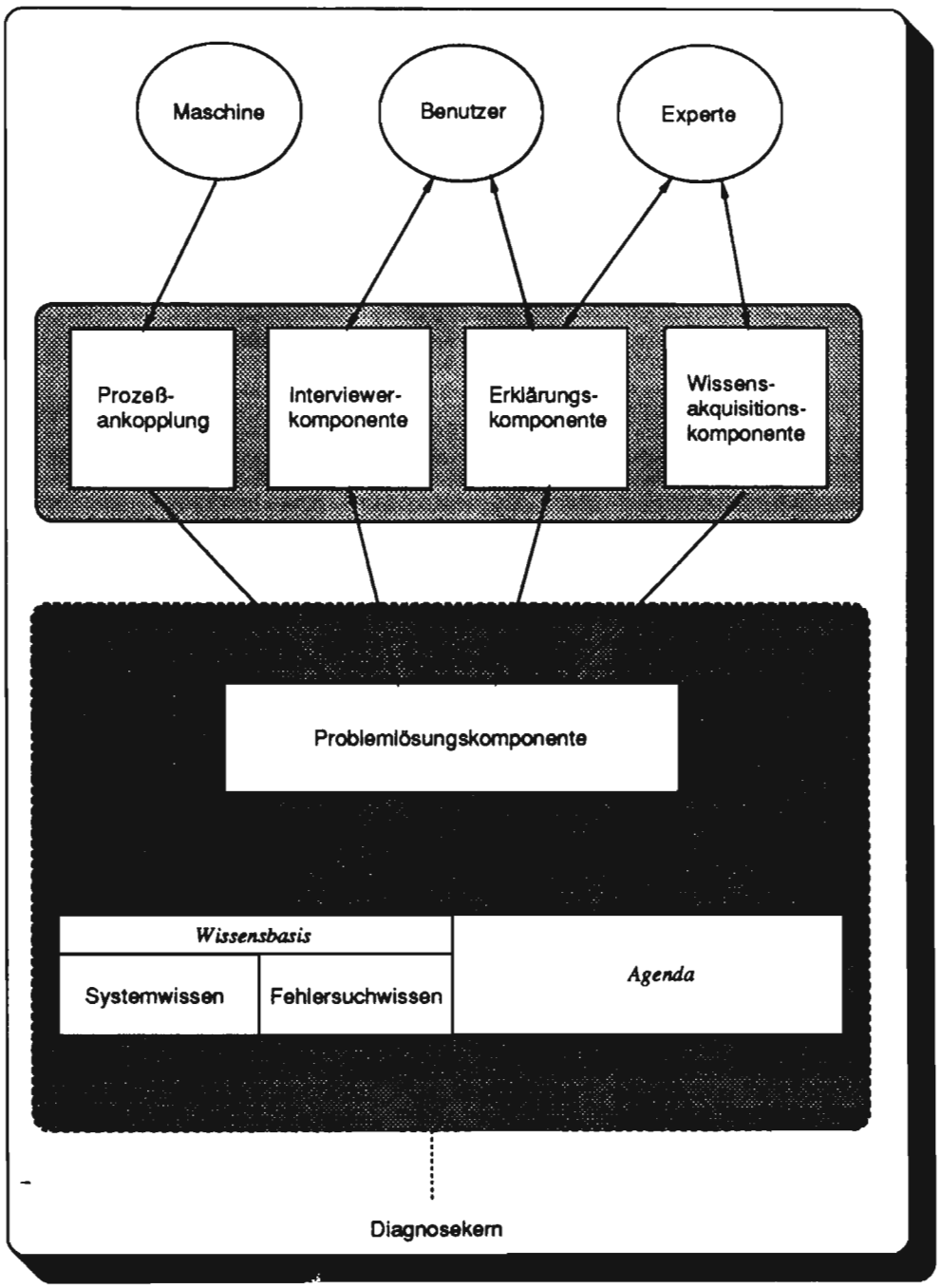

Abbildung 3.17: Allgemeiner Aufbau des Expertensystems zur wissensbasierten Diagnose für Flexible Fertigungssysteme

Das eigentliche gebietsspezifische Wissen wird in der Wissensbasis als kausales, pathophysiologisches Modell abgelegt. Diese kausale Modellierung wird durch eine Zweiteilung der gebietsspezifischen Wissensbasis realisiert, wie auch in Abbildung 3.18 zu sehen ist:

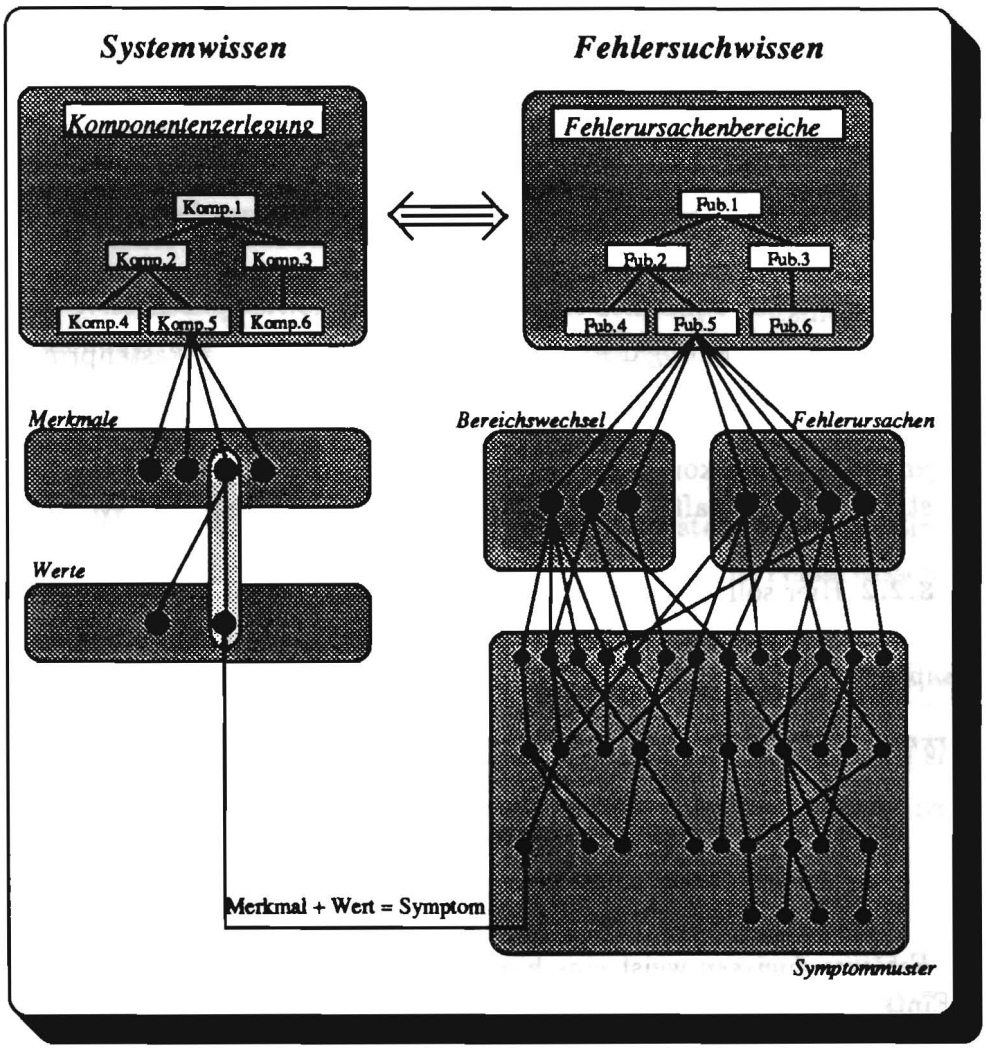

Abbildung 3.18: Aufteilung in System– und Fehlersuchwissen

- *Systemwissen*: dieses Wissen beinhaltet eine Beschreibung der Problemwelt mittels Komponentenzerlegung und entsprechend zugeordnete charakteristische Eigenschaften, sowie Fragentexte; detaillierter gliedert sich das Systemwissen folgendermaßen auf:

  i) Komponentenzerlegung

  ii) Merkmale

  iii) Werte,

wobei ii) und iii) die charakteristischen Eigenschaften der einzelnen Komponenten darstellen.

- *Fehlersuchwissen*: dieses Wissen enthält Fehlerbilder, eingeteilt nach Fehlerbereichen, Fehlerursachen und Symptomen; im Detail bedeutet dies folgende Einteilung:

  i) Fehlerursachenbereiche

  ii) Bereichswechsel/Fehlerursachen

  iii) Symptommuster.

Durch die Unterteilung in *Systemwissen* und *Fehlersuchwissen* erhält man eine weitgehende Entflechtung zwischen der Objektbeschreibung und der Problembeschreibung eines technischen Systems.

Dabei findet die Aufteilung des *Systemwissens* nach dem Baukastenprinzip statt: man macht eine hierarchische Einteilung des zu diagnostizierenden technischen Geräts in Komponenten und Unterkomponenten. Diese Unterteilung darf dabei beliebig weit gehen, wobei auch gemeinsame Teilkomponenten entstehen können. Jeder Komponente lassen sich charakteristische Eigenschaften zuordnen, die sich in Merkmale und Werte untergliedern.

Beispiel 3.2.2 Hier soll ein kleines Beispiel für ein Anlagenteil gezeigt werden:

- *Komponente*: Kabel
- *Merkmal*: Stromdurchfluß
- *Wert*: ausreichend *oder* unterbrochen

Auch das *Fehlersuchwissen* weist eine hierarchische Struktur auf, nur daß es sich diesmal um eine Einteilung in Fehlerursachenbereiche handelt.

Jeder Fehlerursache ist eine Reihe von Symptomen in Form eines Symptommusters zugeordnet, womit dokumentiert werden soll, welche Symptome zutreffen müssen, damit die Fehlerursache eindeutig bestimmt werden kann. Dabei gilt:

$$Symptom = Merkmal + Wert$$

Wenn man das gesamte zu einer Fehlerursache gehörende Symptommuster betrachtet, so ist es möglich, daß mehrere Fehlerursachenbereiche beteiligt sein können. Um diese komplexen Abhängigkeiten zu entflechten, wurden die Begriffe „*lokale Fehlerursachen*“ und „*Bereichswechsel*“ eingeführt.

Lokale Fehler beinhalten nur Symptome, die zu dem entsprechend zugeordneten Fehlerursachenbereich gehören und stellen somit das Ziel einer Diagnose dar. Dagegen führen Symptommuster von Bereichswechseln aus dem betrachteten Fehlerursachenbereich in einen anderen, bis schließlich ein lokaler Fehler festgestellt werden kann.

Wird eine Wissensbasis neu aufgebaut, so sollte mit der Eingabe des Systemwissens begonnen werden. Ist dies eingegeben (inklusive zugehörigen Merkmalen und Werten), so sollte das Fehlerwissen eingegeben werden.
Zur Darstellung der hierarchischen Wissensstruktur wurden Frames gewählt.

Als nächstes soll kurz auf drei der übrigen Komponenten des Systems eingegangen werden.

- *Wissensakquisitionskomponente*: folgende Möglichkeiten sind hier realisiert worden:
  - der Experte kann Änderungen und Erweiterungen an der Wissensbasis vornehmen;
  - die Wissensakquisitionskomponente stellt dem Experten die komplexen Zusammenhänge der Wissensbasis transparent und überschaubar dar, da der Experte über keine speziellen Programmierkenntnisse verfügen muß;
  - die Wissensakquisitionskomponente ist in gewissem Maße automatisiert, was bedeutet:
    * sie verfügt über einen automatischen Konsistenzcheck der Eingabe;
    * es werden getrennte Menüs für Systemwissen und Fehlerwissen angeboten;
- *Prozeßankopplung*: Über die Prozeßankopplung werden aktuelle Prozeßdaten empfangen, die für die interne Weiterverarbeitung aufbereitet werden. Das bedeutet, daß die erforderlichen Symptome nicht nur vom Anwender erfahrbar sind, sondern auch direkt von dem zu diagnostizierenden Gerät selbst. Prozeßdaten sind dabei:
  - *Fehlermeldungen*
  - *Störungsmeldungen*
  - *Statusmeldungen*

  Diese Prozeßdaten können von dem Expertensystem automatisch, ohne Wissen des Benutzers, erfaßt und eingebracht werden.

Zur Problemlösungskomponente ist zu sagen, daß es zwei Möglichkeiten gibt, zu einer Diagnose zu kommen:

- die Problemlösungskomponente sucht die Lösung absolut selbständig;
- der Benutzer gibt dem System eine verdächtige Fehlerursache vor, die es zu bestätigen oder zu verwerfen gilt.

3.3 Fallbasierte Diagnose–Expertensysteme

Ein relativ neues Forschungsgebiet stellen fallbasierte Diagnose–Expertensysteme dar. Fertige Systeme sind auf diesem Gebiet in der Literatur kaum zu finden, so daß in dieser Arbeit lediglich ein System beschrieben wird, nämlich PATDEX, da es das einzige ausführlich beschriebene fallbasierte Diagnose–Expertensystem war. Die meisten Anstrengungen gelten in diesem Gebiet dem maschinellen Lernen, das bedeutet, das Interesse liegt mehr darin, wie man Schlüsse aus der Falldatenbasis ziehen kann (ein Stichwort ist hier analogiebasiertes Lernen) [PS89].

3.3.1 PATDEX

In diesem Kapitel möchten wir uns mit einem System beschäftigen, das ausschließlich mit Wissen arbeitet, das in Form von Fällen und Erfahrungen vorliegt. Es handelt sich hierbei um das System PATDEX (PATtern Directed EXpert system) und seine Weiterentwicklung PATDEX/2, das u. a. von Stefan Weß in der Arbeitsgruppe von Prof. Dr. M. M. Richter an der Universität Kaiserslautern entwickelt wurde [Weß90]. PATDEX selbst ist ein eigenständiger Prototyp, der alleinstehend zur Diagnosefindung verwendet werden kann, während PATDEX/2 eine Ankopplung oder sogar Erweiterung des MOLTKE-Systems darstellt [Ric91], auf das wir im Verlauf dieser Arbeit noch eingehen werden. Aber auch PATDEX/2 soll als alleinstehendes Expertensystem realisierbar sein.

Der prinzipielle Aufbau von PATDEX ist in Abbildung 3.19 dargestellt.

Die Wissensbasis von PATDEX ist demnach in zwei Bereiche unterteilt:

- *Falldatenbasis*
- *Erfahrungsgraph*

Der Problemlösungsprozeß des Systems wird durch den Benutzer gestartet, indem er beobachtete Symptome eingibt, die der aktuellen Problemsituation entsprechen. PATDEX versucht dann, mittels eines Ähnlichkeitsmaßes, den ähnlichsten Fall samt zugehöriger Lösung in der Fallbasis aufzufinden, wobei vom Benutzer gegebenenfalls weitere Symptome und Symptomwerte erfragt werden. Ebenfalls mit in den Problemlösungsprozeß einbezogen wird der Erfahrungsgraph, der in Form eines assoziativen Netzwerkes vorliegt, durch das Zusammenhänge zwischen Symptommustern, Fällen und partiellen Determinationen zwischen Symptomen repräsentiert werden. Der Erfahrungsgraph wird nach jeder Diagnosesitzung durch das System ergänzt, was bedeutet, daß PATDEX in der Lage ist, dieses Wissen aus bereits durchgeführten Sitzungen zu erlernen.

Die Falldatenbasis wird bei Erstellung eines Expertensystems von einem Experten mit einer hinreichend großen Anzahl von bereits diagnostizierten Fällen gefüllt. Außerdem hat der Experte die Möglichkeit, das System in gewisser Weise zu „trainieren“, d. h. er kann am Aufbau des Erfahrungsgraphen mitwirken.

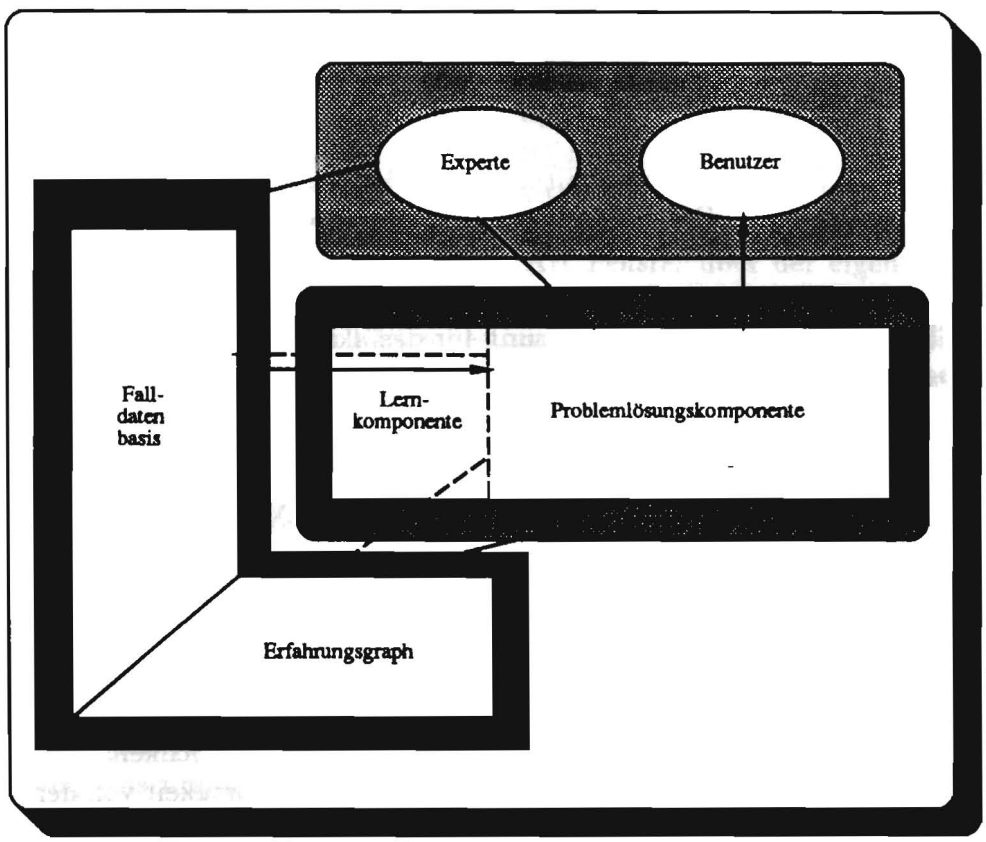

Abbildung 3.19: Allgemeiner Aufbau von PATDEX

Zum einen werden PATDEX also von vornherein eine gewisse Anzahl von Fällen mit auf den Weg gegeben, zum anderen werden in der Falldatenbasis aber auch alle von dem System selber bearbeiteten und gelösten Fälle abgelegt. Dabei können sowohl der Benutzer, als auch der Experte Korrekturen vornehmen, falls eine Diagnose nicht gestimmt hat. Die Diagnosefälle stellen nun den zentralen Repräsentationsformalismus von PATDEX dar. Ein Fall C_k wird im Kontext des PATDEX-Systems als ein 4–Tupel mit $C_k = (Sit_k, \varepsilon_k, \delta_k, D_j)$ verstanden, wobei gilt:

Sit_k ist die Beschreibung einer zu diagnostizierenden Situation

ε_k ist die Eingangsevidenz, mit $0 \leq \varepsilon_k < 1$

δ_k ist die Diagnoseevidenz, mit $0 < \delta_k \leq 1$

D_j ist die zum Fall C_k gehörige Diagnose

Dabei wird durch Sit_k die konkrete Situation, in der ein Servicetechniker die Diagnose D_j gestellt hat, beschrieben. Die Parameter ε und δ haben für den Diagnoseprozeß folgende Bedeutung:

- Ein Fall C_k wird als mögliche Lösung für das aktuelle Diagnoseproblem in Betracht gezogen, falls $\text{sim}(Sit, Sit_k) \geq \varepsilon$ gilt, wobei Sit die Situationsbeschreibung des gerade zu bearbeitenden Falles ist und Sit_k die aus der Fallbasis zum Vergleich herangezogene Situationsbeschreibung des Falles C_k.
- Die im Fall C_k vermerkte Diagnose wird von PATDEX gestellt, falls $\text{sim}(Sit, Sit_k) \geq \delta$ gilt.

Dabei kann der Schwellwert ε_k als ein Wert verstanden werden, ab dem ein Fall C_k als zu einer gerade zu diagnostizierenden Situation hinreichend ähnlich betrachtet wird, so daß die Erhebung weiterer Symptome des Falles C_k sinnvoll ist. Der Schwellwert δ_k kann als Maß für die bei einer Diagnose D_j minimal geforderte Ähnlichkeit zwischen Fall und Situation interpretiert werden. Es bietet sich an, δ_k in Abhängigkeit von der Diagnose D_j zu bestimmen. Wird eine Diagnose D_j irrtümlich gestellt und verursacht dabei geringe Folgekosten, so kann δ_k relativ niedrig angesetzt werden. Fallen durch die Diagnose D_j jedoch hohe Folgekosten an, so ist δ_k entsprechend hoch zu wählen.

Beispiel 3.3.1 Mit den folgenden beiden Beispielen soll der eben ausgeführte Sachverhalt näher erläutert werden.

i) Eine Diagnose „Sicherung defekt“, sollte sie nicht richtig sein, verursacht niedrige Folgekosten, da sie im Normalfall leicht zu überprüfen ist. In diesem Fall kann δ_k sehr niedrig angesetzt werden.

ii) Eine Diagnose „Antriebslager defekt“, sollte sie nicht richtig sein, kann dagegen sehr hohe Folgekosten verursachen, da ihre Überprüfung sich sehr viel aufwendiger gestalten kann. In diesem Fall sollte δ_k sehr hoch angesetzt werden.

Bei Eintreten des Extremfalls δ_k=1 wird eine exakte Übereinstimmung der zu diagnostizierenden Situation und der Situation des zum Vergleich herangezogenen Falles gefordert.

Die Falldatenbasis von PATDEX wird bei einem neu zu bearbeitenden Fall partitioniert. Dabei werden die abgespeicherten Fälle in fünf verschiedene Klassen eingeteilt.

Sei $\phi=sim(Sit,Sit_k)$ mit $-2\leq\phi\leq1$ definiert als die Ähnlichkeit der Situationsbeschreibung des Falles C_k mit der aktuellen Situation Sit, ε_k die Eingangsevidenz und δ_k die im Fall C_k definierte Diagnoseschwelle mit $\varepsilon_k<\delta_k\leq1$. Es entstehen nun folgende Klassen von Fällen:

Bewiesen	falls	$\phi=1$,
Ausreichend	falls	$\delta\leq\phi<1$,
Wahrscheinlich	falls	$\varepsilon\leq\phi<\delta$,
Möglich	falls	$0\leq\phi<\varepsilon$,
Unwahrscheinlich	falls	$\phi<0$.

Diese Art der Klasseneinteilung definiert eine Art *Fenster* über der eigentlichen Falldatenbasis. Der Diagnoseprozeß, also der Kontrollfluß des PATDEX–Systems, wird durch die aktuell betrachtete Klasse von Fällen bestimmt. Damit jedoch keine mögliche Lösung übersehen werden kann, wird das Ähnlichkeitsmaß bei der Erhebung eines Symptoms für jeden Fall, in dem das entsprechende Symptom erfüllt ist, neu berechnet.

Am Ende dieses Abschnittes wird noch kurz auf die Weiterentwicklung von PATDEX, nämlich PATDEX/2 eingegangen. Während, wie bereits erwähnt, der PATDEX-Prototyp als „stand–alone"–System konzipiert wurde, das zur Inferenz nur das Wissen verwenden kann, das in Form von Diagnosefällen vorliegt, ist das PATDEX/2–System in die MOLTKE 3–Werkbank integriert. Daher ist es naheliegend, das technische Hintergrundwissen, das in MOLTKE 3 repräsentiert ist, für PATDEX/2 nutzbar zu machen.

Eine Möglichkeit dafür, dieses Hintergrundwissen sinnvoll zu nutzen, ist die Expansion einer Situationsbeschreibung eines Falles, durch die man den Aufwand einer Diagnoseerstellung häufig einschränken kann. Liegt beispielsweise das Symptom „Licht = brennt" vor, so kann die Situationsbeschreibung des gerade zu bearbeitenden Falles aufgrund einer im Hintergrundwissen abgelegten Regel „Wenn das Licht brennt, so ist die Netzspannung vorhanden" durch das Symptom „Netzspannung = vorhanden" expandiert werden, d. h. einige Symptome können aus dem Vorliegen anderer Symptome geschlossen werden, ohne Extraaufwand für den mit dem System arbeitenden Techniker.

Eine weitere Änderung des PATDEX/2–Systems ist, daß die Fallbasis beim Bearbeiten eines Falles noch detaillierter partitioniert wird.

Die jedoch signifikanteste Änderung im Vergleich zum PATDEX–Prototyp ist, daß der Erfahrungsgraph wegfällt. Beim PATDEX–Prototyp werden gewisse Informationen getrennt, in Form dieses Graphen repräsentiert. Bei PATDEX/2 werden diese Informationen mit den Fällen zusammen in der Falldatenbasis abgelegt, liegen also nicht mehr so explizit vor. Eine weitere wichtige Änderung ist, daß die Symptome in PATDEX/2 nicht mehr gleichgewichtet sein müssen. Das Lernverfahren ist insgesamt zwar verändert worden, aber die eigentliche Fähigkeit des Lernens bleibt dem System erhalten.

3.4 Statistische Diagnose–Expertensysteme

In diesem Abschnitt der Arbeit wird nun das statistische Diagnose–Expertensystem PROSPECTOR vorgestellt. Auch in diesem Bereich ist es sehr schwer, existierende Expertensysteme zu finden. Dies merkt man auch daran, daß hier kein technisches System vorgestellt wird, sondern stellvertretend ein System aus dem Anwendungsgebiet Geologie, an dem prinzipielle Erkenntnisse jedoch ebenfalls gut zu vermitteln sind. Eine Erklärung für den Mangel statistischer Diagnose–Expertensysteme ist bereits vorher schon einmal geliefert worden, nämlich die, daß man zur Erstellung repräsentativer Statistiken hinreichend große Falldatenbasen benötigt, die anscheinend noch nicht vorhanden sind.

3.4.1 PROSPECTOR

Ein Expertensystem, bei dem das Wissen in statistischer Form modelliert wurde, ist PROSPECTOR. Ursprünglich wurde PROSPECTOR entwickelt, um bei der Erforschung von Bodenschätzen zu helfen, besonders bei der Beurteilung regionaler Bodenschätze [DR83]. Später wurde es auch in Gebieten wie der Beurteilung von Schürfstellen, der Auswahl von Bohrstellen, oder der numerischen Simulation eingesetzt.

In dieser Arbeit wird das System vorgestellt, mit dessen Hilfe Schürfstellen bewertet werden sollen.

Der allgemeine Aufbau von PROSPECTOR ist in Abbildung 3.20 graphisch dargestellt.

Den typischen Benutzer von PROSPECTOR – wenn es bei der Bewertung von Schürfstellen eingesetzt wird – stellt ein Geologe dar, der gerade mehrere Tage damit verbracht hat, eine Oberflächenuntersuchung an einer Schürfstelle durchzuführen, dabei einige interessante Informationen erhalten hat, und nun gerne einen Spezialisten zu Rate ziehen möchte bei der Beurteilung, inwieweit die Schürfstelle einem klassischen Modell für Erzablagerungen entspricht.

Die Benutzerschnittstelle übernimmt die Dialogführung mit dem Geologen und stellt ihm zunächst Fragen über Gesteins– und Mineraltypen, die an der Schürfstelle vorgefunden wurden, oder dort vermutet werden. Die meisten Fragen bei diesem Dialog können mit Ja oder Nein beantwortet werden. Im Verlauf des Dialogs versucht PROSPECTOR dann, ein bestimmtes verdächtiges Modell zu bestätigen.

Auch eine Erklärungskomponente steht zur Verfügung, die dem Benutzer Fragen zur Vorgehensweise des Systems[11] beantworten kann.

Eine spezielle Wissensakquisitionskomponente existiert in PROSPECTOR nicht. Das Wissen, also die verschiedenen Modelle für Erzablagerungen, wird nach statistischen Erhebungen von einem Wissensingenieur oder einem Experten direkt in Form von Inferenz-

[11] Ein Beispiel für solche Fragen sind Warum–Fragen. Sie werden vom Benutzer gestellt, wenn er wissen möchte, warum er von dem System nach einer bestimmten Angabe gefragt wird, aus deren Beantwortung für ihn nicht zu erkennen ist, welchen Zweck sie hat.

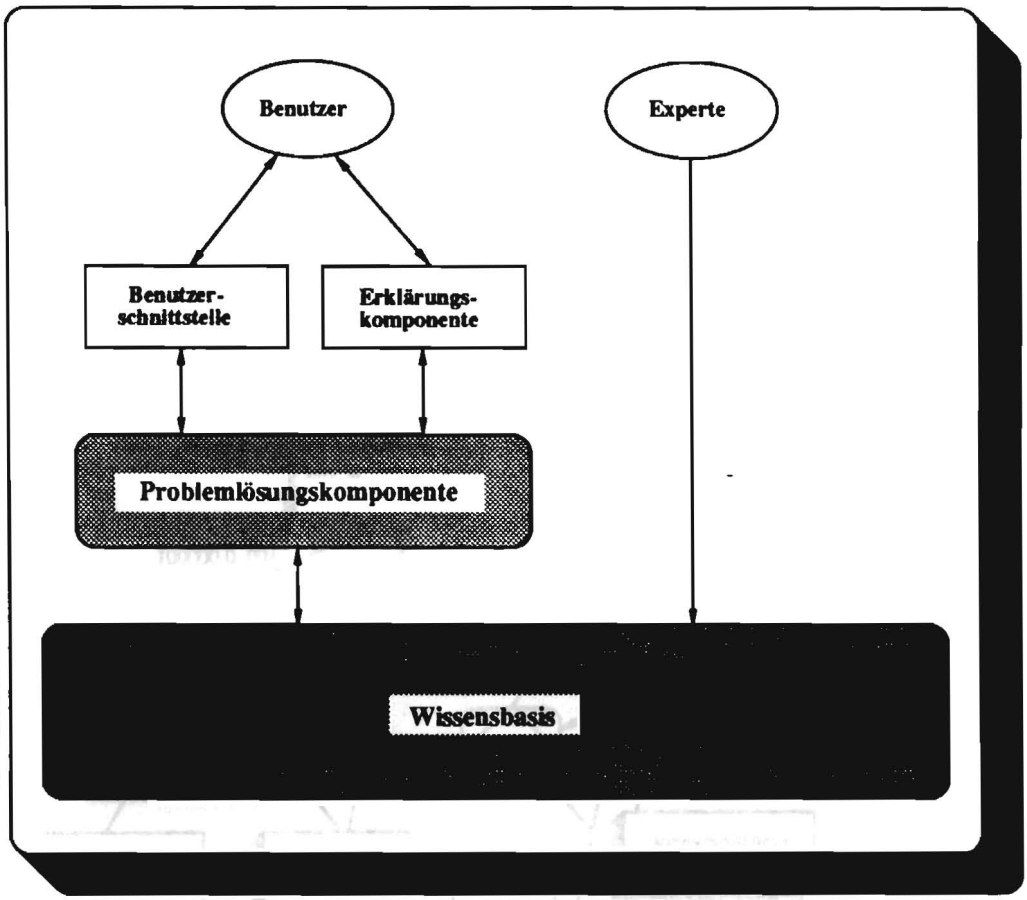

Abbildung 3.20: Allgemeiner Aufbau von PROSPECTOR

netzwerken in die Wissensbasis eingegeben.

Für die Bewertung von Schürfstellen wurden von Geologen Modelle für Erzablagerungen erstellt. Diese Modelle werden in Inferenznetzwerke kodiert und in die Wissensbasis eingegeben. Ein Ausschnitt aus einem solchen Inferenznetzwerk eines Modells für Erzablagerungen ist in Abbildung 3.21 graphisch dargestellt.

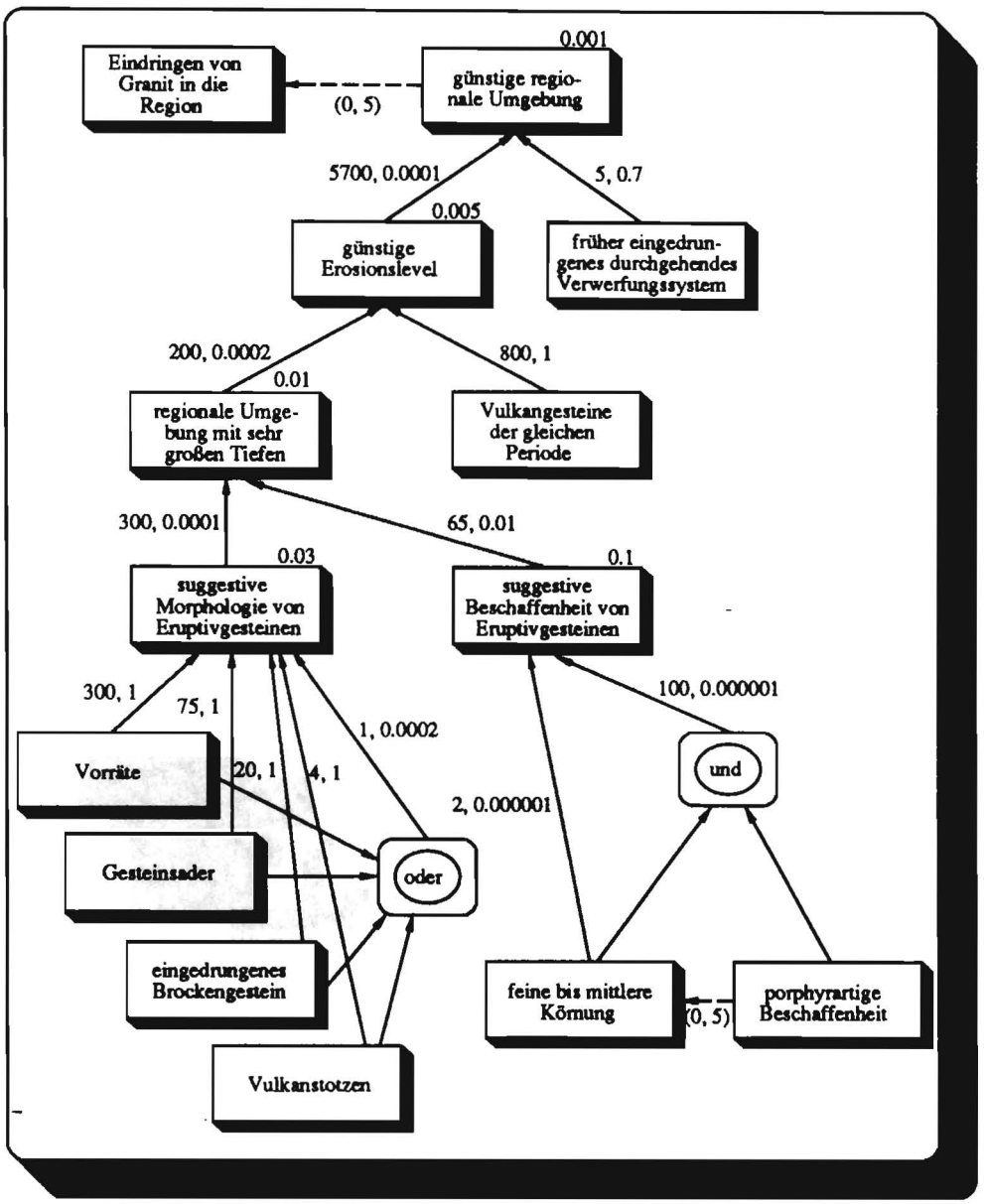

Abbildung 3.21: Ausschnitt eines Inferenznetzwerkes eines Modells für Erzablagerungen

Wie man dem Bild entnehmen kann, ist das Inferenznetzwerk wie ein Baum aufgebaut. Die Söhne eines Knotens stellen die Bedingungen für die Gültigkeit des Vaterknotens dar. Den inneren Knoten des Baums sind nun Werte zugeordnet. Diese Werte stellen die

a priori Wahrscheinlichkeiten für die Gültigkeit eines Knoten dar. An den Kanten des Baums stehen zwei Werte: λ und $\bar{\lambda}$.

λ beinhaltet die Chance, mit der die a priori Wahrscheinlichkeit des Vaterknotens multipliziert wird, wenn die in dem Sohnknoten enthaltene Bedingung erfüllt ist. $\bar{\lambda}$ beinhaltet die Chance, mit der die a priori Wahrscheinlichkeit multipliziert wird, wenn die in dem Sohnknoten enthaltene Bedingung nicht zutrifft.

In dem graphisch abgebildeten Inferenznetzwerk in Abbildung 3.21 bedeutet dies beispielsweise, daß das Vorhandensein von feiner bis mittlerer Körnung die Wahrscheinlichkeit für das Vorliegen suggestiver Beschaffenheit von Eruptivgesteinen verdoppelt (0.1x2), während das Nichtvorhandensein von feiner bis mittlerer Körnung dazu führt, daß die Wahrscheinlichkeit für das Vorliegen suggestiver Beschaffenheit von Eruptivgesteinen so gut wie Null ist (0.1x0.000001).

Nun fehlt noch die Erklärung für die gestrichelten Pfeile in Abbildung 3.21. Knoten, von denen ein solcher Pfeil ausgeht, stellen Kontexte dar. Damit nun für den Vaterknoten eines solchen Kontext–Knotens weitere Evidenz gesammelt werden kann, muß der Kontext mit der innerhalb des an der Kante angegebenen Intervalls liegenden Sicherheit etabliert werden.

Wie nun die einzelnen Werte genau verrechnet werden, wird in dieser Arbeit nicht weiter ausgeführt, da diese Aufgabe von der Problemlösungskomponente erfüllt wird.. Die Verrechnungsschemata von PROSPECTOR sind in [Bec92] zu finden.

3.5 Diagnose–Expertensysteme, die Wissen in Form von Entscheidungsbäumen verwenden

Im letzten Teil dieses Kapitels soll nun ein Diagnose–Expertensystem beschrieben werden, das Wissen in Form eines Entscheidungsbaumes verwendet. Es handelt sich um DIWA, und gehört zu den wenigen Expertensystemen die auf diesem Gebiet existieren.

3.5.1 DIWA

Ein System, bei dem das Wissen in Form von Flußdiagrammen modelliert wird, ist DIWA (DIagnosewerkzeug mit graphischer WissensAkquisition), das unter Leitung von Gabriele Schmiedel in der zentralen Forschungs– und Entwicklungsabteilung von SIEMENS entwickelt wurde [Sch91].

Mit DIWA wurde unter anderem ein Werkzeug entwickelt, das eine automatisierte Entwicklung eines wissensbasierten Systems ermöglichen soll, und den traditionellen Entwicklungsweg, nämlich die Wissensakquisition durch einen Wissensingenieur, drastisch verkürzen soll. Bei DIWA ist es möglich, daß der Experte sein Wissen direkt über eine graphische Wissensakquisitionskomponente in die Wissensbasis eines Expertensystems eingibt.

Der prinzipielle Aufbau von DIWA ist in Abbidung 3.22 dargestellt. Dieser Abbildung ist zu entnehmen, daß DIWA aus dem DIWA–Entwicklungssystem und aus dem DIWA–Ablaufsystem besteht. Das Entwicklungssystem besteht aus den Moduln:

- *Graphische Wissensakquisitionskomponente:* Realisiert die graphische Benutzeroberfläche für den Experten zum Erstellen der graphischen Wissensstrukturen. Hier werden einfache Überprüfungen vorgenommen (unerlaubte Verknüpfungen von Basis–Symbolen,etc.) und deren Verletzungen dem Benutzer sofort angezeigt.

- *Wissensbasisgenerierungskomponente:* Erzeugt aus den vom Experten eingegebenen graphischen Wissensstrukturen die ablauffähige Wissensbasis. Hierbei werden gewisse Konsistenzüberprüfungen vorgenommen. Sind Fehler vorhanden, so werden in einem Übersetzungsprotokoll die entsprechenden Meldungen festgehalten.

- *Testumgebung:* Entspricht der Benutzeroberfläche des Ablaufsystems mit zusätzlicher Trace–Funktionalität und einem Übersichtsgraphen zum einfachen Testen von erzeugten Wissensbasen.

- *Diagnosemodell der „defekten Maschine“* und *Strategie der Modellabarbeitung:* Realisiert zum einen die statische Wissensstruktur in Form der Basis–Einheiten (Problemknoten, Reparaturanleitungen, Indizien,...) und deren mögliche Verknüpfungen und zum anderen den dynamischen Teil des Wissens (Problemursachensuche, Rücksprünge,etc.), also die Abarbeitung der statischen Wissensstrukturen, die mit Hilfe des Modells der defekten Maschine beschrieben sind.

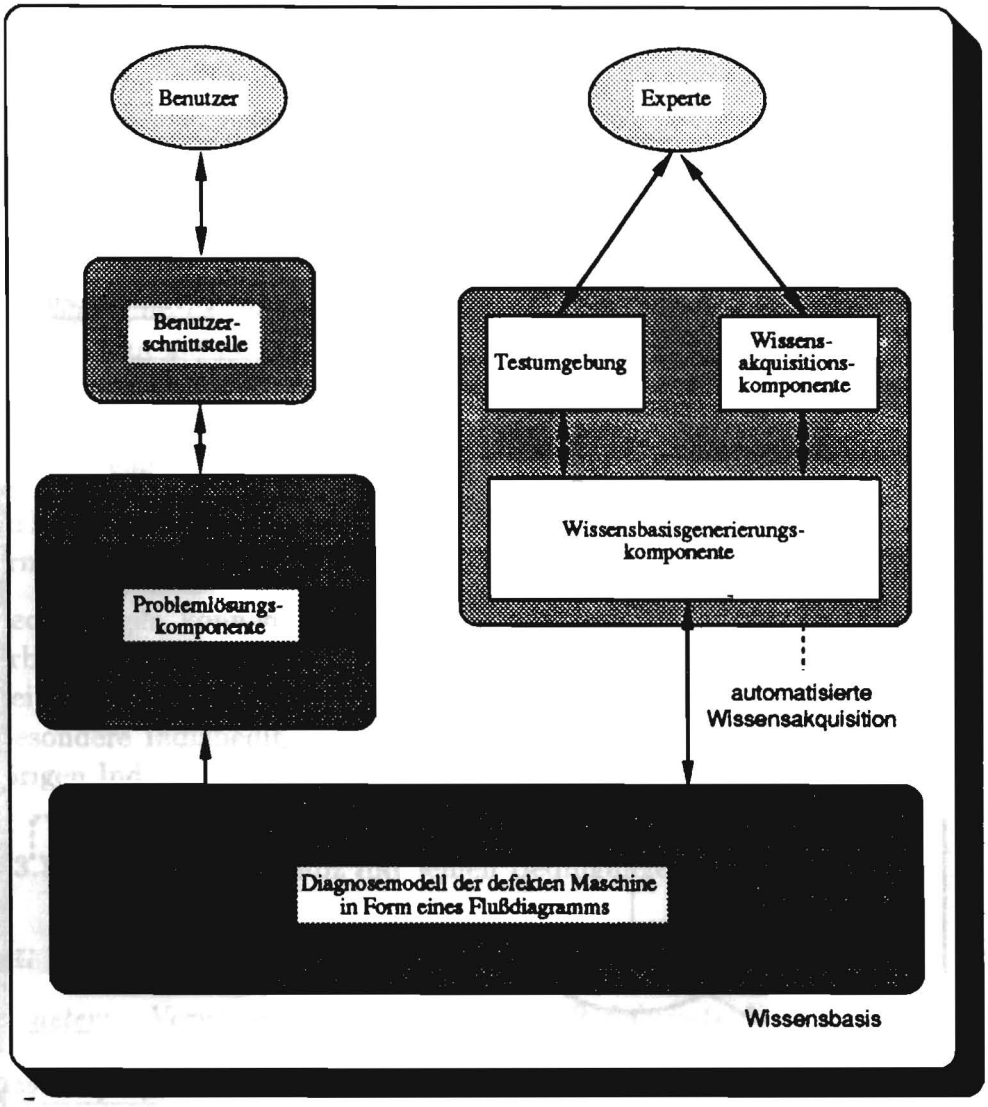

Abbildung 3.22: Allgemeiner Aufbau von DIWA

Das Ablaufsystem besteht aus den Moduln

- *Benutzeroberfläche des Ablaufsystems:* Realisiert die Dialoge mit dem Benutzer während einer Sitzung. Es werden Parameterwerte erfragt, Erläuterungen und Reparaturanleitungen ausgegeben. Der Benutzer kann sich weitere Informationen über die Sitzung anzeigen lassen.
- *Diagnosemodell der „defekten Maschine"* und *Strategie der Modellabarbeitung:* siehe oben.

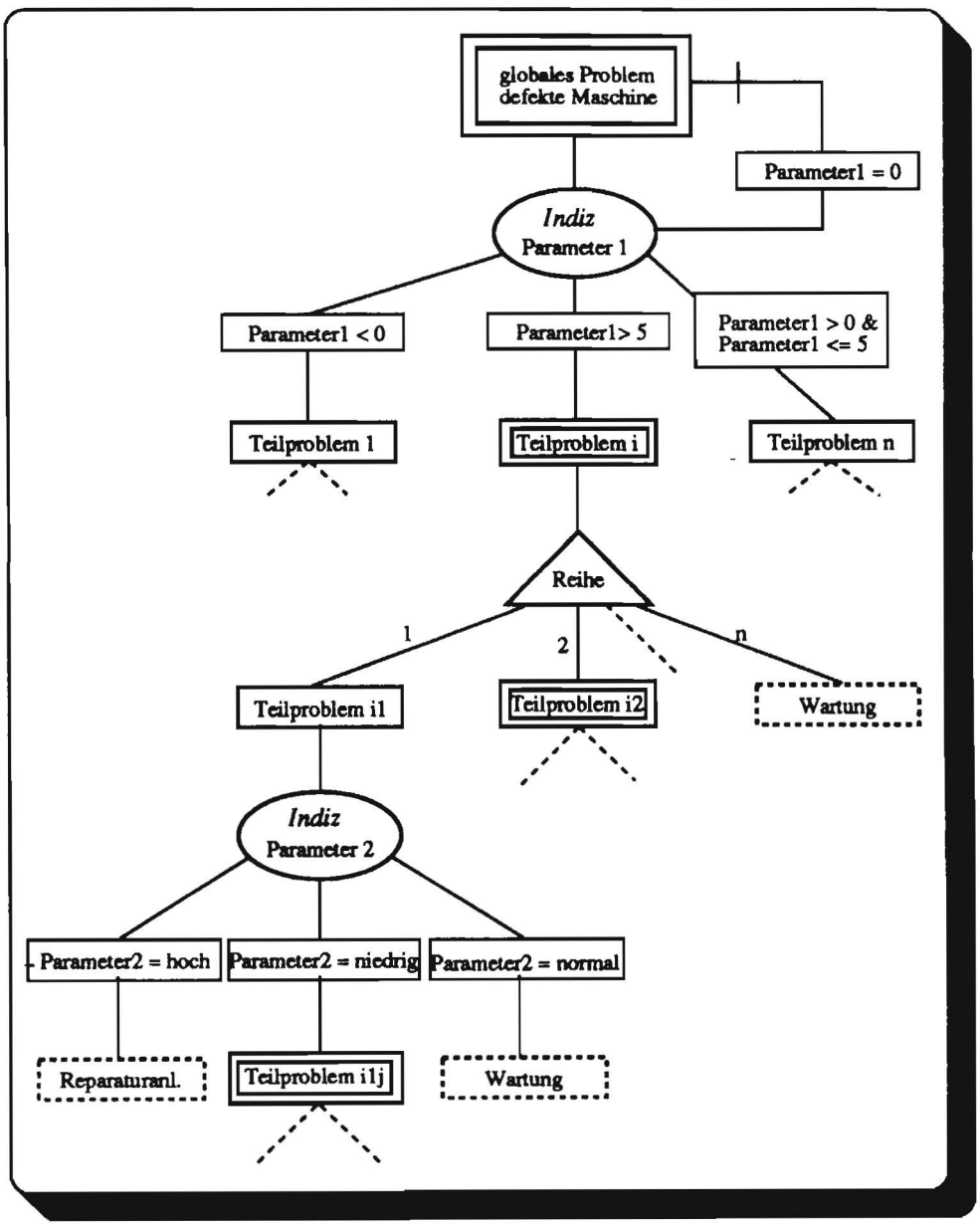

Abbildung 3.23: Diagnosemodell der „defekten Maschine"

In der Wissensbasis von DIWA sind das Modell der defekten Maschine und die Abarbeitung dieses Modells miteinander verzahnt abgelegt.

Das in Abbildung 3.23 dargestellte *Diagnosemodell der defekten Maschine* beschreibt die Zerlegung des globalen Problems „*defekte Maschine*" in Teilprobleme. Diese Zerlegung ist hierarchisch und endet bei den Fehlerursachen, die durch die zugehörigen *Reparaturanleitungen* repräsentiert werden, oder bei den sogenannten *Wartungsausgängen*, mit denen Situationen bezeichnet werden, in denen die Rufbereitschaft oder der Experte um Hilfestellung gebeten werden, da das Expertensystemwissen nicht mehr ausreicht, um eine Diagnose zu erstellen.

Das somit erhaltene Modell läßt sich graphisch als Baum darstellen, bei dem das globale Problem *defekte Maschine* die *Wurzel*, die Teilprobleme die *Knoten* und die Reparaturanleitungen oder Wartungsausgänge die *Blätter* darstellen.

Durch die Kanten werden in diesem Baum die Übergänge von einem Teilproblem zum nächsten dargestellt. Sie sind durch zwei Beschriftungstypen beschriftet, durch *Indizien* und *Reihenfolgen*, um die Fehlerlokalisierung durch möglichst frühzeitige Einschränkung der Suche im Baum auf einen Teilbaum zu beschleunigen. Dabei besteht ein *Indiz* aus einer Menge von Konstanten und Parametern und aus mehreren *Indizbedingungen*, die jeweils einem Nachfolger zugeordnet sind und in denen wiederum nur Parameter und Konstanten verwendet werden, die im Indiz angegeben sind.

Für die Indizbedingungen gibt es eine Sprache, die die Verknüpfungsmöglichkeiten von Parametern und Konstanten zu einer Indizbedingung beschreibt.

Indizbedingungen können aus komplexen Klauseln bestehen, die durch „*und*" und „*oder*" verbunden sind. Die zu einem Indiz gehörigen Indizbedingungen müssen nicht disjunkt sein.

Eine besondere Indizbedingung ist *SONST*, mit der das Komplement aller zu einem Indiz gehörigen Indizbedingungen bezeichnet wird.

Beispiel 3.5.1 Hier soll ein Indiz mit seinen Bedingungen gezeigt werden:

Indiz: Indiz–1

Parameter: Vorwärtsleistung, Solleistung, Reflektierte–Leistung, Plasma

Indizbedingungen:

Indizbedingung–1:
BETRAG(Vorwärtsleistung – Solleistung) $\leq$ 10%(Solleistung) &
Reflektierte – Leistung $\geq$ 0 &
Reflektierte – Leistung $\leq$ 3%(Solleistung) &
Plasma = vorhanden

Indizbedingung–2:
Vorwärtsleistung = 0 &
Solleistung > 0 &
Reflektierte - Leistung = 0 &
Plasma = nicht - vorhanden

Indizbedingung–3:
SONST

Gibt es kein geeignetes Indiz, mit dem die Vorgehensweise festgelegt werden kann, so wird eine *Reihenfolge* genommen, die für die Teilprobleme eines Problems eine Abarbeitungsreihenfolge vorgibt, die vom Experten während der Entwicklungsphase aufgrund seines Erfahrungsschatzes (z. B. Reparaturkosten, Auftrittshäufigkeit, Überprüfungsaufwand) festgelegt wird. Außerdem kann jeder Problemknoten, wenn er betrachtet wird, eine *Erklärung* liefern[12].

Nun gibt es auch noch einen besonderen Problemknoten, den sogenannten *Rücksprungknoten.* Dieser ist dadurch gekennzeichnet, daß er an der nachfolgenden Kante eine Indizbeschriftung hat, zu der es eine Indizbedingung gibt, die das Nichtvorhandensein des zum Rücksprungknoten gehörenden Problems charakterisiert.

Um in dem Diagnosemodell von der Größe her handhabbare Strukturen zu erhalten, wird die gesamte Wissensbasis von DIWA automatisch modularisiert, d. h. die abgeschlossene Wissensstruktur–Wissensbasis wird in Wissensmodule zerlegt, die der hierarchischen Modularisierung des komplexen Modells dienen. So erhält man sich eine gewisse Übersicht, die bei komplexen technischen Geräten, die diagnostiziert werden sollen, von Nöten ist.

In DIWA wird diese automatische Wissensmodulgenerierung bei der Wissensbasiserstellung nach folgendem Prinzip vorgenommen:

Jeder Rücksprungknoten erzeugt ein neues Wissensmodul, dessen oberster Knoten (Startknoten) er selbst wird. Im darüberliegenden Wissensmodul wird dann anstelle des Rücksprungknotens ein Stellvertreter eingehängt, der Verwaltungsinformationen und den Verweis auf das durch ihn vertretene Wissensmodul enthält.

Neben dieser automatischen Wissensmodulgenerierung, kann auch der Experte selbst weitere Wissensmodule definieren, die einen beliebigen Problemknoten als Startknoten besitzen.

Der Sachverhalt der Modularisierung ist in Abbildung 3.24 graphisch dargestellt.

DIWA stellt, wie bereits zu Beginn erwähnt wurde, ein Expertensystem dar, bei dem bei der Abarbeitung der Wissensbasis jeder Schritt von vornherein genau vorgegeben ist, also beispielsweise mit der Abarbeitung eines MODULA–Programms verglichen werden könnte. Auf die Einsetzbarkeit eines solchen Expertensystems wird in Kapitel 4 näher eingegangen.

[12]Eine solche Erklärung kann zum Beispiel so aussehen, daß in ihr vor einem nachfolgenden Indiz eine Anleitung zum Ausbau bestimmter Teile enthalten ist.

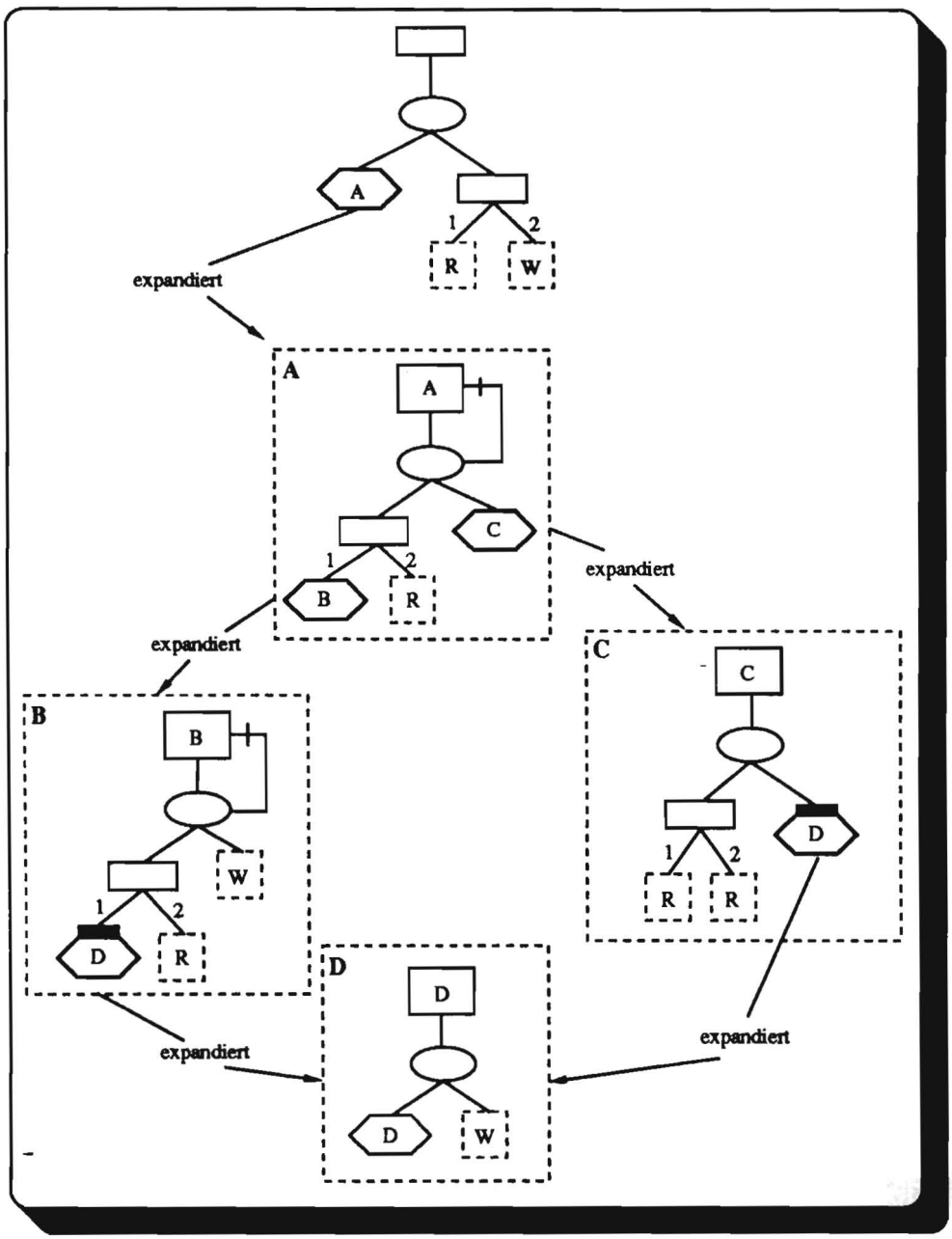

Abbildung 3.24: Modularisierung von Wissensbasen in DIWA

Kapitel 4

Bewertung der verschiedenen Wissensmodellierungsarten und Vorstellung einer Alternative in Form des Hybrids MOLTKE 3

Nachdem in den vorherigen Kapiteln die grundlegenden Wissensmodellierungsarten für technisches Diagnosewissen und einige bereits existierende Diagnose–Expertensysteme, die vorwiegend auf nur einer Wissensform basieren, vorgestellt wurden, soll in diesem Kapitel darauf eingegangen werden, welche Vorteile und welche Probleme durch die einförmige Modellierung technischen Wissens entstehen, und warum es auf lange Sicht sinnvoller ist Diagnose–Expertensysteme zu entwickeln, die mit verschiedenartig modelliertem Wissen arbeiten, sogenannte Hybride. Im Anschluß an diese Diskussion wird dann der Hybrid MOLTKE 3 vorgestellt, der sowohl mit heuristischem, als auch modellbasiertem und fallbasiertem Wissen arbeitet.

Die Probleme, die bei der Verarbeitung von in nur einer Form modelliertem Wissen auftreten, sind bei den verschiedenen Wissensmodellierungsarten unterschiedlich. Während bei den einen Systemen beispielsweise der Aufwand bei der Wissensakquisition enorm hoch ist, liegt das Problem bei anderen Systemen wiederum mehr in der vernünftigen Verarbeitung des akquirierten Wissens.

Die Vor– und Nachteile der einzelnen Wissensmodellierungsarten werden wieder abschnittsweise in folgender Reihenfolge beschrieben:

- heuristisch
- modellbasiert
- fallbasiert
- statistisch
- Entscheidungsbäume und –tabellen

Die Beurteilung der verschiedenen Wissensmodellierungsarten findet nach folgenden Gesichtspunkten statt:

Objektivierbarkeit (Korrektheit innerhalb des Modells?)

Korrektheit (Korrektheit in Bezug auf die Realität?)

Effizienz

Anwendungsspektrum

Anwendbarkeit für komplexe Probleme

Benutzerinteraktion (Kann der Benutzer eigene Vorschläge einbringen?)

Verständlichkeit

Aufwand bei der Wissensakquisition

Änderungsfreundlichkeit

Unsicherheiten (Inwieweit können Unsicherheiten erfaßt werden?)

Die Vor- und Nachteile und eine Bewertung der verschiedenen Wissensmodellierungsarten nach den gerade genannten Gesichtspunkten sind im Anschluß an diesen Abschnitt in Tabelle 4.1 und in Abbildung 4.1 noch einmal übersichtlich zusammengefaßt.

4.1 Heuristische Wissensmodellierung

Kernpunkt der heuristischen Wissensmodellierung sind die mit Unsicherheit behafteten, auf der Erfahrung eines Experten beruhenden Symptom–Diagnose–Assoziationen.

Die Tatsache, daß mit diesem Wissen auch Unsicherheiten modelliert werden, ist einer der Vorteile heuristischer Systeme, da das Expertenwissen nicht immer nur sicher ist, sondern auch der menschliche Experte mit unsicherem Wissen an einer Diagnose arbeitet. Aber nicht nur die menschliche Vorgehensweise ist ein Grund, Wissen heuristisch zu modellieren, sondern auch das Problem, daß man in manchen Bereichen aufgrund ihrer Unvollständigkeit gar keine andere Möglichkeit hat, als das Wissen durch Symptom–Diagnose–Assoziationen zu erfassen (man kann nicht in jedem Bereich ein Modell erstellen). Ein Beispiel ist hier die Erfassung medizinischen Wissens.

Wieder ein weiterer Vorteil liegt darin, daß man bei vom Experten gut geschätzten Assoziationen und guten Heuristiken bei der Verarbeitung des unsicheren Wissens häufig sehr gute Resultate erzielen kann. Mit sehr guten Resultaten ist hierbei gemeint, daß die Trefferquote, was richtige Diagnosen anbetrifft, bei heuristischen Systemen trotz der Verarbeitung vorwiegend unsicheren Wissens erstaunlich hoch liegen kann. Dies zeigt sich auch bei dem Diagnose–Expertensystem DAX, daß, wie in Kapitel 3 bereits erwähnt, auf der Expertensystemshell MED2 entwickelt wurde, und eine Trefferquote von 85% erreicht.

Außerdem kommt man in heuristischen Systemen meistens schnell zu einer Diagnose (geringer Overhead, keine Simulation im komplexen Modell).

In den Unsicherheiten als solche sind aber auch einige Nachteile heuristischer Modellierung zu finden. So ist es dem menschlichen Experten zwar häufig noch möglich, Wahrscheinlichkeiten für die einzelnen Symptom–Diagnose–Assoziationen anzugeben (wenngleich die meisten Experten sich hier nur sehr ungern festlegen lassen), aber auf die Frage, wie man diese Unsicherheiten durch ein heuristisches Verfahren angemessen verarbeiten kann, kann der Experte keine Antwort mehr geben, weil er bei sich selbst eigentlich auch nicht genau weiß, wie er dieses Wissen verarbeitet, um eine Enddiagnose stellen zu können. Es wird dem Ersteller des Systems überlassen, sich auf die Gratwanderung der Suche nach einem adäquaten Verfahren zu begeben. Voraussetzung für die Erzielung guter Resultate ist demnach einerseits ein guter Experte und andererseits ein guter Wissensingenieur.

Ein anderer Nachteil heuristischen Wissens liegt darin, daß man nur Fehler finden kann, die schon vorher einmal aufgetreten sind, und dementsprechend berücksichtigt wurden. Einen Vollständigkeitsanspruch kann man an solche Systeme also nur schwer stellen. So treten auch bei dem Auffinden von Mehrfachfehlern[1] häufig große Probleme auf, deren Auffinden bei einem modellbasierten System kein Problem darstellt.

Ein weiterer Nachteil dieser Wissensmodellierungsart liegt darin, daß man einen Bereich nicht unbedingt ganz in Form von Symptom–Diagnose–Assoziationen erfassen kann. Dies ist meistens nur bis zu einem bestimmten Detaillierungsgrad möglich, d. h. irgendwann muß dann z. B. auf kausales Wissen zurückgegriffen werden.

Wieder ein Nachteil ist, daß heuristisches Wissen nicht objektivierbar ist, wie es bei statistischem Wissen der Fall ist. Ein von einem heuristischen System erzieltes Ergebnis ist somit nicht so aussagekräftig, wie das eines statistischen Systems, weil kein mathematisches Modell dahintersteht, in dem korrekt gearbeitet wird.

Abhängig von dem Einsatzgebiet eines Diagnose–Expertensystems und den Ansprüchen, die an ein solches System gestellt werden, ist es insgesamt nicht unbedingt zu empfehlen, nur heuristisches Wissen zur Erstellung einer Diagnose heranzuziehen.

Heuristisch modelliertes Wissen kann dann vernünftig eingesetzt werden, wenn eine gewisse Fehlertoleranz gegeben ist und wenn Erfahrungswissen vorliegt. So kann man heuristisches Wissen beispielsweise bei der Autodiagnostik sehr gut einsetzen, da hier sehr viel Erfahrungswissen vorliegt und auch die ein oder andere Fehldiagnose nicht weiter tragisch ist (außer für die Geldbörse des Autobesitzers, der den Aufwand bezahlen muß). Anders sieht dies aus, wenn man Störfälle in einem Atomkraftwerk diagnostizieren möchte. Hierbei nur heuristisch modelliertes Wissen einzusetzen, wäre sicher zu riskant, da hier schon die erste Diagnose die richtige sein muß.

Die Nachteile wiegen alles in allem in manchen Gebieten sehr viel stärker, als in anderen, und es ist daher abzuwägen, wann man allein mit heuristischem Wissen diagnostizieren kann und soll.

[1] Unter Mehrfachfehlern versteht man das gleichzeitige Auftreten mehrerer Defekte.

4.2 Modellbasierte Wissensmodellierung

Bei der reinen modellbasierten Wissensmodellierung wird auf Erfahrungswissen verzichtet und stattdessen Struktur- und Funktionswissen des Diagnoseobjektes in den Vordergrund gestellt.

Daraus ergibt sich der Vorteil, daß man keine Unsicherheiten zu verarbeiten hat. Das Wissen ist damit bei vollständigem Modell objektivierbar. Mit einem vollständigen Modell des zu diagnostizierenden Objektes ist es außerdem möglich, alle Fehler, die innerhalb des Objektes auftreten können, zu finden. So stellen auch Mehrfachfehler hier kein Problem dar.

Ein weiterer Vorteil ist, daß man durch die Beschreibung des Gerätes auf verschiedenen Abstraktionsleveln die Möglichkeit hat, auf momentane Diagnosebedürfnisse seitens des Benutzers einzugehen (bei einigen Komponenten muß man nicht weiter in Details absinken; ein Autogetriebe wird beispielsweise häufig ganz ausgewechselt, ohne nach dem Fehler in seinem Inneren zu forschen).

Ein anderer Vorteil liegt darin, daß man relativ einfach einen Wechsel zwischen verschiedenen Modellen erfassen kann, da man das Wissen komponentenweise austauschen kann. Bei der heuristischen Modellierung müssen bei einem Modellwechsel eventuell Wahrscheinlichkeiten geändert werden, neue Regeln eingesetzt werden und alte herausgeworfen oder modifiziert werden, was alles in allem sehr schwer zu lokalisieren ist.

Die modellbasierte Wissensmodellierung bringt aber auch viele schwergewichtige Nachteile mit sich. So bedeutet diese Art der Modellierung einen schon für kleine Geräte enormen Akquisitionsaufwand. Für komplexe Anlagen ist eine vollständige kausale Modellierung eigentlich schon nicht mehr machbar. Dies kann auch bei den Systemen QUASIMODIS und Davis' System beobachtet werden. So schreibt [Reh91] selbst schon, daß man für ein sinnvolles Arbeiten mit QUASIMODIS ein darüberliegendes heuristisches Diagnose–Expertensystem benötigt, das mögliche Fehlerursachen vorschlagen soll, die dann von QUASIMODIS überprüft werden. Bei Davis' System werden nur sehr kleine Diagnoseobjekte modelliert.

Außerdem unterliegt die Arbeit mit modellbasiertem Wissen sehr starken Effizienzverlusten, da jeder Vorgang innerhalb des Diagnoseobjekts durch das Modell simuliert wird. Je komplexer das Diagnoseobjekt ist, desto stärker werden hierbei die Effizienzverluste.

Ein weiterer Nachteil modellbasierter Wissensmodellierung ist, daß Erfahrungswissen überhaupt nicht erfaßt wird. Aber gerade dieses Erfahrungswissen führt beim menschlichen Experten dazu, daß er im Durchschnitt sehr schnell zu einer richtigen Diagnose kommt. Außerdem werden auch Umwelteinflüsse, wie z. B. zu hohe Luftfeuchtigkeit, unter Umständen nicht berücksichtigt.

Modellbasiertes Wissen kann prinzipiell in jedem Bereich eingesetzt werden, in dem das Wissen vollständig vorliegt. Aus Effizienzgründen ist sein Einsatz jedoch nur in weniger komplexen Anwendungsgebieten zu empfehlen. So könnte z. B. ein Roboterarm in einer Bandstraße zur Autoproduktion kausal modelliert werden, weil er kein allzu komplexes Problem darstellt und vollständig zu modellieren ist. Eine komplexe CNC–Maschine würde dagegen nicht mehr vollständig modellbasiert erfaßt werden können (da das Wissen

einfach zu umfangreich ist).

Insgesamt führen die Nachteile „großer Aufwand bei der Modellierung“ und „starke Effizienzverluste bei der Diagnoseerstellung“ dazu, daß man die Erstellung eines Diagnose–Expertensystems mit modellbasiertem Wissen als einzige Wissensform nicht empfehlen kann, obwohl man hiermit bei vollständiger und korrekter Modellierung immer richtige Ergebnisse erhält.

4.3 Fallbasierte Wissensmodellierung

Mit der fallbasierten Wissensmodellierung wird das Erfahrungswissen erfaßt, das in Form von bereits diagnostizierten Fällen vorliegt.

Ein Vorteil dieser Wissensmodellierungsart ist die Effizienz, mit der es teilweise möglich ist, zu einer Diagnose zu kommen. Hat man einen entsprechenden Fall parat, so kann eine Diagnose sehr schnell gestellt werden. Der sich daraus ergebende Nachteil ist, daß, wenn man keinen entsprechenden Fall hat, gar keine Diagnose gestellt werden kann. Jeder Fehler muß also durch Fälle erfaßt werden, wenn man ihn diagnostizieren möchte.

Ein großer Vorteil fallbasierter Modellierung ist die prinzipielle Lernfähigkeit, die man Systemen, die mit diesem Wissen arbeiten, mit auf den Weg geben kann. Einmal lernt das System schon allein durch die selbständige Auffüllung seiner Wissensbasis durch von ihm gelöste Fälle. Zum anderen lernt es auch durch die Modifizierung von Symptomgewichtungen, d. h. es kann Änderungen daran vornehmen, wie relevant ein Symptom für einen Fall ist.

Um eine ganze Domäne diagnostizieren zu können, sind, je nach Größe der Domäne, sehr viele Fälle notwendig, da eine repräsentative Falldatenbasis benötigt wird. Das bedeutet, daß die Wissensakquisition einigen Aufwand verursachen kann (natürlich auch die anschließende Implementierung).

Außerdem kann man Wissen nur dann fallbasiert modellieren, wenn man schon entsprechende Fälle bearbeitet hat. Man kann zwar auch Analogien zwischen verschiedenen Maschinenmodellen ziehen, aber hier tritt dann gleich das nächste Problem auf. Die Analogiebildung selbst erweist sich als problematisch [Rus86].

Ein ideales Diagnose–Einsatzgebiet für die fallbasierte Wissensmodellierung ist die Medizin. Hier liegen Aufzeichnungen in Form von Patientenkarteien schon über Jahrhunderte vor, die eigentlich „nur“ in Falldatenbasen eingegeben werden müssen. Ungeeignet für die ausschließlich fallbasierte Modellierung sind alle neuen Gebiete, in denen es auf eine sichere Diagnose ankommt. Für die Fehlerdiagnose an einem Spaceshuttle würde man demnach nicht ausschließlich fallbasiertes Wissen verwenden.

Alles in allem führen die Nachteile großer Aufwand bei der Erstellung einer repräsentativen Falldatenbasis und die Tatsache, daß man nur schon bekannte Fehler diagnostizieren kann, dazu, daß von einer Diagnoseerstellung, die nur auf fallbasiertem Wissen beruht, in den meisten Anwendungsdomänen abgesehen werden sollte.

4.4 Statistische Wissensmodellierung

Die statistische Wissensmodellierung ist eigentlich die einzige Wissensform, die beim menschlichen Experten so nicht direkt vorliegt. Der menschliche Experte hat sein Wissen nicht in Form von statistischen Auswertungen vorliegen. Dies ist eher eine zusätzliche Möglichkeit, die man durch die Arbeit mit einem Rechner erhält. Mit statistischem Wissen werden, wie bei heuristischem Wissen auch, Beziehungen zwischen Symptomen und Diagnosen erfaßt, mit dem Unterschied, daß die Bewertungen dieser Regeln auf statistischen Auswertungen beruhen, und nicht einfach durch einen Experten geschätzt werden.

Ein Vorteil der statistischen Wissensmodellierung ist, ebenso wie bei der heuristischen Modellierung, daß hier unsicheres Wissen verarbeitet wird. Allerdings hat man hier wegen des dahinterstehenden Modells objektivierbare Ergebnisse.

Die statistische Modellierung hat jedoch einige Nachteile. So braucht man, um nur einigermaßen repräsentative statistische Auswertungen erstellen zu können, große Falldatenbasen. Diese hat man jedoch häufig nicht zur Verfügung.

Ein weiterer Nachteil ist, daß die Vollständigkeitsanforderung, ebenso wie bei der fallbasierten Modellierung, fast nie erfüllt werden kann; es können wiederum nur Fehler diagnostiziert werden, die in der Falldatenbasis erfaßt wurden.

Weitere Voraussetzungen, die die mathematisch korrekten Verfahren an das von ihnen zu verarbeitende Wissen stellen (siehe Kapitel 2), müssen erfüllt sein, sind jedoch nur selten erfüllt (z. B. Unabhängigkeit der Symptome untereinander). Verstößt man jedoch gegen eine von ihnen, so ist der Vorteil der Objektivierbarkeit der Ergebnisse hinfällig. Darüber kann dann keine Aussage mehr gemacht werden.

Bei statistischen Diagnosesystemen wird am häufigsten das Verfahren von Bayes zur Verrechnung der Wahrscheinlichkeiten eingesetzt. Hieraus entsteht auch wieder ein wichtiger Nachteil der statistischen Wissensmodellierung. Es kann nur direkt von Symptomen auf Enddiagnosen geschlossen werden, ohne daß über Zwischendiagnosen gegangen wird. Daraus ergibt sich das Problem, daß sämtliche Symptome schon zu Beginn der Diagnoseerstellung bekannt sein müssen, was für den Benutzer meistens sehr aufwendig ist.

Aufgrund ihrer schwerwiegenden Nachteile ist die statistische Wissensmodellierung als alleinige Modellierungsart für keine Anwendungsdomäne zu empfehlen. Als zusätzliche Modellierungsart zu einem fallbasierten (und eventuell auch noch heuristischen) System, kann sie jedoch in den Bereichen, in denen die zur Korrektheit des mathematischen Modelles benötigten Voraussetzungen erfüllt sind, sinnvoll eingesetzt werden. Beispielsweise könnten in der Autodiagnostik die Heuristiken anhand einer Falldatenbasis durch ein statistisches System teilweise verbessert werden.

4.5 Wissensmodellierung in Form von Entscheidungsbäumen und Entscheidungstabellen

Mit Entscheidungsbäumen wird Wissen modelliert, bei dem der Weg von Symptomen zu Diagnosen deterministisch festgelegt ist. Entscheidungstabellen geben für jede mögliche Diagnose explizit an, welche Symptome vorkommen, und welche Symptome nicht auftreten, d. h. jede Regel (siehe Kapitel 2) stellt einen Pfad von der Wurzel zu einem Blatt in einem entsprechenden Entscheidungsbaum dar.

Unter dieser Voraussetzung ist ein entscheidender Vorteil von Entscheidungsbäumen, daß man durch sie außerordentlich schnell zu einer Diagnose kommt, weil das Wissen nicht von der Verarbeitungsstrategie getrennt wird. Der gleiche Vorteil gilt für Entscheidungstabellen, da die Diagnose durch einen Sprung in die richtige Spalte der Tabelle (siehe Kapitel 2) erreicht wird.

Möchte man einen Entscheidungsbaum erweitern, oder auch nur abändern, so ist dies mit erheblichem Aufwand verbunden. Sie sind also sehr änderungsfeindlich. Bei Entscheidungstabellen tritt dieser Nachteil aber nicht auf.

Ein weiterer Nachteil ist die Größe und Unübersichtlichkeit beider Wissensmodellierungsarten bei komplexen Problembereichen. Dieser Sachverhalt macht die Akquisition des zur Diagnose benötigten Wissens sehr aufwendig.

Ein anderes Manko von Entscheidungbäumen ist die Gängelung des Benutzers. Es ist ständig eine Eingabe des Benutzers in vorgegebener Reihenfolge erforderlich. Dadurch ist eine Interaktion zwischen Benutzer und System, die es dem Benutzer ermöglicht eigene Vorschläge einzubringen, unmöglich. Auch Erfahrungen des Benutzers im Umgang mit einem solchen System können hier nicht berücksichtigt werden. Systeme, die mit dieser Wissensart arbeiten sind somit meistens sehr benutzerunfreundlich. Kommt die Eingabe jedoch direkt von einer Maschine, so kann sich der gerade beschriebene Nachteil als effizienter Vorteil erweisen.

Entscheidungstabellen fordern vom Benutzer sehr viele Symptome an. Kann der Benutzer die Symptome nicht liefern, so kann auch keine Diagnose gestellt werden.

Insgesamt sollte Wissen, daß in Form von Entscheidungsbäumen oder Entscheidungstabellen modelliert werden kann, nur in sehr kleinen Bereichen eingesetzt werden. In diesen Bereichen sollte man aus Effizienzgründen jedoch durchaus auf diese Wissensmodellierungsart zugreifen. Ein Beispiel für den Einsatz eines Entscheidungsbaumes oder einer Entscheidungstabelle wäre die Diagnose an einem Pneumatikzylinder, der wenig komplex und vollständig zu erfassen ist. Ungeeignet für die Modellierung durch einen Entscheidungsbaum oder eine Entscheidungstabelle ist der Bereich KFZ aufgrund seiner Komplexität. Entscheidungsbäume und Entscheidungstabellen sind als alleinige Wissensmodellierungsarten in den wenigsten Anwendungsdomänen zu benutzen.

Bei Entscheidungsbäumen und -tabellen sollte vorausgesetzt werden, daß das Wissen, das durch sie modelliert wird, auch hierfür geeignet sein muß, damit die Nachteile nicht so schwer ins Gewicht fallen.

Prinzipiell kann man zwischen zwei Arten von Entscheidungsbäumen und -tabellen

Modellierungsart	Vorteile	Nachteile
heuristisch	- Unsicherheiten können modelliert werden - Effizienz - gute Benutzerinteraktion	- sinnvolle Verrechnung von Unsicherheiten häufig schwierig - findet nur bereits bekannte Fehler - schwierig bei Mehrfachfehlern - nicht objektivierbar
modellbasiert	- keine Unsicherheiten - bei korrektem Modell objektivierbar - bei vollständigem Modell sind alle Fehler zu finden (auch Mehrfachfehler) - gute Benutzerinteraktion - gute Änderbarkeit	- aufwendige Wissensakquisition - geringe Effizienz - kein Erfahrungswissen
fallbasiert	- Effizienz - Lernfähigkeit - Unsicherheiten können modelliert werden	- fast nie vollständig, daher werden nur bekannte Fehler gefunden - aufwendige Wissensakquisition bei großen Falldatenbasen - Probleme bei Analogiebildung
statistisch	- objektivierbar - Unsicherheiten können modelliert werden	- viele Voraussetzungen für Korrektheit im Modell - große Falldatenbanken zur Auswertung nötig - fast nie vollständig, daher werden nur bekannte Fehler gefunden - benutzerunfreundlich (wenn reine Bayes–Form verwendet wird)
Entscheidungs-baum	- Effizienz - Unsicherheiten werden erfaßt	- unübersichtlich bei komplexen Problemen - aufwendige Wissensakquisition - änderungsfeindlich - keine Benutzerinteraktion
Entscheidungs-tabelle	- Effizienz - Unsicherheiten werden erfaßt	- aufwendige Wissensakquisition - unübersichtlich bei komplexen Problemen

Abbildung 4.1: Vor– und Nachteile der verschiedenen Wissensmodellierungsarten

4.6 Neue Wege in Form von Hybriden

Das Ziel eines technischen Diagnose–Expertensystems sollte sein, möglichst schnell mit möglichst wenig Aufwand zu einer richtigen Diagnose zu kommen. Beim menschlichen Experten ist dieser Sachverhalt meistens erfüllt. Das Problem liegt also darin, die menschlichen Fähigkeiten durch einen Rechner nachzuahmen.

Wie die vorherigen Abschnitte gezeigt haben, sind die verschiedenen Wissensmodellierungsarten, verwendet man sie alleine, für die meisten Domänen wenig geeignet. In entsprechenden Teilbereichen hat jedoch jede Wissenmodellierungsart ihre Vorzüge. Daher wird immer häufiger dazu übergegangen, sogenannte hybride Diagnose–Expertensysteme zu entwickeln. Durch solche Hybride ist es möglich, jeden Bereich des Domänewissens seiner Natur entsprechend zu modellieren. Das bedeutet zwar, daß man einiges Wissen auch mehrfach modelliert und ein derartiges System mit erheblichem Aufwand bei der Erstellung verbunden ist, man jedoch gleichzeitig die Möglichkeit hat, den menschlichen Experten dadurch zu simulieren. Die Verteilung sieht hierbei meistens folgendermaßen aus:

- Durch die *heuristische* Wissensmodellierung erhält man die Möglichkeit, durch Symptom–Diagnose–Assoziationen die Fehlerursache schnell auf gewisse Ursachenbereiche einzuschränken.
- Durch die *modellbasierte* Wissensmodellierung kann man einerseits die durch heuristisches Wissen erhaltenen Vorschläge, wo der Fehler liegt, überprüfen. Andererseits kann man aber auch an Stellen, wo man mit heuristischem Wissen (oder auch fallbasiertem Wissen) nicht mehr weiterkommt, ansetzen, um nun durch die Simulation im Modell zu einer Diagnose zu kommen.
- An den Stellen, wo man das Gefühl hat, mit heuristischem Wissen so nicht mehr weiter zu kommen, aber der Zugriff auf modellbasiertes Wissen viel Aufwand bedeutet, kann man *fallbasiertes* Wissen zur Diagnosefindung heranziehen. Häufig handelt es sich hierbei um komplexere Situationen oder um Ausnahmesituationen, in denen auch der menschliche Experte überlegt, ob er einen solchen Fall nicht schon einmal hatte (beim Auto könnte beispielsweise ein bestimmtes Geräusch, tritt es für den Experten zum ersten Mal auf, sehr viel Aufwand bedeuten, bis er schließlich eventuell zu einer Diagnose kommt, an die er im ersten Moment nicht im entferntesten gedacht hat. Tritt es jedoch wieder einmal auf, so erinnert er sich sicherlich an den Fall, daß es früher schon einmal da war und er zu einer für ihn verblüffenden Diagnose kam. Er wird also diesmal erst nach dem damaligen Fehler forschen, und erst wenn es nicht der richtige Fehler war, den restlichen Aufwand wieder auf sich nehmen).
- Wie bereits erwähnt wurde, liegt *statistisches* Wissen als solches nicht direkt beim Menschen vor. Bringt man aber Expertenwissen auf den Rechner, so hat man die Möglichkeit durch Auswertung einer Falldatenbasis die heuristische Wissensbasis zu objektivieren.

- Für kleine Bereiche kann man schließlich auf *Entscheidungsbäume* oder *Entscheidungstabellen* zurückgreifen. Häufig handelt es sich beim menschlichen Experten dabei um stereotype Handlungsabläufe, die sich in bestimmten Situationen als besonders schnell und sicher erweisen. Es ist auf bestimmten Ebenen manchmal im Interesse der Effizienz einfach sinnvoll, solches Wissen zu verwenden, anstatt durch komplizierte Heuristiken oder durch die aufwendige Simulation mit einem Modell eine Diagnose zu erstellen.

Wie das Wissen in den einzelnen Teilbereichen letztendlich modelliert werden soll ist insgesamt abhängig von der Anwendungsdomäne. Technisches Wissen kann beispielsweise immer kausal modelliert werden, aber es sollte immer eine Abwägung stattfinden, ob dies sinnvoll ist. Die gesunde Abwägung bleibt im Endeffekt wieder dem Experten und dem Wissensingenieur überlassen.

Im folgenden Abschnitt wird nun der Hybrid MOLTKE 3 vorgestellt, der mit heuristischem, modellbasiertem und fallbasiertem Wissen arbeitet, und bei dem das heuristische Wissen durch statistische Auswertung der vorhandenen und ständig erweiterbaren Falldatenbasis verbessert werden kann.

4.7 Der Hybrid MOLTKE 3 – Verschiedene Wissensmodellierungsarten in einem System

Nachdem einige Diagnose–Expertensysteme vorgestellt wurden, die hauptsächlich auf Wissen zugreifen, das in nur einer Form modelliert wurde, und nachdem die Vor– und Nachteile einer solchen Vorgehensweise diskutiert und die Entwicklung von Hybriden motiviert wurden, soll in diesem Abschnitt ein Diagnose–Expertensystem vorgestellt werden, das bei der Diagnoseerstellung auf verschiedene Wissensformen zurückgreifen kann, also ein Hybrid ist. Es handelt sich um MOLTKE 3 (Models, Learning and Temporal Knowledge in Expert Systems for Technical Diagnosis), ein Expertensystem zur Diagnose an CNC–Maschinen, das an der Universität Kaiserslautern unter Leitung von Prof. Dr. M. M. Richter entwickelt wurde [Ric91]. MOLTKE 3 ist, wie man dem Namen bereits entnehmen kann, eine Weiterentwicklung des ursprünglichen MOLTKE–Systems. Es ist aus dem MOLTKE 2 Basis–System [Ric91], PATDEX/2 [Weß90] und iMAKE [Reh91] zusammengesetzt. Auf alle Komponenten gehen wird im Verlauf dieses Abschnitts eingegangen.

Zunächst jedoch wird in Abbildung 4.2 der Gesamtaufbau von MOLTKE 3 graphisch dargestellt. Ebenso, wie bei den Systemen, die bisher vorgestellt wurden, liegt auch bei MOLTKE 3 wieder die Dreiteilung in

- *Dialogschnittstellen*
- *Problemlösungskomponente*
- *Wissensbasis*

vor. In diese Komponenten des Expertensystems wurden bei MOLTKE 3 nun noch weitere Systeme eingegliedert, die für bestimmte Teilaufgaben zuständig sind, die mit der Erstellung der gesamten Wissensbasis und deren Nutzung zusammenhängen. Was diese Systeme genau tun, wird im Verlauf dieses Abschnitts noch erläutert.

Die Wissensbasis setzt sich aus mehreren Bereichen zusammen:

- *Falldatenbasis*: hier werden die Fälle abgelegt, die vom Experten eingegeben wurden, und die Fälle, die MOLTKE (oder PATDEX) mit der Zeit löst.
- *Fallgedächtnis*: in dem Fallgedächtnis sind alle Fälle, die in der Falldatenbasis enthalten sind, repräsentiert. Mit diesem Fallgedächtnis arbeiten sowohl die PATDEX–Komponente, als auch die GenRule–Komponente.
- *heuristische Regelbasis*: in der heuristischen Regelbasis wird das heuristische Wissen in Form von Regeln abgelegt. Weitere heuristische Regeln werden mit der Zeit von der GenRule–Komponente hinzugefügt.
- *kausale Regelbasis*: hier wird das Modell der zu diagnostizierenden Maschine in Form von Kontexten und kausalen Regeln abgelegt.
- *Arbeitsspeicher*: der Arbeitsspeicher enthält die Informationen des gerade zu bearbeitenden Problems.

Die Dialogschnittstellen in MOLTKE 3 untergliedern sich in

- *Benutzerschnittstelle*
- *Erklärungskomponente*
- *Wissensakquisitionskomponente*

Über die Benutzerschnittstelle können sowohl der eigentliche Benutzer des Systems, als auch der Experte, der die Wissensbasis, die Problemlösungskomponente, etc. , testen möchte, mit MOLTKE 3 kommunizieren. Die *Erklärungskomponente* gibt zur Vorgehensweise des Systems Erklärungen, und zwar wiederum an den Benutzer und den Experten.

Eine der interessanten Komponenten in MOLTKE 3 ist die *Wissensakquisitionskomponente*. Über sie wird die Wissensbasis gefüllt, d. h. die Falldatenbasis, die heuristischen Regeln und der kausale Teil werden hier vom Experten eingegeben. Interessant ist dabei die Erstellung der kausalen Regelbasis. Der Experte hat die Möglichkeit, sein Wissen in Nicht–Regelform in die Wissensakquisitionskomponente einzugeben. Daraufhin kann nun das System iMAKE in Aktion treten.

iMAKE (incremental Model–based Automatic Knowledge Extractor) ist ein Wissensakquisitionswerkzeug, mit dessen Hilfe ein Maschinenmodell direkt vom Experten, d. h. ohne Zwischenschaltung eines Wissensingenieurs, in ein Diagnose–Expertensystem, in diesem Fall MOLTKE 3, eingegeben werden kann. Der Experte hat nun die Möglichkeit das Maschinenstrukturwissen und Maschinenfunktionswissen in Form von Aufbauplänen

und Funktionsschemata in das Diagnose–Expertensystem einzugeben. iMAKE bereitet dieses Wissen dann durch dessen Transformation in Regelform für MOLTKE 3 auf. In dieser Form kann das so erhaltene Wissen in den bereits vorhandenen, kontextorientierten Regelmechanismus integriert werden. Welches Wissen von MAKE für MOLTKE 3 aufbereitet wird, wird in Abbildung 4.3 gezeigt. Es handelt sich um modellbasiertes,

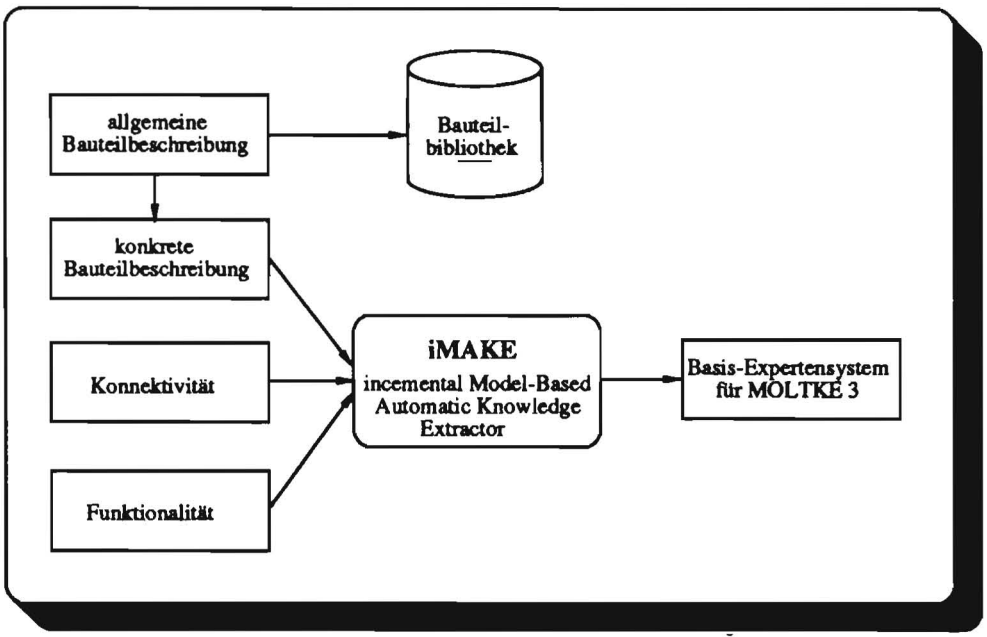

Abbildung 4.3: Funktion von iMAKE

funktionales Wissen, das in allgemeine Bauteilbeschreibung, konkrete Bauteilbeschreibung, Konnektivität der Bauteile und Funktionalität der Bauteile unterteilt ist, wie es auch schon bei Davis' System beobachtet werden konnte.

Für die Erstellung eines Tiefenmodells der zu diagnostizierenden Maschine werden die in der Maschine verwendeten Bauteile zunächst durch eine allgemeine Bauteilbeschreibung erfaßt. Da eine Komponente in einer Maschine mehrfach verwendet werden kann, wird ihre Bauteilbeschreibung möglichst allgemein definiert. Diese allgemeine Bauteilbeschreibung wird dann auch in eine Bauteilbibliothek übernommen. Ziel einer solchen Bauteilbibliothek ist es, daß bei der Modellierung einer neuen Maschine möglichst wenige Bauteile definiert werden müssen.

Der Maschinenaufbau wird hierarchisch repräsentiert. Unterschieden wird hierbei zwischen primitiven (atomaren) Bauteilen, die nicht weiter verfeinert werden können und komplexen Bauteilen, die aus mehreren primitiven und komplexen Bauteilen zusammengesetzt sind.

Die Verhaltensbeschreibung für ein primitives Bauteil muß vom Modellierer mitangegeben werden. Das korrekte Verhalten für ein komplexes Bauteil ermittelt iMAKE wiederum aus dem Zusammenwirken seiner Unterbauteile. Aber auch hier kann der Modellierer das Verhalten der Komponente angeben. iMAKE vergleicht eine solche Angabe mit dem von ihm selbst ermittelten Verhalten. Treten Inkonsistenzen auf, so wird die

Komponentenbeschreibung zurückgewiesen. Der Modellierer muß dann überprüfen, wodurch diese Inkonsistenzen aufgetreten sind. Zusätzlich wird dafür gesorgt, daß die Verhaltensbeschreibung jedes Bauteils vollständig ist, d. h. es wird überprüft, ob für jede Belegungskombination von Eingängen das Verhalten beschrieben ist.

Auch die Strukturangaben für komplexe Bauteile werden auf Konsistenz überprüft. Das bedeutet, es wird festgestellt, ob die miteinander verbundenen Unterkomponenten und die Verbindungen zwischen Ports der Ober- und Unterkomponenten bezüglich Port–Typ und Port–Richtung kompatibel sind, sowie ob alle Ports von Unterkomponenten Teil einer Verbindung sind.

Zusätzlich zu der Struktur- und Konnektivitätsbeschreibung eines Bauteils benötigt MAKE, um letztendlich eine Diagnose erstellen zu können, auch Informationen über die Fehler, die in den einzelnen Bauteilen auftreten können. Diese Fehlerbeschreibungen werden immer bei dem Bauteil mitabgespeichert, dessen Defekt für das Auftreten des Fehlers verantwortlich ist. Es handelt sich hierbei sowohl um primitive, als auch um komplexe Bauteile, abhängig davon, wieweit ein Fehler zurückverfolgt werden kann, oder wieweit eine Aufschlüsselung sinnvoll ist (das Getriebe eines Autos wird beispielsweise im Normalfall als ganzes Teil ausgewechselt, da es einfach zuviel Aufwand (im Sinne von Kosten und Nutzen) bedeutet, innerhalb dieses Bauteils nach einer genaueren Ursache zu forschen). Die Fehler, die bei einem Bauteil auftreten können, werden zusammen mit den bereits oben aufgeführten Beschreibungen in der Bauteilbibliothek abgelegt. Außerdem kann es sich für eine effiziente Diagnoseerstellung als durchaus sinnvoll erweisen die einzelnen Fehler mit Werten zu versehen, die die Häufigkeit ihres Auftretens repräsentieren (solche Werte können aus Statistiken, die vielleicht zu schon vorher existierenden Maschinen erstellt wurden oder von den Herstellern der Maschinenteile, gewonnen werden).

Schließlich muß auch das *intendierte Gesamtverhalten* der Maschine modelliert werden und in der Wissensbasis festgehalten werden. Mit dem intendierten Gesamtverhalten wird beschrieben, welche Verhaltensweisen der Maschine wesentlich für deren Funktionieren ist, und welche Verhaltensweisen nur zufällige Folgen der Maschinenzusammensetzung sind. Hierzu muß bekannt sein, wie die vorgesehene Belegungskombination der Input–Ports (d. h. die korrekten Eingaben) und die zu erwartenden Output–Ports aussehen.

Wie bereits oben erwähnt wurde, wird die heuristische Regelbasis von einem Experten erstellt. MOLTKE 3 verfügt aber auch über eine Lernkomponente, die GenRule (Generator of Empirical MOLTKE Rules) heißt und die die heuristische Regelbasis im Verlauf der Nutzung von MOLTKE 3 mit weiteren heuristischen Regeln auffüllt. Durch GenRule können jedoch auch Regeln aus der heuristischen Wissensbank wieder entfernt werden. GenRule verwendet für die Generierung neuer Regeln das Fallgedächtnis von MOLTKE 3. Aus dem Fallgedächtnis zieht GenRule neue Informationen, die dann in Form von heuristischen Regeln in die Wissensbasis integriert werden sollen. Es handelt sich dabei um dasselbe Fallgedächtnis auf das auch PATDEX zugreift. Mit dem von MAKE erstellten Modell der Maschine sollen diese neu generierten Regeln dahingehend getestet werden, ob die mit ihnen gezogenen Verbindungen überhaupt möglich sind.

Bei den von GenRule generierten Regeln handelt es sich um Abkürzungsregeln, d. h. um Regeln, die den Weg vom Symptom zur Diagnoseerstellung abkürzen. GenRule erlernt sie aus Analogien zwischen bereits in der Wissensbasis integrierten Diagnosepfaden

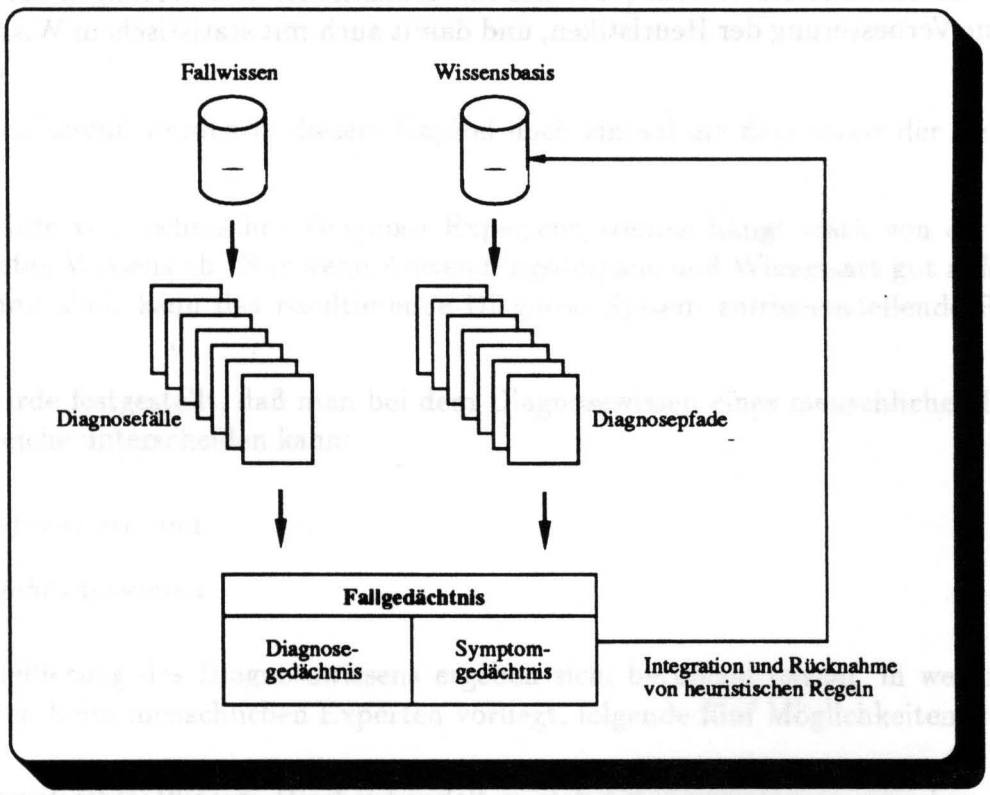

Abbildung 4.4: Die Vorgehensweise von GenRule

und neu präsentierten Fällen und orientiert sich dabei an dem „abkürzungsorientierten diagnostischen Problemlösen“ erfahrener Servicetechniker. Die Vorgehensweise von GenRule ist in Abbildung 4.4 graphisch dargestellt. Durch die neu generierten, in die heuristische Wissensbasis integrierten Abkürzungsregeln wird das Problemlösungsverhalten von MOLTKE 3 verbessert [Alt91].

Über die PATDEX-Komponente ist es in MOLTKE 3 nun möglich auch fallbasiert zu arbeiten. Dabei ist es möglich, daß MOLTKE 3 selbständig entscheidet, wann es fallbasiert arbeitet (also auf PATDEX/2 zugreift) und wann es mit dem anderen Wissen arbeitet.

Insgesamt ist MOLTKE 3 einer der wenigen Hybride, die auf heuristisches, modellbasiertes und fallbasiertes Wissen (teilweise mit statistischen Auswertungen der Falldatenbasis zur Verbesserung der Heuristiken, und damit auch mit statistischem Wissen) Zugriff haben.

Kapitel 5

Schlußbemerkung

Zusammenfassend werden in diesem Kapitel noch einmal die Ergebnisse der Arbeit aufgeführt.

Die Güte von technischen Diagnose–Expertensystemen hängt stark von der Art des verwendeten Wissens ab. Nur wenn Anwendungsdomäne und Wissensart gut aufeinander abgestimmt sind, kann das resultierende Diagnose–System zufriedenstellende Resultate liefern.

Es wurde festgestellt, daß man bei dem Diagnosewissen eines menschlichen Experten zwei Bereiche unterscheiden kann:

- *Tiefenwissen* und
- *Erfahrungswissen*

Zur Modellierung des Diagnosewissens ergeben sich, beruhend darauf, in welcher Form das Wissen beim menschlichen Experten vorliegt, folgende fünf Möglichkeiten:

- *heuristisches Wissen*: Hierbei handelt es sich um Assoziationen zwischen Symptomen und Diagnosen, die auf der Erfahrung eines Experten beruhen.
- *modellbasiertes Wissen*: Hierbei handelt es sich um das Struktur– und Funktionswissen des zu diagnostizierenden Gerätes.
- *fallbasiertes Wissen*: Hierbei handelt es sich um Wissen, das in Form von bereits gelösten Fällen vorliegt.
- *statistisches Wissen*: Hierbei handelt es sich um statistische Auswertungen bereits gelöster Fälle vor dem Hintergrund eines mathematischen Modells.
- *Wissen in Form von Entscheidungsbäumen oder –tabellen*: Wissen, das in Form von Testhierarchien bzw. Relationen von Symptomen und Diagnosen vorliegt.

In der folgenden Tabelle ist noch einmal übersichtlich dargestellt, welche Wissensmodellierungsart in welchen Wissensbereich vorwiegend eingeordnet werden kann:

	Erfahrungswissen	*Tiefenwissen*
heuristisch	X	
modellbasiert		X
fallbasiert	X	
statistisch	X	
Entscheidungstabelle	X	
Entscheidungsbaum	X	

Tabelle 5.1: Einordnung der verschiedenen Modellierungsarten

Bei der Diskussion der einzelnen Modellierungsarten stellte sich heraus, daß keine von ihnen uneingeschränkt für die Diagnose technischen Wissens zu empfehlen ist. Dennoch lassen sich rudimentäre Aussagen darüber machen, in welchen Anwendungsdomänen sich welche Wissensarten besonders eignen, oder sich als unbrauchbar erweisen. Am ehesten uneingeschränkt nutzbar, und deshalb wohl auch am meisten verbreitet, sind heuristische Diagnose–Expertensysteme. Aber auch die modellbasierte Modellierung findet zunehmend mehr Verwendung. Da aber auch diese beiden Wissensarten, wie alle anderen auch, viele Nachteile mit sich bringen, wurde vielerorts dazu übergegangen, mehrere Wissensarten in einem Diagnosesystem zu kombinieren. Die so entstandenen Hybride vermeiden die Nachteile der einzelnen Wissensmodellierungsarten und nutzen deren Vorteile. Natürlich bringt die Kombinierung mehrerer Wissensarten auch einen erhöhten Akquisitions- und damit Kostenaufwand mit sich, so daß im Einzelfall abgeklärt werden muß, ob die Vorteile des Hybrids gegenüber einem reinen System die Kostennachteile kompensieren. In Abbildung 5.1 ist die Beziehung zwischen Kosten und Nutzen in Form von einer Kurve graphisch dargestellt.

Es bleibt abzuwarten, ob die Tendenz zu hybriden Diagnose–Expertensystemen anhält und wie diese sich in der Praxis bewähren.

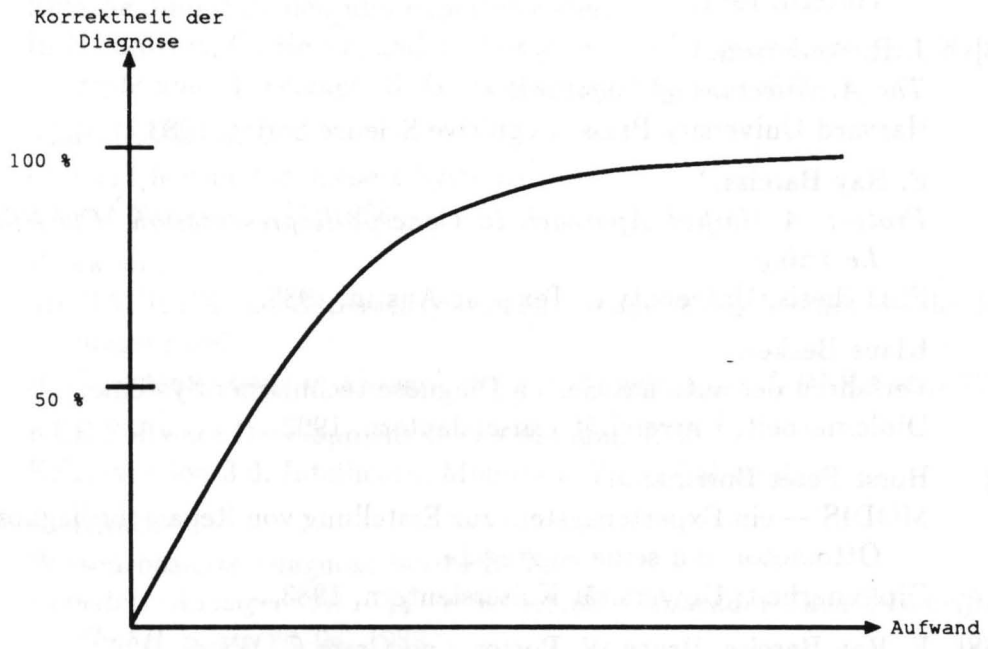

Abbildung 5.1: Verhältnis von Kosten und Korrektheit eines Diagnosesystems

Literaturverzeichnis

[Alt91] Klaus-Dieter Althofff.
Lernen von abkürzungsorientiertem diagnostischen Problemlösen.
SEKI Report SWP-91-05 (SFB), Fachbereich Informatik, Universität Kaiserslautern, 1991.

[And83] J. R. Anderson.
The Architecture of Cognition.
Harvard University Press, Cognitive Science Series, 1983.

[Bar88] E. Ray Bareiss.
Protos: A Unified Approach to Concept Representation, Classification and Learning.
PhD thesis, University of Texas at Austin, 1988.

[Bec92] Klaus Becker.
Verfahren der automatisierten Diagnose technischer Systeme.
Diplomarbeit, Universität Kaiserslautern, 1992.

[Bor83] Horst Peter Borrmann.
MODIS — ein Expertensystem zur Erstellung von Reparaturdiagnosen für den Ottomotor und seine Aggregate.
Diplomarbeit, Universität Kaiserslautern, 1983.

[BPW88] E. Ray Bareiss, Bruce W. Porter, und Craig C. Wier.
Protos: An Exemplar–Based Learning Apprentice.
In *Machine Learning: An Artificial Intelligence Approach*, volume 3. Morgan Kaufman Publishers, Inc., 1988.

[CM85] E. Charniak und D. McDermott.
Introduction to Artificial Intelligence.
Addison–Wesley, 1985.

[CMM86] Jaime G. Carbonell, Ryszard S. Michalski, und Tom M. Mitchell.
An Artificial Intelligence Approach.
In *Machine Learning*, volume I und II. Morgan Kaufman Verlag, 1986.

[Coh85] Anthony G. Cohn.
Deep Knowledge Representation Techniques.
In *Expertensysteme 85, Proceedings of the Fifth Technical Conference of the British Computer Society Specialist Group on XPS*, 1985.

[Dav84] Randall Davis.
Diagnostic Reasoning Based on Structure and Behaviour.
Artificial Intelligence, 24:347–410, 1984.

[DR83] Richard O. Duda und René Reboh.
AI and Decision Making: The PROSPECTOR Experience.
In Walter Reitman, editor, *Artificial Intelligence Applications for Business.* Alex Publishing Corporation, 1983.

[EPZ87] R. Eichhorn, R. D. Pütz, und J. Ziegler.
Expertensysteme zur Fehlerdiagnose an CNC–Maschinen.
In H. Balzert, G. Heyer, und R. Lutze, editors, *Expertensysteme '87 — Konzepte und Werkzeuge.* B. G. Teubner Stuttgart, 1987.

[GSJM87] U. Güntzer, C. Schöll, D. Jüttner, und K. Moll.
Entscheidungstabellen und expertensysteme.
In H. Balzert, G. Heyer, und R. Lutze, editors, *Expertensysteme '87 — Konzepte und Werkzeuge.* B. G. Teubner Stuttgart, 1987.

[Jac88] Alain H. Jackson.
Concept learning in Expert Systems.
Expert Systems, 5(2), 1988.

[Kar89] P. Karsten.
MEGAFILEX, ein Expertensystem zur Diagnose des Megafile — der Entwicklungsprozeß.
In *Expertensysteme in Entwicklung und Konstruktion.* VDI Berichte 775, 1989.

[kee86] KEE Software Development User's Manual, 1986.
KEE–Version 3.0, Intellicorp, Mountain View, California.

[Kir88] G. Kiratli.
Wissensbasierte Diagnose bei FFS.
Industrie–Anzeiger Nr. 44 vom 3.6.88, Konradin–Fachzeitschriftenverlag GmbH, pages 82–96, 1988.

[LME88] T. Legleitner, P. Mertens, und G. Ernst.
DAX — ein Echtzeit Diagnosesystem als Integrationskomponente in einem Prüfprozeß.
VDI–Z, 130(5), 1988.

[PS89] Markus Pischel und Kerstin Schäfer.
Maschinelles Lernen.
Projektarbeit, Universität Kaiserslautern, 1989.

[Pup86] Frank Puppe.
Assoziatives diagnostisches Problemlösen mit dem Expertensystem–Shell MED2.
Dissertation, Universität Kaiserslautern, 1986.

[Pup87] Frank Puppe.
Diagnostisches Problemlösen mit Expertensystemen.
Springer–Verlag, 1987.

[Pup88] Frank Puppe.
Einführung in Expertensysteme.
Springer–Verlag, 1988.

[Pup90] Frank Puppe.
Problemlösungsmethoden in Expertensystemen.
Springer–Verlag, 1990.

[Reh91] Robert Rehbold.
Integration modellbasierten Wissens in technische Diagnostik–Expertensysteme.
Dissertation, Universität Kaiserslautern, 1991.

[Ric89] Michael M. Richter.
Prinzipien der Künstlichen Intelligenz.
B. G. Teubner, 1989.

[Ric91] Michael M. Richter.
Das MOLTKE–Buch.
(in Vorbereitung), 1991.

[RS89] C. K. Riesbeck und R. C. Schank.
Inside case–based reasoning.
Lawrence Earlbaum, 1989.

[Rus86] S. J. Russel.
Analogical and inductive Reasoning.
PhD thesis, Stanford University, 1986.

[Sch89] J. Schneider.
Stand und Entwicklungstrends von Expertensystemen für Entwicklung und Konstruktion.
In *Expertensysteme in Entwicklung und Konstruktion.* VDI Berichte 775, 1989.

[Sch91] Gabriele Schmiedel.
DIWA — Diagnosewerkzeug mit graphischer Wissensakquisition.
Siemens AG, ZFE IS INF 31, 1991.

[Weß90] Stefan Weß.
PATDEX/2 – ein System zum adaptiven, fallfokussierenden Lernen in technischen Diagnosesituationen.
Diplomarbeit, Universität Kaiserslautern, 1990.

[WK89] M. Weck und G. Kiratli.
Wissensbasierte Diagnose für Flexible Fertigungssysteme.
In *Künstliche Intelligenz in der Fertigungstechnik.* Carl Hanser Verlag, 1989.

Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

DFKI
-Bibliothek-
PF 2080
D-6750 Kaiserslautern
FRG

DFKI Publikationen

Die folgenden DFKI Veröffentlichungen sowie die aktuelle Liste von allen bisher erschienenen Publikationen können von der oben angegebenen Adresse bezogen werden.
Die Berichte werden, wenn nicht anders gekennzeichnet, kostenlos abgegeben.

DFKI Publications

The following DFKI publications or the list of all publisched papers so far can be ordered from the above address.
The reports are distributed free of charge except if otherwise indicated.

DFKI Research Reports

RR-91-17
Andreas Dengel, Nelson M. Mattos:
The Use of Abstraction Concepts for Representing and Structuring Documents
17 pages

RR-91-18
John Nerbonne, Klaus Netter, Abdel Kader Diagne, Ludwig Dickmann, Judith Klein:
A Diagnostic Tool for German Syntax
20 pages

RR-91-19
Munindar P. Singh: On the Commitments and Precommitments of Limited Agents
15 pages

RR-91-20
Christoph Klauck, Ansgar Bernardi, Ralf Legleitner
FEAT-Rep: Representing Features in CAD/CAM
48 pages

RR-91-21
Klaus Netter: Clause Union and Verb Raising Phenomena in German
38 pages

RR-91-22
Andreas Dengel: Self-Adapting Structuring and Representation of Space
27 pages

RR-91-23
Michael Richter, Ansgar Bernardi, Christoph Klauck, Ralf Legleitner: Akquisition und Repräsentation von technischem Wissen für Planungsaufgaben im Bereich der Fertigungstechnik
24 Seiten

RR-91-24
Jochen Heinsohn: A Hybrid Approach for Modeling Uncertainty in Terminological Logics
22 pages

RR-91-25
Karin Harbusch, Wolfgang Finkler, Anne Schauder:
Incremental Syntax Generation with Tree Adjoining Grammars
16 pages

RR-91-26
M. Bauer, S. Biundo, D. Dengler, M. Hecking, J. Koehler, G. Merziger:
Integrated Plan Generation and Recognition
- A Logic-Based Approach -
17 pages

RR-91-27
A. Bernardi, H. Boley, Ph. Hanschke, K. Hinkelmann, Ch. Klauck, O. Kühn, R. Legleitner, M. Meyer, M. M. Richter, F. Schmalhofer, G. Schmidt, W. Sommer:
ARC-TEC: Acquisition, Representation and Compilation of Technical Knowledge
18 pages

RR-91-28
Rolf Backofen, Harald Trost, Hans Uszkoreit:
Linking Typed Feature Formalisms and Terminological Knowledge Representation Languages in Natural Language Front-Ends
11 pages

RR-91-29
Hans Uszkoreit: Strategies for Adding Control Information to Declarative Grammars
17 pages

RR-92-17
Hassan Aït-Kaci, Andreas Podelski, Gert Smolka:
A Feature-based Constraint System for Logic Programming with Entailment
23 pages

RR-92-18
John Nerbonne: Constraint-Based Semantics
21 pages

RR-92-19
Ralf Legleitner, Ansgar Bernardi, Christoph Klauck
PIM: Planning In Manufacturing using Skeletal Plans and Features
17 pages

RR-92-20
John Nerbonne: Representing Grammar, Meaning and Knowledge
18 pages

RR-92-21
Jörg-Peter Mohren, Jürgen Müller
Representing Spatial Relations (Part II) -The Geometrical Approach
25 pages

RR-92-22
Jörg Würtz: Unifying Cycles
24 pages

RR-92-24
Gabriele Schmidt: Knowledge Acquisition from Text in a Complex Domain
20 pages

RR-92-25
Franz Schmalhofer, Ralf Bergmann, Otto Kühn, Gabriele Schmidt: Using integrated knowledge acquisition to prepare sophisticated expert plans for their re-use in novel situations
12 pages

RR-92-26
Franz Schmalhofer, Thomas Reinartz, Bidjan Tschaitschian: Intelligent documentation as a catalyst for developing cooperative knowledge-based systems
16 pages

RR-92-27
Franz Schmalhofer, Jörg Thoben: The model-based construction of a case-oriented expert system
18 pages

RR-92-29
Zhaohur Wu, Ansgar Bernardi, Christoph Klauck:
Skeletel Plans Reuse: A Restricted Conceptual Graph Classification Approach
13 pages

DFKI Technical Memos

TM-91-11
Peter Wazinski: Generating Spatial Descriptions for Cross-modal References
21 pages

TM-91-12
Klaus Becker, Christoph Klauck, Johannes Schwagereit: FEAT-PATR: Eine Erweiterung des D-PATR zur Feature-Erkennung in CAD/CAM
33 Seiten

TM-91-13
Knut Hinkelmann:
Forward Logic Evaluation: Developing a Compiler from a Partially Evaluated Meta Interpreter
16 pages

TM-91-14
Rainer Bleisinger, Rainer Hoch, Andreas Dengel:
ODA-based modeling for document analysis
14 pages

TM-91-15
Stefan Bussmann: Prototypical Concept Formation
An Alternative Approach to Knowledge Representation
28 pages

TM-92-01
Lijuan Zhang:
Entwurf und Implementierung eines Compilers zur Transformation von Werkstückrepräsentationen
34 Seiten

TM-92-02
Achim Schupeta: Organizing Communication and Introspection in a Multi-Agent Blocksworld
32 pages

TM-92-03
Mona Singh
A Cognitiv Analysis of Event Structure
21 pages

TM-92-04
Jürgen Müller, Jörg Müller, Markus Pischel, Ralf Scheidhauer:
On the Representation of Temporal Knowledge
61 pages

TM-92-05
Franz Schmalhofer, Christoph Globig, Jörg Thoben
The refitting of plans by a human expert
10 pages

TM-92-06
Otto Kühn, Franz Schmalhofer: Hierarchical skeletal plan refinement: Task- and inference structures
14 pages

DFKI Documents

D-91-16
Jörg Thoben, Franz Schmalhofer, Thomas Reinartz: Wiederholungs-, Varianten- und Neuplanung bei der Fertigung rotationssymmetrischer Drehteile
134 Seiten

D-91-17
Andreas Becker:
Analyse der Planungsverfahren der KI im Hinblick auf ihre Eignung für die Abeitsplanung
86 Seiten

D-91-18
Thomas Reinartz: Definition von Problemklassen im Maschinenbau als eine Begriffsbildungsaufgabe
107 Seiten

D-91-19
Peter Wazinski: Objektlokalisation in graphischen Darstellungen
110 Seiten

D-92-01
Stefan Bussmann: Simulation Environment for Multi-Agent Worlds - Benutzeranleitung
50 Seiten

D-92-02
Wolfgang Maaß: Constraint-basierte Plazierung in multimodalen Dokumenten am Beispiel des Layout-Managers in WIP
111 Seiten

D-92-03
Wolfgan Maaß, Thomas Schiffmann, Dudung Soetopo, Winfried Graf: LAYLAB: Ein System zur automatischen Plazierung von Text-Bild-Kombinationen in multimodalen Dokumenten
41 Seiten

D-92-06
Hans Werner Höper: Systematik zur Beschreibung von Werkstücken in der Terminologie der Featuresprache
392 Seiten

D-92-07
Susanne Biundo, Franz Schmalhofer (Eds.):
Proceedings of the DFKI Workshop on Planning
65 pages

D-92-08
Jochen Heinsohn, Bernhard Hollunder (Eds.): DFKI Workshop on Taxonomic Reasoning Proceedings
56 pages

D-92-09
Gernod P. Laufkötter: Implementierungsmöglichkeiten der integrativen Wissensakquisitionsmethode des ARC-TEC-Projektes
86 Seiten

D-92-10
Jakob Mauss: Ein heuristisch gesteuerter Chart-Parser für attributierte Graph-Grammatiken
87 Seiten

D-92-11
Kerstin Becker: Möglichkeiten der Wissensmodellierung für technische Diagnose-Expertensysteme
92 Seiten

D-92-12
Otto Kühn, Franz Schmalhofer, Gabriele Schmidt: Integrated Knowledge Acquisition for Lathe Production Planning: a Picture Gallery (Integrierte Wissensakquisition zur Fertigungsplanung für Drehteile: eine Bildergalerie)
27 pages

D-92-13
Holger Peine: An Investigation of the Applicability of Terminological Reasoning to Application-Independent Software-Analysis
55 pages

D-92-15
DFKI Wissenschaftlich-Technischer Jahresbericht 1991
130 Seiten

D-92-16
Judith Engelkamp (Hrsg.): Verzeichnis von Softwarekomponenten für natürlichsprachliche Systeme
189 Seiten

D-92-17
Elisabeth André, Robin Cohen, Winfried Graf, Bob Kass, Cécile Paris, Wolfgang Wahlster (Eds.):
UM92: Third International Workshop on User Modeling, Proceedings
254 pages
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

D-92-18
Klaus Becker: Verfahren der automatisierten Diagnose technischer Systeme
109 Seiten

D-92-19
Stefan Dittrich, Rainer Hoch: Automatische, Deskriptor-basierte Unterstützung der Dokumentanalyse zur Fokussierung und Klassifizierung von Geschäftsbriefen
107 Seiten

D-92-21
Anne Schauder: Incremental Syntactic Generation of Natural Language with Tree Adjoining Grammars
57 pages

Möglichkeiten der Wissensmodellierung für technische Diagnose-Expertensysteme

Kerstin Becker

D-92-11
Document