KfK 5049
Mai 1992

Verbesserung der Betriebsführung technischer Anlagen durch den Einsatz von Methoden der Künstlichen Intelligenz

Projektbeschreibung INPRO

H. Keller, Th. Weinberger, E. Kugele,
A. Jaeschke, B. große Osterhues

Institut für Datenverarbeitung in der Technik
Projekt Schadstoff- und Abfallarme Verfahren

Kernforschungszentrum Karlsruhe

Kernforschungszentrum Karlsruhe

Institut für Datenverarbeitung in der Technik
Projekt Schadstoff- und Abfallarme Verfahren

KfK 5049

Verbesserung der Betriebsführung technischer Anlagen durch den Einsatz von Methoden der Künstlichen Intelligenz

Projektbeschreibung *INPRO*

H. Keller, Th. Weinberger, E. Kugele,
A. Jaeschke, B. große Osterhues

Kernforschungszentrum Karlsruhe GmbH, Karlsruhe

Als Manuskript gedruckt

Kernforschungszentrum Karlsruhe GmbH
Postfach 3640, 7500 Karlsruhe 1

ISSN 0303-4003

Zuammenfassung:

Das Ziel einer *innovativen Prozeßführung* ist die Optimierung des Betriebes technischer Anlagen/Verfahren im Sinne einer *prozeßintegrierten Vermeidung der Bildung von Schadstoffen* (aktiver Umweltschutz). Dies bedeutet die Integration verschiedener Methoden, um eine optimale Prozeßführung auch bei schwer zugänglichen und komplexen Prozessen zu ermöglichen. Solche Methoden reichen dabei von den konventionellen *numerischen Verfahren* über Beobachter, parametrische Diagnosemodelle und einer symbolischen Funktionsverarbeitung bis zur sogenannten *wissensbasierten Verarbeitung*.
Der vorliegende Bericht beschreibt die Problemstellung, die Anforderungen, die Konzepte und die Vorgehensweise im Projekt INPRO. Die verschiedenen Bereiche, welche konzeptionell integriert werden und im Bereich der Prozeßführung Anwendung finden sollen, werden skizziert. Die Problemstellungen werden grob umrissen und Lösungsmöglichkeiten aufgezeigt

Optimisation of process control through application of artificial intelligence methods in technical plants

Abstract:

The optimisation of the operation of technical systems or processes, in order to avoid the production of ecologically harmful substances, in sense of an active conservation of the environment, is the aim of an innovative process-control. Therefore various methods have to be integrated, in order to make an optimal operation of a technical process possible, even if it is complex and hardly accessible. Such methods vary from conventional numerical methods over observer, parametrical methods for diagnosis and a symbolic processing to so called knowledge-based methods.

This report describes the problem, the requirements, the concepts and the proceedings in the project INPRO. The different fields which have to be integrated conceptually and applied in the field of process-control are scetched. The problems are described and possible solutions are presented.

Inhaltsverzeichnis

1 Einleitung 1

1.1 Modellgestützte Messverfahren 2

1.2 Diagnoseverfahren 2

1.2.1 Parametrische und numerische modellbasierte Verfahren 2

1.2.2 Heuristische Diagnose 3

1.3 Wissensbasierte Prozeßführung 4

1.4 Automatische Wissenserfassung 5

2 Methoden der Informationsverabeitung 7

2.1 Analytisches Wissen - konventionelle Verfahren 7

2.2 Wissensbasierte Methoden 8

2.2.1 Allgemeine wissensbasierte Verfahren 9

2.2.2 Technisch orientierte wissensbasierte Verfahren 11

2.2.3 Das Problem der Wissensakquisition 12

3 Wissensbasierte Prozeßführung 13

3.1 Allgemeine Störungserkennung und -diagnose 13

3.2 Wissensbasierte Störungsfrühdiagnose 14

3.3 Anwendung neuronaler Netze 15

4 Berücksichtigung von analytischem Wissen 17

4.1 Elementare Verfahren 17

4.2 Modellbasierte Verfahren 18

4.2.1 Numerische modellbasierte Verfahren 18

4.2.1.1 Modellgestützte Messung 18

4.2.1.2 Modellgestützte Diagnose 19

4.2.1.3 Beispiel 20

4.2.1.4 Constraint-Techniken 21

4.2.2 Symbolische modellbasierte Verfahren 21

4.2.2.1 Algebraische Verarbeitung, Termersetzungsverfahren 21

4.2.2.2 Kausale Verarbeitung (Ursache-Wirkungs-Beziehungen), qualitative Verarbeitung 22

5 Berücksichtigung von explizitem heuristischem Wissen 23

5.1 Regelorientierte Verarbeitung 23

5.2 Objektorientierte Ansätze 24

5.2.1 Die Basiskonzepte objektorientierter Programmierung 24

5.3 Schemaorientierte Ansätze 25

5.4 Wahrscheinlichkeitsbetrachtungen und unscharfes Wissen 27

5.4.1 Wahrscheinlichkeitsbetachtungen 27

5.4.2 Fuzzy Logik 27

5.4.2.1 Beispiel 27

6 Berücksichtigung von implizitem Wissen (neuronale Netze) 29

6.1 Die Grobstruktur des menschlichen Gehirns 29

6.2 Die Grundelemente neuronaler Netze 30

6.2.1 Das Prozeßelement 30

6.2.2 Die Funktionen (Prozesse) des Prozeßelements 31

6.2.2.1 Lernfunktionen 32

6.2.3 Die Kombination von Prozeßelementen zu einem Layer 33

6.2.4 Die Kombination mehrerer Layer zu einem mehrschichtigen neuronalen Netz 33

6.3 Lernen in neuronalen Netzen 35

6.3.1 Verschiedene Formen des Lernens in neuronalen Netzen 36

6.3.2 Lernregeln 37

7 Das Projekt "Innovative Prozeßführung" 41

7.1 Methodische Ansätze 41

7.2 Randbedingungen 43

8 Werkzeuge 47

8.1 Das K_advice-System 47

8.1.1 Allgemeines 47

8.1.2 Die Graphische Oberfläche GAP / XGAP 48

8.1.3 Modellierung 48

8.1.4 Simulationslaufzeitsystem 50

8.1.5 Analysekomponente 50

8.1.6 Erweiterungen / Übertragbarkeit 50

8.2 Sprache 1 51
8.2.1 Allgemeines 51
8.2.2 Sprachkonzept 52
8.2.2.1 Regeldarstellung 53
8.2.2.2 Regelausführung 53
8.2.2.3 Konfliktlösungsstrategie 54
8.2.3 Beispiel "Kompressor" 55
8.2.3.1 Objekt Pumpe P 56
8.2.3.2 Objekt Ventil V1 56
8.2.3.3 Objekt Ventil V2 57
8.2.3.4 Objekt Ventil V3 57
8.3 Das C^3R-System zur automatischen Wissenserfassung 58
8.3.1 Allgemeines 58
8.3.2 Das Ziel des C^3R-Systems 59
8.3.3 Das Konzept des C^3R-Systems 60
8.3.4 Stand/Planung 62

9 Literaturüberblick, geordnet nach relevanten Fachbereichen 63

9.1 Echtzeit-Anwendungen 63
9.2 Expertensysteme 65
9.3 Fuzzy Logik 66
9.4 Künstliche Intelligenz (allgemein) 67
9.5 Maschinelles Lernen 68
9.6 Neuronale Netze 69
9.7 Projekt INPRO 70

10 Begriffe 73

Referenzen 77

1 Einleitung

Der anhaltende Fortschritt im Bereich der Hardware, welcher weiter steigende Leistung bei sinkenden Preisen bietet, und die kontinuierliche Weiterentwicklung der Softwaretechniken erschließen heute neue Möglichkeiten, den Wirkungsgrad und die Verfügbarkeit einer Anlage mit Hilfe weiter verbesserter, d. h. intelligenterer Prozeßführungssysteme noch zu steigern.

Zur Integration von betriebs- und prozeßspezifischer Intelligenz in Prozeßführungssystemen bietet sich u. a.

- der Einsatz von Prozeßmodellen und
- der Einsatz von wissensbasierten Methoden

an, deren Möglichkeiten in der Praxis bisher noch nicht voll genutzt werden.

Betrachtet man einen technischen Prozeß hinsichtlich der Bildung von Schadstoffen, so muß ein *intelligenter Ansatz zur Prozeßführung*, aufgrund der offensichtlichen Vorzüge der Schadstoffvermeidung gegenüber der Schadstoffbeseitigung, grundsätzlich eine Minimierung bzw. Vermeidung der entstehenden Schadstoffe anstreben, um somit die Problematik der Beseitigung von Schadstoffen quasi "an der Wurzel" anzugehen.

Technische Anlagen sind komplexe Prozesse und besitzen einen hohen Vernetzungsgrad bzw. eine große Anzahl an Rückführungen. Ihre Führung stellt daher eine hohe kognitive Belastung für den Bediener dar. Dies geht sogar so weit, daß eine aufkommende Störung aufgrund der Kombination geringer Abweichungen mehrerer Prozeßgrößen unter Umständen von einem Bediener im Ansatz nicht erkannt werden kann. Erst nach Eintritt der Stör- ung ist diese für ihn als solche erkennbar und erlaubt es ihm erst dann Gegenmaßnahmen zu ergreifen. Durch eine rechnergestützte Überwachung der Prozeßgrößen erweitert um Methoden der Störungsfrüherkennung können Störungen identifiziert und auch Ursachen in Form von Kombinationen geringer Abweichungen erkannt und bewertet werden.

Zur Störungs-(früh-)diagnose können Gradientenverfahren mit Bereichsanalysen, parametrische oder modellbasierte Verfahren eingesetzt werden /21/. Mathematische Modelle sind dabei auch symbolisch bzgl. den definierten kausalen Beziehungen verarbeitbar und liefern die möglichen Beeinflussungsgrößen, welche die Störung verursacht haben können.

Da man bei regelungstechnisch schwer beherrschbaren bzw. nicht zugänglichen Prozesse über kein mathematisches Modell verfügt können solche Prozesse nur durch Identifikation der zugehörigen heuristischen Beziehungen stationär stabil gefahren werden. Die Grundlagen sind die heuristischen Modelle (Strategien) der Bediener. Die heuristischen Modelle beschreiben dabei die grundsätzlichen kausalen Zusammenhänge und sind mit Methoden der Künstlichen Intelligenz (KI) auf Rechner übertagbar und damit zur Prozeßführung einsetzbar.

Hierzu werden im Projekt INPRO (Innovative Prozeßführung) Werkzeuge auf der Basis von

- algorithmischen / imperativen,
- symbolischen und
- konnektionistischen

Methoden entwickelt und angewandt.

1.1 Modellgestützte Messverfahren

Modellgestützte Messungen oder Meßverfahren dienen zur Bereitstellung von Prozeßkenngrößen (-zuständen), die ähnlich wie Meßgrößen die Kenntnis des aktuellen Prozeßzustandes verbessern /21/.

Die zur rechnergestützten Überwachung der Prozeßgrößen notwendigen Informationen sind in Form von Meßwerten oder über Verfahren, die auf einem *formalen* oder *mathematischen Modell* basieren (z. B. /2/), verfügbar. Einfachste Verfahren hierzu sind die Berechnung von Gradienten und Bilanzgrößen.

Eine Klasse modellgestützter Meßverfahren sind die sogenannten **Zustandsschätzverfahren**. Bei der Zustandsschätzung besteht die Aufgabe darin, aus der Kenntnis der Prozeßparameter und der vorliegenden Meßgrößen unzugängliche innere Zustandsgrößen eines Prozesses zu berechnen. Zu diesem Zweck werden mathematische Gleichungen eines geeigneten Prozeßmodells in Echtzeit parallel zum Prozeß gelöst. Die ausgewählten Zustandsgrößen können dann an diesem Modell abgegriffen werden.

Bestimmte Prozeßbereiche sind allerdings weder einer direkten, noch einer modellgestützten meßtechnischen Methode zugänglich. Versagen solche analytischen und experimentellen Methoden, so können nur noch Methoden, die auf heuristische Beziehungen basieren eingesetzt werden.

1.2 Diagnoseverfahren

1.2.1 Parametrische und numerische modellbasierte Verfahren

Die Aufgabe der modellbasierten Diagnose besteht darin, daß eine (im Ansatz) erkennbare Störung (ein Symptom) über eine vorhandene Beschreibung des kausalen Wirkungsgefüges (Modell) auf die zugrundeliegende Ursache zurückgeführt werden muß. Als Hilfsmittel können hierzu auch die zur modellgestützen Messung in Form eines mathematischen Modells

vorliegenden mathematischen Beziehungen verwendet werden (primär theoretische Modelle).

Eine klassische Methode der modellbasierten Diagnose ist die Fehlerdiagnose mittels **Parameterschätzung**. Parameterschätzverfahren können, zum Zwecke einer Störungsfrühdiagnose, zur genauen Ermittlung der meßtechnisch nicht direkt zugänglichen Prozeßmodellparameter bzw. deren Abweichungen von einem Nominalzustand eingesetzt werden. Der Grundgedanke bei einem Einsatz der Parameterschätzung zur Fehlerdiagnose besteht darin, daß viele Prozeßfehler als Änderung von Prozeßparametern auftreten. Durch dieses modellbasierte Vorgehen können mehr Störungen früher erkannt und lokalisiert werden und es sind hierzu lediglich die Signale weniger, im günstigsten Fall bereits vorhandener, robuster Sensoren notwendig (/6/).

Schematisch kann man die Vorgehensweise bei der parametrischen, modellbasierten Diagnose wie folgt darstellen:

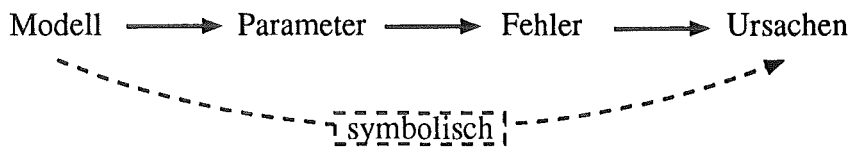


Abbildung 1

Das mathematische Modell liegt in der Regel in Form eines algorithmisch formulierten Programms, d. h. explizit und prozedural, vor. Um die in einem mathematischen Modell definierten Beziehungen (in Form von Ursache-Wirkungs-Beziehungen) ausnutzen zu können, muß das Modell in einer symbolischen Form analog wie Regeln u. ä. auf dem Rechner dargestellt und verarbeitet werden. Die klassische Form der Verarbeitung von Modellen zur Ableitung numerischer Werte ist demnach um die symbolische Verarbeitung der Modelle zu erweitern. Die Vorgehensweise der symbolischen modellbasierten Diagnose besteht dann, wie oben bereits dargestellt, aus einer direkten Ableitung der Ursachen vom gegebenen Modell.

1.2.2 Heuristische Diagnose

Ist ein analytischer Zugang, wie er oben beschrieben wurde, nicht möglich, so besteht noch die Möglichkeit zur Diagnose heuristische Beziehungen einzusetzen. Diese Beziehungen spiegeln in aller Regel das Wissen eines Bedieners wider, der über große Erfahrung im Umgang mit dem entsprechenden Prozeß verfügt. Daß man jedoch eine strikte Trennung zwischen analytischen und heuristischen Beziehungen durchführen muß, sei durch die folgenden Punkte verdeutlicht:

- Heuristische Beziehungen spiegeln unsichere, vage und nicht analytische Zusammenhänge wider.

- Der Bediener kann Beziehungen, die das Verhalten des Prozesses oder des Systemes beschreiben leichter in heuristischer Form, d. h. in Form von "Daumenregeln" wieder-

geben als in Form exakter Beziehungen (Regeln), da die heuristische Form dem menschlichen Naturell näher kommt.

- In gewisser Weise sind heuristischen Beziehungen mächtiger als analytische Beziehungen, da zum einen die Heuristik dort beginnt, wo die Analyse keine Erfolge mehr verspricht (sei es aufgrund der Komplexität oder weil ihr einfach die Mittel fehlen) oder zum zweiten weil sich die Heuristik auch auf Wissen zur Verwendung von Wissen (sogenanntes Metawissen vgl. Abschnitt 1.3) bezieht.

Doch nicht allein das durch einen eingeschränkten analytischen Zugang stark begrenzte Einsatzgebiet analytischer Verfahren und Methoden zur Prozeß- und Systemsteuerung machen den Einsatz neuer Techniken und Methoden notwendig. Bei den meisten Anwendungen handelt es sich, im Gegensatz zum klassischen *off-line* Einsatz von Expertensystemen, um Anwendungen innerhalb eines Prozeßführungssystems unter Echtzeitbedingungen (*on-line*). Erschwerend kommt hinzu, daß der zugehörige technische Prozeß dabei örtlich verteilt und die dynamischen Abläufe hochgradig parallel sein können. Situationen, welche gerade relevant waren und in Bearbeitung sind, können irrelevant werden bzw. kritische Folgesitutationen erzeugen. Die Dringlichkeit einer Bearbeitung (Situation, Symptomatik) kann sich innerhalb bestimmter Zeitspannen (Prozeßdynamik) völlig verändern.

1.3 Wissensbasierte Prozeßführung

Das Ziel der wissensbasierten Prozeßführung besteht in der Führung eines technischen Prozesses bzgl. vorgegebener Sollwerte und nicht nur in der Diagnose von Fehlern. Heuristisches Wissen spiegelt die Erfahrung der Bediener wider. Es hat einen statistischen Charakter und ist nicht immer durch eine Axiomatik erklärbar. Prozeßbediener führen einen technischen Prozeß ohne dies analytisch auf die den Prozeß beschreibenden Differentialgleichungen/ Ursache-Wirkungs-Beziehungen zurückführen zu können. Heuristisches Wissen ist auch Wissen um die Anwendung von Wissen (z. B. die Auswahl einer Lösungsmethode). Wissen über die Verarbeitung von Wissen wird als **Metawissen** bezeichnet und im Bereich der Kognitionswissenschaften/Lerntheorien untersucht. Regeln bieten eine guten Ausgangspunkt, um Expertenwissen auf den Rechner zu übertragen. Eine auf heuristischem Wissen basierende Regel kann wie folgt aussehen:

```
WENN Temperatur T_1 > 180° und Druck D_1 < 30 bar
     DANN Regler R_2 auf Position 3.
```

Einfache Verfahren werden bereits vereinzelt in industriellen Anlagen eingesetzt (z. B. wird die Fuzzy Control bei der Überwachung einer Zementbrennanlage von einem dänischen Anlagehersteller seit Jahren erfolgreich eingesetzt, /33/).

Expertensysteme besitzen im Bereich der Informationssysteme bevorzugt Anwendungsmöglichkeiten auf der Prozeßführungs- und der Betriebsführungsebene z. B. im Bereich der Betriebsplanung (z. B. Vorbereitung und Durchführung von Wartungs- und Reparatur-

maßnahmen), Ablaufsteuerung, -optimierung und -überwachung (Prozeßsituationsabhängige Bedienerstrategien, Störfall- und Notstrategien), Bedienoberfläche (Prozeßsituationsabhängige Informationsauswahl und -darstellung) und Fehlerdiagnose (Nachweis von Geräteausfällen, Gerätefehler, Bedienerfehler).

Die bisher häufigsten Anwendungen von Expertensystemen in Prozeßinformationssystemen liegen im Bereich der Fehlerdiagnose. In der Betriebsplanung sind bereits zahlreiche Systeme im Einsatz, dies allerdings nur dezidiert für einzelne Planungsfunktionen und nicht integriert in Prozeßinformationssysteme. Expertensysteme zur intelligenten Steuerung von Bedieneroberflächen existieren erst in Ansätzen. Im Bereich Ablaufsteuerung und Optimierung sind nur wenige Entwicklungen bekannt.

1.4 Automatische Wissenserfassung

Alle Arten von erfaßten Beziehungen (Wissen) werden grundsätzlich in einer formalen Weise dargestellt (repräsentiert). Das Vertrauensintervall (confidence Interval, d. h. das Maß der Gültigkeit) der erhaltenen Beziehungen nimmt dabei ausgehend von den theoretischen Modellen bis zu letztlich rein heuristischen Beziehungen deutlich ab.

Da die Wissenserfassung durch Befragung eines oder mehrerer Experten sehr problematisch ist, muß man versuchen dieses Wissen auf eine andere Art und Weise zu erhalten. Kausale Zusammenhänge können über (maschinelle) Lernverfahren automatisch abgeleitet und zur Unterstützung des Bedieners bei der Symptom- und Situationsklassifikation sowie der Ableitung von Gegenmaßnahmen eingesetzt werden.

Die Anwendung neuronaler Netze (siehe Kapitel 3.3 und 6) stellt eine weitere Möglichkeit, Bedienerwissen automatisch abzuleiten, dar.

2 Methoden der Informationsverabeitung

2.1 Analytisches Wissen - konventionelle Verfahren

Bei technischen Systemen besteht das Wissen aus der Kenntnis über die einzelnen Komponenten, deren Struktur bzw. deren Funktion und deren Interaktionsmuster. Das Wissen besitzt dabei statische und dynamische Eigenschaften und kann in analytisches und heuristisches Wissen unterteilt werden, wobei die Grenze zwischen analytischem und heuristischem Wissen stark von der Prozeßkenntnis abhhängt. Durch den Wissenserwerb wird im allgemeinen im Laufe der Zeit das analytische Wissen auf Kosten des heuristischen Wissens größer, /10/.

Analytisches Wissen läßt sich auf einer unterliegenden Unterscheidungsebene durch die Dinge und deren Beziehungen auf dieser Ebene erklären (Axiomatik). Analytisches Wissen ist eng verknüpft mit seiner Darstellungs- und Verarbeitungsform (Symbolik, Axiomatik, Theorie), da erklärbares konzeptionelles Wissen immer auf eine abstrakte Darstellung abgebildet wird. Der einer abstrakten Darstellung zugrundeliegende Formalismus stellt zugleich das Erklärungs- und Verarbeitungsinstrument für das Wissen (den betrachteten Sachverhalt) dar. Wegen der Komplexität der Zusammenhänge sind nur die wenigsten Sachverhalte exakt und komplett in analytisches Wissen übertragbar. In der Regel kann das in einem Sachverhalt enthaltene explizite und implizite Wissen nur zu einem Teil in analytisches Wissen umgesetzt werden. Allerdings wirkt das Axiomensystem (Formalismus) als Filter, der das nicht darstellbare Wissen unterdrückt und hat daher nur Modellcharakter.

Die Realität wird vereinfacht, es wird von unwesentlichen Gegebenheiten abstrahiert. Die erfaßten Beziehungen im Sinne ihrer Gültigkeit bleiben allgemein nachprüfbar. Eine allgemeine Transparenz über mehrere Ebenen ist nicht immer gegeben. Beispielsweise läßt sich eine Strömung makroskopisch über den Druck, den Rohrquerschnitt und über eine Reibungszahl beschreiben. Der Vorgang ist auf dieser Ebene erklärbar. Wollte man dieses System auf der mikroskopischen Ebene der Molekel erklären, so schlägt dies aufgrund der Komplexität der Beziehungen und Parametergrößen fehl. Die Interaktion der Flüssigkeitsteilchen miteinander und der Wand, sowie deren örtliche Veränderung mit Impuls usw. kann höchstens qualitativ in Teilen erklärt werden. Ein makroskopisches Ergebnis ist dann nur über eine Wahrscheinlichkeitsbetrachtung erreichbar.

Wissen über einen technischen Prozeß kann von einfacher oder komplizierter Natur sein:

- Wert einer Prozeßgröße, z. B. $T_M = 5{,}50°C$,
- Wert und zugehöriger statischer Referenzbereich,
- dem Gradienten einer Größe,
- dem dynamischen Bereich für das Übergangsverhalten,
- einem Modell des dynamischen Verhaltens,
- mit Korrekturterm als Beobachter zur Schätzung nichtmeßbarer Größen,

- statistische Beschreibungen,
- Wenn-Dann-Beziehungen (kausale Zusammenhänge).

Die konventionelle Form zur Repräsentation und Verarbeitung von Wissen ist die imperative Programmierung mit der expliziten Angabe der Zusammenhänge und der Verarbeitungsfolge in elementaren Schritten. Die Sprachen bzw. deren Verarbeitungsmodell orientieren sich stark an der "von Neumann" Architektur. Es handelt sich um algorithmische bzw. prozedurale/imperative Sprachen. Das explizite Wissen wird in Form von Anweisungen und deren Reihenfolge auf einer sehr elementaren Ebene dargestellt (Automaten). Die Form ist effizient bzgl. der Prozessierung, aber unflexibel bzgl. einer modifizierten Aufgabenstellung. Alle abzudeckenden Probleme bzw. Alternativen müssen vorher explizit auf der prozeduralen Ebene berücksichtigt werden. Die "semantische Lücke", diese entspricht dem Aufwand der Übertragung der in einer natürlichen Sprache formulierten Problemstellung in die zur Lösung bereitgestellten Sprachelemente, ist groß. Allerdings sind mit dieser Verarbeitungsform mächtige Modelle, wie z. B. Simulationssysteme (Interpreter) für diskrete Prozesse realisiert worden. Die klassische Form ist durch die numerische (zahlenmäßige) Datenverarbeitung gegeben.

2.2 Wissensbasierte Methoden

Bei komplexen verfahrenstechnischen Anlagen (z. B. Verbrennungsprozese) sind die physikalischen und chemischen Wechselwirkungen weder theoretisch noch über experimentelle Identifikation ableitbar. Eine modellorientierte Prozeßführung wie z. B. die Berechnung der notwendigen Stellgrößen über ein mathematische Modell anhand der Meßgrößen ist nicht möglich. Erschwerend kommt oft hinzu, daß die Eingangsgrößen (Stoffzusammensetzungen) nicht vollständig bekannt sind.

Ungeachtet dieser Probleme können die Operateure einen solchen Prozeß hinreichend stabil fahren. Die Strategien der Operateure bauen auf ursächlichen kausalen Beziehungen. Sie berücksichtigen direkt keine physikalischen Parameter und sind oft vage oder unscharf in ihren Beziehungen (*falls die Temperatur T_{250} schnell steigt, dann kühle stark*). Solche Strategien werden als heuristische Modelle (hier bzgl. der Prozeßführung) bezeichnet und können mit modernen Verfahren der Informatik auf Rechner abgebildet und ausgeführt werden. Die notwendigen Methoden werden als wissensbasiert (symbolisch im Gegensatz zu numerisch) bezeichnet und stammen aus dem Bereich der Künstlichen Intelligenz. Die Summe aller heuristischen Beziehungen stellt in Verbindung mit einer Faktenmenge die Wissensbasis eines rechnergestützten Systems zur Führung komplexer Prozesse dar.

2.2.1 Allgemeine wissensbasierte Verfahren

Die KI-orientierten Formen beschreiben (im Idealfall) nur die betrachteten Objekte und deren Zusammenhänge, die Verarbeitungsreihenfolge ergibt sich direkt (implizit) aus den daraus resultierenden gültigen Beziehungen.

Bei den nicht-konventionellen Verarbeitungsmodellen kann in folgende Arten unterschieden werden:

- Logik
  - Aussagenlogik,
  - Prädikatenlogik,
  - Modallogik (allgem. mehrwertige Logiken),
  - Fuzzy-Logik,
  - Temporale Logik.

- Rahmen
  - Semantische Netze,
  - Frames,
  - Klassen (objektorientiert),
  - Skripten.

- Regeln
  - einfache Regelform,
  - probabilistisches Schließen (mit Wahrscheinlichkeiten),
  - nichtmonotones Schließen (ohne Konfluenz, mit Rücknahmen).

- qualitatives / algebraisches Schließen
  - quantitatives Schließen mit Werten, Wertemengen (Intervalle),
  - qualitatives Schließen mit relationalen Operatoren,
  - Computeralgebra und Termersetzungsverfahren.

- Lernverfahren (symbolische, konnektionistische/neuronale).

- kognitive Modelle.

Das Verarbeitungsmodell einer Sprache implementiert den Kalkül des abstrakten Formalismus. Im Idealfall entspricht das Verarbeitungsmodell einer Sprache dem Kalkül. Im Normalfall ist jedoch die Implementierung eines Verarbeitungsmodells nicht ideal, sondern verwendet bestimmte Verfahren um die mehrdeutigen Abhängigkeiten zwischen den zu verarbeitenden Aussagen aufzulösen.

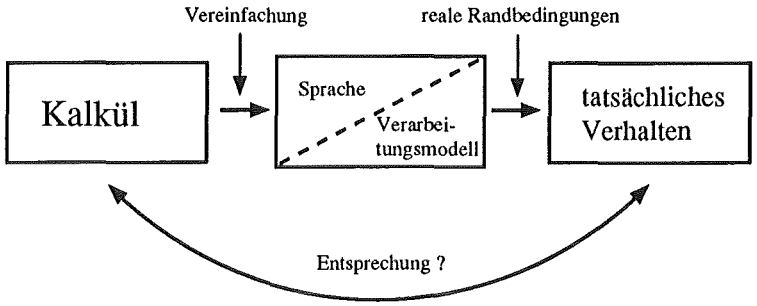


Abbildung 2

Da diese Modelle auf klassische Rechner abgebildet werden, basieren sie letztlich auch auf dem imperativen Modell (ein Regelinterpreter kann nicht durch sich selbst realisiert werden). Außerdem sind zur Komplexitätsreduzierung bzw. zur Entscheidbarkeit (z. B. Regelkonflikt) immer heuristische Methoden (Prozeduren) notwendig (z. B. Breitensuche, Tiefensuche, Verzweigen und Begrenzen etc. /16/). Diese heuristischen Komponenten der realisierten Verarbeitungsmodelle wirken auf die Auswertungen und ihre Ergebnisse zurück.

Die Diskrepanz zwischen einem formalen Kalkül und dem tatsächlichen Verhalten läßt sich am Beispiel PROLOG (PROgramming in LOGic) sehr gut darstellen. Der zugrundeliegende Kalkül ist der *Hornklauselkalkül* (ein Derivat der Prädikatenlogik erster Ordnung). Bereits bei der Auswahl des Hornklauselkalkül, dieser ist wiederum ein Derivat des Resolutionskalkül, spielte der Verarbeitungsaspekt eine wesentliche Rolle, so daß die Vereinfachung im wesentlichen schon hier erfolgte. Die wesentlichen Verarbeitungsmechanismen sind die *Rückwärtsverkettung von Regeln* als Ableitungsmechanismus und die *Unifikation* um Gleichheit zwischen den durch Konjunktion verbundenen Komponenten der Hornklauseln herzustellen. Realen Randbedingungen, wie z. B. eine kombinatorische Explosion bei der Anwendbarkeit einer zu großen Regelmenge und ein damit verbundener Stack overflow, kann nur zum Teil durch Kontrollwissen, das in der Programmsequenz kodiert wird (*Rechtsrekursion*) begegnet werden; d. h. es bestehen zumindest Restriktionen, die von der Rechnerkapazität abhängig sind (/26/).

In der Logik unterscheidet man zwischen statischer und dynamischer Logik. Die statische Logik bezieht sich auf die rein mathematische/logische Beschreibung, die dynamische Logik bezieht sich auf die ausführungsorientierte Form mit ihren heuristischen Komponenten. Die Reihenfolge der Eingabe bei Produktionensystemen oder bei der prädikatenlogik-orientierten Sprache PROLOG (siehe oben) beeinflußt die Auswertung und das Ergebnis bis hin zu einer fehlenden Terminierung, was einer Nichtentscheidbarkeit des Problems gleich kommt.

Bei den konnektionistischen Systemen ist anzumerken, daß sie ausschließlich Mustervergleiche und keine Erkennung im Sinne inhaltlicher Begriffszuordnung durchführen. Erst eine Erweiterung konnektionistischer Systeme um symbolische Verarbeibeitungsmethoden erlaubt dies. Systeme, die konnektionistische und symbolische Verarbeitungsmodelle miteinander verbinden, werden als "hybrid" bezeichnet.

Im Rahmen des Projektes INPRO wird im Bereich der lernenden Verfahren über kognitionstheoretische Konzepte versucht ein Formalismus des Lernens abzuleiten. Die Basis sind kognitive Schemata, Metawissen Reflexion und ähnliches. Für sehr vereinfachte Problemstellungen (Interpolation bei gegebenem Ziel und Ausgangsstellung) existieren solche Formalismen (Situationskalkül), eine umfassende Theorie ist jedoch erst in Ansätzen erkennbar (wenn überhaupt).

2.2.2 Technisch orientierte wissensbasierte Verfahren

Von den wissensbasierten Methoden finden in technischen Systemen im wesentlichen die folgenden Verfahren Anwendung (/16/):

(1) Regelorientierte Verfahren,

(2) logische Verfahren,

(3) konnektionistische / neuronale Verfahren,

(4) elementare symbolische Lernverfahren.

Allen Verfahren liegt der (teils idealistische) deklarative Ansatz zugrunde, daß der Mensch nur das "was" beschreibt und das "wie" sich implizit, automatisch durch den zugrundegelegten Formalismus, d. h. die Verarbeitungsstrategie, ergibt.

Bei **regelorientierten Verfahren** beschreibt der Mensch die Aktionen, die bei bestimmten Bedingungen ausgeführt werden, das Schlußfolgerungssystem (der Regelinterpreter) berechnet aufgrund der vorliegenden Situation die auszuführenden Regeln und führt diese dann auch aus.

Im allgemeinen werden sprachliche Beschreibungen von Situationen vorliegen, die nicht direkt in Zahlenwerte umsetzbar sind. Solche Formuliereungen werden als vage, unscharf bezeichnet und sind z. B. über die Methoden der **Fuzzy Logic** darstellbar und ausführbar (siehe Abschnitt 5.4).

Die Schwierigkeit, eine Situation zur erkennen, oder einer Situation entsprechende Handlungen zuzuorden, hat zur Entwicklung **neuronaler Systemen** geführt. Die herausragenden Fähigkeit neuronaler oder konnektionistischer Systeme besteht darin, daß sie mit Hilfe einer netzähnlichen Struktur unter Verwendung von Gewichten eine Zuordnung von Elementen einer Zielmenge zu Elementen einer Eingabemenge vornehmen können. Dabei können die Gewichte anhand ausgewählter Trainingssequenzen über Rückkopplungsmechanismen automatisch gelernt werden. Da der Zuordnungsprozeß über dieser netzähnlichen Struktur immer von einer großen Anzahl an Gewichten beeinflußt wird, d. h. das "Entscheidungswissen" ist verteilt, sind neuronale Systeme sehr Fehlertolerant, d. h. auch gestörte Eingabemuster können dem "richtigen" Ausgabemuster zugeordnet werden; es wird nach dem größtmöglichen Ähnlichkeitsmaß zugeordnet. Dem Einsatz in der Prozeßführung steht allerdings die Schwierigkeit gegenüber, daß ein neuronales Netz im allgemeinen nicht erklären kann, anhand welcher Merkmale es die Zurordnung getroffen hat. Das Wissen eines neuronalen Systems ist implizit (in den Gewichten verteilt) und nicht explizierbar.

Symbolische Lernverfahren können bei vorgegebenen Primärattributen anhand von Trainingsbeispielen semantische Kategorien lernen und Eingaben entsprechend klassifizieren. Die Ergebnisse (Klassenzuordnungen) sind erklärbar. Transformationsfolgen im Sinne von Zeitreihen oder Situationsabläufen sind jedoch schwierig lernbar.

Der Einsatz zur Prozeßführung erfordert die unbedingte Erklärbarkeit der Schlußfolgerungen symbolischer Systeme, die Berücksichtigung von Echtzeitanforderungen

(eine Schlußfolgerung muß in einer definierten Zeitspanne berechnet sein) und die automatische Anpassung der Wissensbasis im Betrieb (Validierung, Erweiterung).

2.2.3 Das Problem der Wissensakquisition

Als Problem stellt sich bei wissensbasierten Verfahren allerdings, die Wissensakquisition und -validierung (z. B. selbstverständliche Annahmen des Operateurs) dar. Mit Methoden des automatischen Lernens können heuristische Beziehungen (z. B. eine Situation im Prozeß und der zugeordneter Eingriff des Operateurs) automatisch in die Wissensbasis übernommen und im Bezug auf früher erkannte Beziehungen validiert werden.

In wissensbasierten Systemen und speziell in Expertensystemen bildet die Akquisition und die Validierung von Wissen einen komplizierten Prozeß. Für diese Tatsache sprechen beispielsweise die folgenden Aussagen, (/26/):

- Experten können häufig ihr Wissen nicht in Form von Regeln ausdrücken und formulieren.
- Auch hochqualifizierte Experten wissen nicht immer exakt, wie ein Problem zu lösen ist. Sie stützen sich daher auf Hypothesen, die sie im Verlauf der Problemlösung evaluieren.
    - Die Darstellung solch vagen, d. h. unsicheren und unvollständigen Wissens ist nicht unproblematisch.
- Experten vergessen häufig viele wichtige Faktoren zu nennen, da sie diese für selbstverständlich halten.
- Die Ermittlung von Expertenwissen ist ein sehr zeitaufwendiger Vorgang.

Aufgrund dieser Fakten wird die Wissensakquisition häufig als Flaschenhals der Expertensystemtechnologie bezeichnet, (/13/), und man versucht diesem potentiellen Problem aus dem Weg zu gehen, indem das notwendige Wissen automatisch akquiriert wird. Aber selbst wenn man diesen Schritt bewältigt hätte, so könnte man keineswegs zufrieden sein. Eine automatische Wissensakquisition würde nämlich das Problem der Validierung des automatisch aufgenommenen Wissens nach sich ziehen, d. h. der Flaschenhals wäre nun auf Seiten der Validierung des Wissen und man hätte nur wenig gewonnen.

Folglich kann eine Lösung dieser Probleme nur darin bestehen, daß das Wissen im System sowohl automatisch akquiriert, als auch automatisch validiert wird. Ein grundsätzlich neuer Ansatz zur Lösung dieses Problemes wird im Abschnitt 8.3 vorgestellt.

3 Wissensbasierte Prozeßführung

Die Störungsfrühdiagnose dient dazu, sich anbahnende Störungen im Prozeß frühzeitig zu erkennen und die Störungsursache zu diagnostizieren, um durch rechtzeitiges gezieltes Eingreifen größere Sollabweichungen zu verhindern. Im Gegensatz zur herkömmlichen Fehlerdiagnose, die erst nach dem Auftreten einer Störung diese analysiert, besitzt die Frühdiagnose die Fähigkeit, bereits Entwicklungen und Trends, die möglicherweise auf Prozeßstörungen hinauslaufen, zu behandeln.

3.1 Allgemeine Störungserkennung und -diagnose

Im Bereich der Prozeßführung und der Maschinensteuerung werden seit einiger Zeit Methoden zur Erkennung und Diagnose von Störungen eingesetzt. Dabei handelt es sich in der Regel um Methoden zur nachträglichen Diagnose von eingetretenen oder zur frühzeitigen Erkennung und parametrischen Zuordnung eintretender Störungen. In /21/ werden existierende Verfahren hierfür untersucht.

Die nachträgliche Diagnose erfolgt anhand der vorliegenden Meßwerte und unterstützt die Bediener bei der Ursachenforschung. Die Methoden basieren teils auf analytischer, teils auf heuristischer Beschreibung und verwenden Verarbeitungsmodelle der Logik, der qualitativen Analyse oder regelorientierte Formen. Die Verfahren mit einer analytischen Beschreibung beschränken sich, aus den bereits dargestellten Gründen, auf einzelne Apparate oder deren Komponenten, siehe Abb. 3.

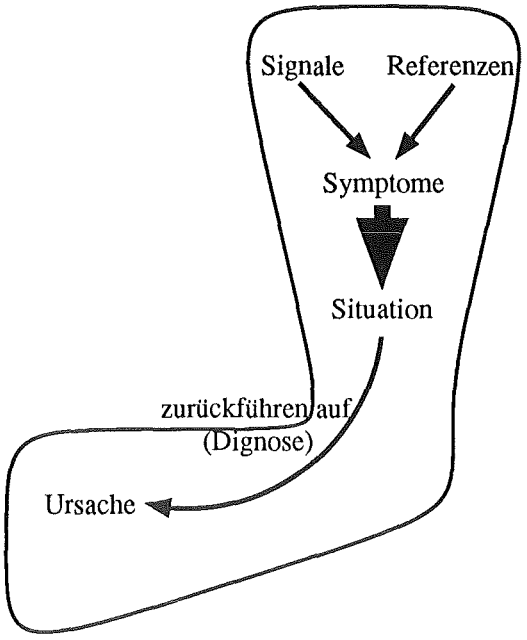


Abbildung 3

Beim direkten on-line Einsatz werden parametrische Modelle (analytisch) verknüpft mit heuristischen Erkenntnissen zur Fehlerzuordnung verwendet. Bei einfachen Sachverhalten

sind auch Methoden der Fuzzy Logik oder Fehlerbaumanalysen einsetzbar. Die parametrischen Modelle beschränken sich auch hier nur auf einzelne Apparate oder deren Komponenten. Die Methoden aus dem Bereich der Prozeßdatenverarbeitung finden hier noch keinen Eingang. Eine echte Verknüpfung verschiedener Ansätze zusammen mit einer Strukturierung nach anwendungsorientierten Kriterien findet nicht statt, oder ist aufgrund der Einfachheit nicht notwendig.

Diese Formen einer Störungsdiagnose oder -erkennung sind nicht vollständig in ein Konzept einer wissensbasierten Prozeßführung eingebunden. Eine Überprüfung von abgeleiteten Maßnahmen in bezug zur Ausgangssituation findet nicht statt oder ist aufgrund der einfachen, eng begrenzten Betrachtung bei parametrischen Verfahren nicht notwendig.

Diese Art einer Störungs(-früh-)diagnose wird als "passive SFD" bezeichnet und basiert auf mathematischen Modellen und Parameterbewertungen. Sie dient nicht einer kontinuierlichen Prozeßführung, sondern stellt einen diskreten Bewertungsvorgang dar.

3.2 Wissensbasierte Störungsfrühdiagnose

Komplexe Problemstellungen sind mit den oben genannten Verfahren nur schlecht zu bearbeiten, da eine Strukturierung nach bedeutungsorientierten Merkmalen (Prozeßtopologie, Komponenten, semantischen Bezügen, abstrakten/konkreten sinnorientierten Operationen, Situationen und deren Verallgemeinerungen oder Spezialisierungen) nicht möglich ist und zudem der Echtzeitcharakter der Problemstellung vernachlässigt wird.

Eine Integration von regelorientierter Beschreibung und Verarbeitung zur Modellierung heuristischen Wissens mit dem analytischen Wissen in Form von mathematischen Prozeßmodellen (qualitativ, quantitativ) erfolgt ebenfalls nur selten.

Eine echte Einbindung von Methoden der Störungsfrühdiagnose erfordert eine Supervisor-Ebene zur Überwachung aller Operationen und Ergebnisse. Eine aktive, wissensbasierte Störungsfrühdiagnose deckt den Pfad vom technischen Prozeß bis zur Ursachenbestimmung ab und beschränkt sich nicht auf parametrische oder rein heuristische Formen. Sie integriert diese beiden Formen und ermöglicht darüber hinaus auch eine symbolische Verarbeitung von funktionalen Zusammenhängen als weitere Möglichkeit, einzelne Ursachen in einer kausalen Kette von Ursachen zu ermitteln.

Graphisch kann man die Erweiterung der passiven SFD zur aktiven SFD nach Abb. 4 darstellen.

Die wesentlichen Unterschiede zwischen der passiven- und der aktiven Störungsfrühdiagnose sind:

- Bei der aktiven Störungsfrühdiagnose werden nicht nur Ursachen gefunden, sondern es wird aktiv, durch eine oder mehrere Therapien, an der Beseitigung der Störung gearbeitet.

- Durch eine Validierung der Diagnose erfolgt eine Bewertung des Therapieerfolges.
- Damit bildet die aktive SFD einen bewertenden Zyklus.

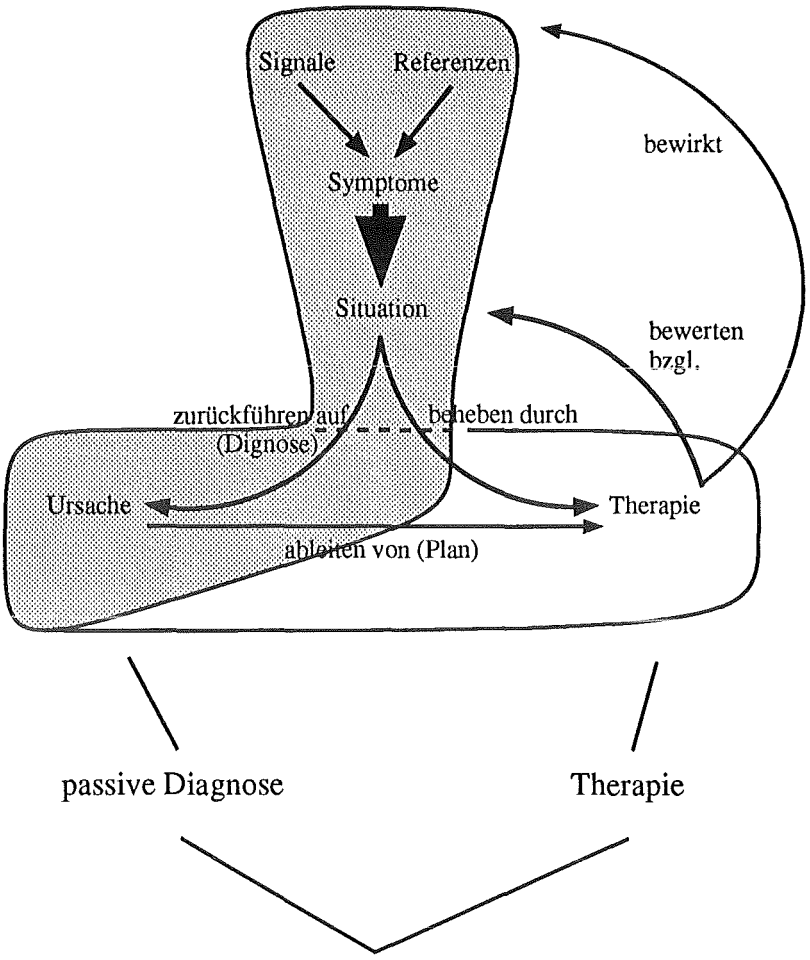


Abbildung 4

Gerade im Bereich der Prozeßführung komplexer technischer Anlagen ist eine Bewertung von Störungsfolgen bzgl. deren Wichtigkeit und gegenseitigen Abhängigkeit nötig. Dringlichere Folgestörungen müssen vorrangig bedient werden, disjunkte Störungen sind gleichzeitig zu bearbeiten.

3.3 Anwendung neuronaler Netze

Im Bereich der Prozeßführung können neuronale Netze eingesetzt werden, um von einem menschlichen Bediener in bestimmten Prozeßsituationen ausgeführte Handlungen zu lernen und selbständig zu reproduzieren. In der Lernphase ist es notwendig, die dem Bediener

zugänglichen Informationen dem neuronalen Netz als Eingangsgrößen und die Bedienereingriffe als zu erzeugende Ausgangsgrößen einzugeben.

Eine gute Lernbarkeit, bzw. ein gutes Konvergenzverhalten und eine anschließend gute Interpolation dürfte durch ein topologieerhaltendes Netz erreicht werden können. Bei dynamischen Prozessen leitet sich der Bedienereingriff auch von der Historie ab, deshalb sind Situationsfolgen (Zeitreihen von t_k bis t_{k-i}) oder Änderungsgrößen (d/dt) einzuspeisen. Die Struktur, um Bedienerhandlungen bei bestimmten Prozeßsituationen zu lernen, zeigt Abb. 5.

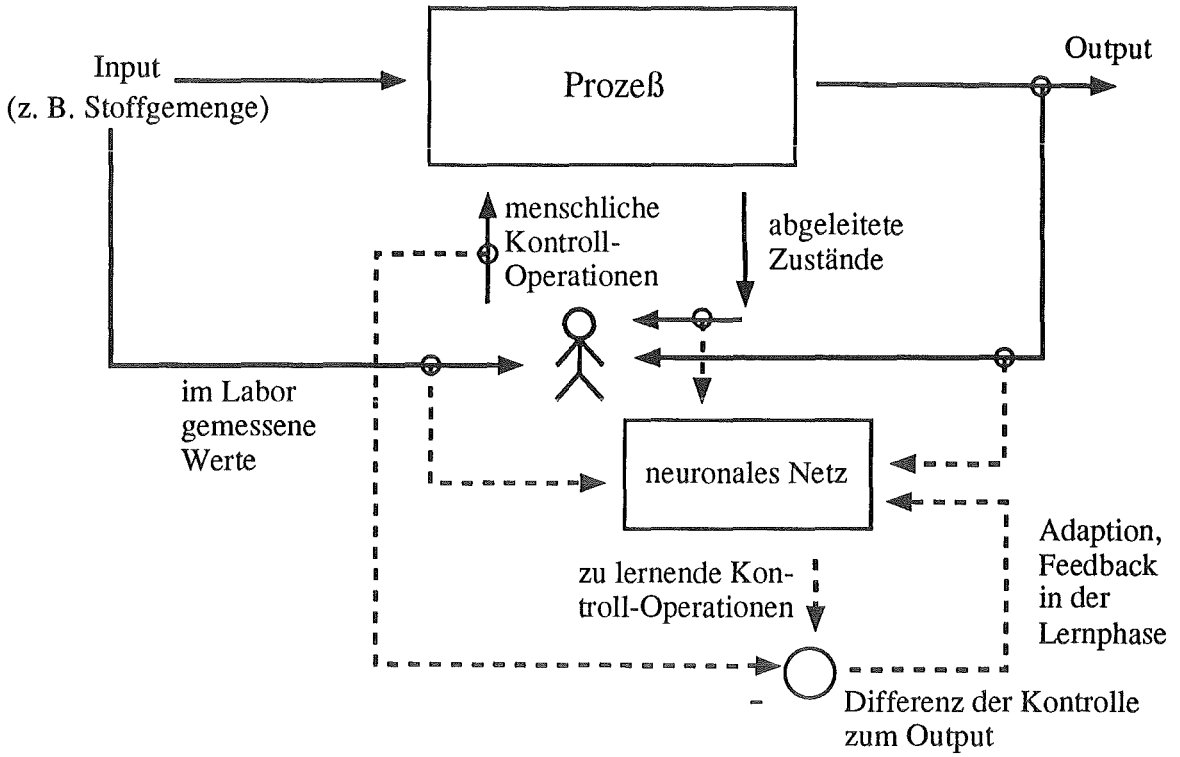


Abbildung 5

Neuronale Netze können neben dieser Art der Anwendung auch eingesetzt werden bei:

- Zuordnung von Prozeßsituationen und Prozeßeingriffen zu späteren Analysewerten (z. B. Dioxin in der Müllverbrennung).
- Optimierung der Prozeßeingriffe durch eine Gewichtsanalyse (partielle Abhängigkeit der Ausgänge von den Eingängen).
- Analyse der Gewichte bzgl. der Übertragbarkeit in Regelform (Fuzzy-Regeln).
- Nachbildung des dynamischen Verhaltens eines dynamischen Systems und der Möglichkeit Eingriffe durchzuführen (Simulator).

4 Berücksichtigung von analytischem Wissen

4.1 Elementare Verfahren

Die elementaren Verfahren kann man nach statischer und dynamischer Bewertung der einzelnen Prozeßgrößen unterscheiden.

Bei der Überwachung von technischen Prozessen werden die direkt meßbaren Größen auf Überschreitung oder Unterschreitung von vorgegebenen Grenzwerten (Absolutwertkontrolle) hin kontrolliert. Dadurch können zwar einige Fehler im Prozeß erkannt werden, dies ist aber erst möglich, nachdem sie sich bereits auf die meßbaren Ausgangsgrößen ausgewirkt haben.

Eine Über- bzw. Unterschreitung des Meßwertes bzgl. vorgegebener Grenzwerte ist im allgemeinen bei solchen Systemen als Aufforderung zur Beurteilung der aktuellen Prozeßsituation, Fehlerlokalisierung und -behebung durch das Bedienpersonal zu sehen. Werden die Grenzwerte bei der Kontrolle von Prozeßsignalen nach der sicheren Seite hin festgelegt, erhält man damit mehr Zeit für Gegenmaßnahmen, dies kann aber zu häufigen Störmeldungen führen. Diese sind dann unnötig, wenn die betreffende Größe ohne äußeren Eingriff wieder in den Normalzustand zurückkehrt.

Dieser Nachteil läßt sich beheben, wenn man das betreffende Signal vorhersagen kann (Trendberechnung). Wendet man die Absolutwertkontrolle auf das vorhergesagte (geschätzte) Signal an, dann kann man einerseits früher Grenzwertüberschreitungen erkennen und andererseits unnötige Störmeldungen vermeiden, wenn die betrachtete Größe ohne zusätzliche Maßnahmen wieder in den Normalzustand zurückkehrt.

Eine etwas weitergehenderere Möglichkeit ist, die Unterscheidung zwischen zulässigen und nichtzulässigen Prozeßbereichen für die jeweiligen Prozeßgrößen, wobei der zulässige Wertebereich nochmals in einen normalen Arbeitsbereich und in mehrere Grenzbereiche unterteilt werden kann. Mit Hilfe dieser Werte läßt sich das Verhalten einer Größe bzgl. der Bereiche sowohl statisch als auch dynamisch differenzierter bewerten.

Über Bilanzierungsrechnungen können des weiteren auch Stoffverteilungen o. ä. in größeren Prozeßbereichen verfolgt werden.

4.2 Modellbasierte Verfahren

4.2.1 Numerische modellbasierte Verfahren

4.2.1.1 Modellgestützte Messung

Modellgestützte Meßverfahren dienen zur Bereitstellung von Prozeßkenngrößen, die ähnlich wie die Meßgrößen die Kenntnis des aktuellen Prozeßzustands verbessern. Man unterscheidet

- Zustandsschätzverfahren (Beobachter) und
- Parameteridentifikationsverfahren.

Bei der Zustandsschätzung besteht die Aufgabe darin, unzugängliche innere Zustandsgrößen eines Prozesses aus den vorliegenden Meßgrößen zu berechnen. Zu diesem Zweck werden Gleichungen eines geeigneten Prozeßmodells zusammen mit einem Korrekturterm (z. B. Luenberger-Beobachter) in Echtzeit parallel zum Prozeß gelöst. An diesem Modell können die interessierenden Zustandsgrößen abgegriffen werden. Die Struktur eines Beobachters ist in Abb. 6 dargestellt.

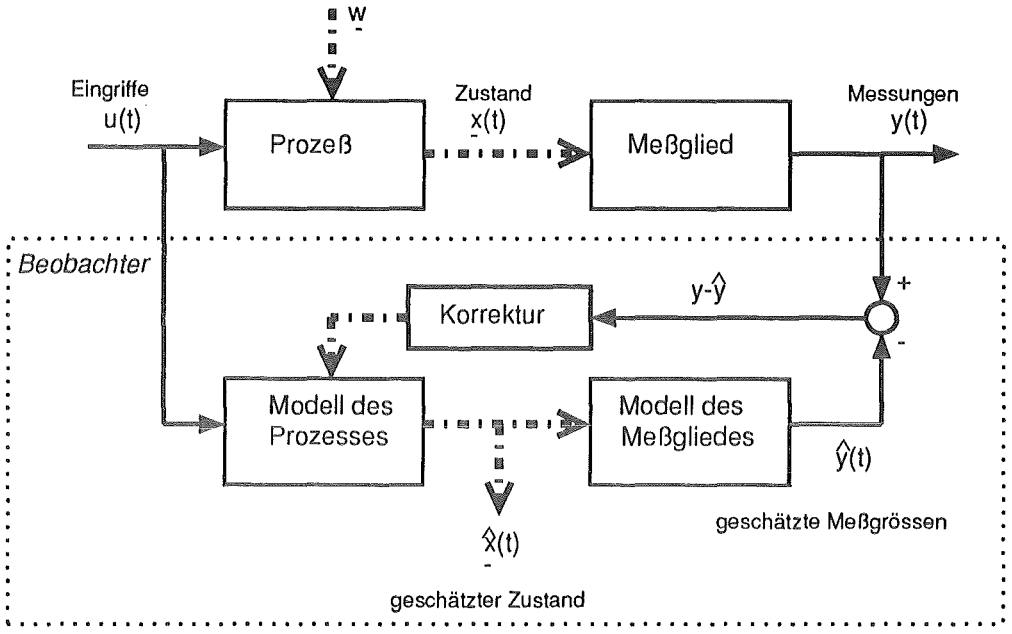


Abbildung 6

Das Parameteridentifikationsverfahren wird dann eingesetzt, wenn die Parameter eines Prozesses, z. B. für eine Adaption von Reglerparametern oder bei der "Fehlerdiagnose mittels Parameterschätzung" ständig bestimmt werden sollen.

Auf der Basis dieser Größen (Zustände, Modellparameter) kann dann eine weitere Bewertung erfolgen.

4.2.1.2 Modellgestützte Diagnose

Eine modellgestützte Prozeßdiagnose zur Erkennung von kritischen Zuständen erfolgt mit Hilfe modellgestützter Meßverfahren.

Hierzu sind im wesentlichen zwei klassische Methoden zu nennen:

- Parameterschätzung zur Diagnose
- Fehlererkennung durch parallele Simulation mehrerer Prozeßmodelle (Multi-Modell-Diagnose-System).

Der Grundgedanke bei der ***"Parameterschätzung zur Diagnose"*** besteht darin, daß viele Prozeßfehler als Änderungen von Prozeßparametern (z. B. Widerstände, Dämpfung, Wärmeübergangszahlen) erscheinen. Diese Prozeßparameter sind in den Parametern eines Prozeßmodells enthalten (Prozeßmodellparameter).

Im Prozeß bezeichnet man als *Prozeßmodellparameter* diejenigen Größen, die bei der mathematischen Beschreibung des Zusammenhangs von Ein- und Ausgangsgrößen als Konstanten auftreten; z. B.

$$y(t) + a_1\, y'(t) + a_2\, y''(t) + \ldots + a_n\, y^n(t) = b_0\, u(t) + b_1\, u'(t) + b_2\, u''(t) + \ldots + b_m\, y^m(t)$$

a_i, b_i = Prozeßmodellparameter M_p.

Die Vorgehensweise besteht nun darin, daß zunächst die Prozeßgleichung für die meßbaren Ein- und Ausgangsgrößen

$$Y(t) = f(U(t), M_p)$$

auf theoretischen oder experimentellen Wege aufgestellt wird. Es werden dann die Prozeßmodellparameter M_p auf der Basis von Messungen der Signale Y(t) und U(t) geschätzt. Diese Prozeßmodellparameter gelten für den fehlerfreien Fall. Anhand der Ein- und Ausgangs- größen werden dann im On-line-Betrieb die Prozeßmodellparameter ermittelt und mit den Normalwerten verglichen. Aus einem Fehlerkatalog, in dem der Zusammenhang zwischen Prozeßfehlern und Änderungen der Prozeßmodellparameter festgelegt ist, kann dann auf eventuelle Prozeßfehler zurückgeschlossen werden (Mustererkennung).

Diese Methode setzt jedoch voraus, daß Störungen signifikant als Änderungen in den Prozeßmodellparametern erscheinen. Ob dies der Fall ist oder nicht, kann aus dem a-priori Wissen abgeleitet werden. Eine weitere Bedingung ist eine genügend stark angeregte Prozeßdynamik (Änderungen an den Ein- und Ausgängen).

Im wesentlichen sind es drei Punkte, die bei dieser Methode beachtet werden müssen:

- Es muß ein analytisches Prozeßmodell (gleichungsorientiert) vorhanden sein,
- es müssen leistungsfähige Parameterschätzmethoden existieren und
- es bedarf einer genügend stark angeregter Prozeßdynamik.

4.2.1.3 Beispiel

Ein in /11/ beschriebenes Fallbeispiel wird hier verkürzt wiedergegeben. Es handelt sich um einen Wärmeaustauscher (schematische Darstellung siehe Abb. 7).

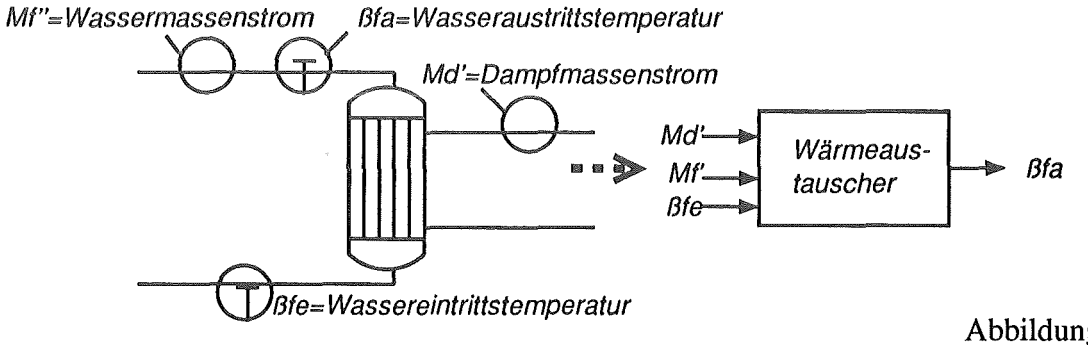


Abbildung 7

Für den Wärmeaustauscher wurde ein vereinfachtes Modell erstellt (Übertragungsfunktionen, konzentrierte Parameter). Im Modell wird *ßfa* als Ausgangsgröße betrachtet, die anderen Größen sind Eingangsgrößen. Für jede Eingangsgröße wurde die Übertragungsfunktion erstellt. Für den Dampfstrom als Eingangsgröße ergab sich beispielsweise:

$$F_{Md'(s)} = \frac{\triangle ßfa(s)}{\triangle Md'(s)} = \frac{K_d}{(1+T_{1d}\,s)\,(1+T_{2d}\,s)}\; e^{-T_{td}\,s}$$

Diese Übertragungsfunktion wurde noch wie folgt vereinfacht (da kleines T_{td}).

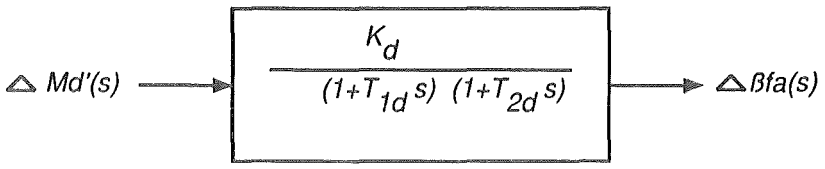


Abbildung 8

Die Modellparameter sind demnach K_d, T_{1d}, T_{2d}. Die Prozeßkoeffizienten wurden nicht betrachtet. Es wurden zunächst die Modellparameter aufgrund von Übergangsfunktionsmessungen der Fluidaustrittstemperatur *ßfa* nach Änderungen der Eintrittstemperatur *ßfe* des Dampfstromes Md' und des Fluidmassenstromes Mf' geschätzt. Als nächstes wurden die Modellparameter für vier künstlich erzeugte Fehler ermittelt. Dann wurden die Modellparameter im Normalfall mit denjenigen im Störungsfall verglichen. Die Fehler bewirkten eine Änderung der Modellparameter bei allen drei Übetragungsfunktionen.

Das Änderungsmuster zum Normalfall bei der Übetragungsfunktion sah wie folgt aus (gleich: =, kleiner: <, sehr viel kleiner: <<, usw.):

Fehler	K_d	T_{1d}	T_{2d}
F1	>	<<	=
F2	<<	<<	>
F3	>	>	<<
F4	=	>	<

Bei der "***Fehlererkennung durch parallele Simulation mehrerer Prozeßmodelle***" wird ein Modell des bestimmungsgemäßen Normalbetriebes mit Modellen, in denen einzelne relevante Fehlzustände modelliert sind, parallel zum Prozeß betrieben. Durch eine statistische Analyse der Zustandsschätzergebnisse, läßt sich dasjenige Modell bestimmen, welches das wirkliche Verhalten am besten beschreibt.

Es wird angenommen, daß sich der Prozeß in demjenigen Betriebszustand befindet, dessen Modell die beste Übereinstimmung mit der Wirklichkeit besitzt.

4.2.1.4 Constraint-Techniken

Die Massen und Energiebilanzen, Änderungsgleichungen und Gleichgewichtsbeziehungen in einem Prozeß liefern eine Menge von Randbedingungen an die Werte der Prozeßvariblen. Bei Constraint-Techniken werden signifikante Verletzungen solcher Randbedingungen als Indikator von Prozeßfehlern herangezogen.

Die Methode basiert auf der Verknüpfung jeder quantitativen Randbedingung in einem Prozeß mit einer Menge von Fehlern, die Ursache der Verletzung dieser Randbedingung sein können. Die Klassifikation, ob Verletzungen von Randbedingungen (constraint violations) vorliegen erfolgt über boolesche Techniken. Der Ansatz beruht auf kausalem (analytischem) Wissen (vgl. /8/).

4.2.2 Symbolische modellbasierte Verfahren

4.2.2.1 Algebraische Verarbeitung, Termersetzungsverfahren

Ein Gebiet, das sich mit algorithmischen Verfahren zur Behandlung von Problemen, deren Angaben und Lösungen algebraische Objekte sind, befaßt ist die *Computer-Algebra* (symbolic oder algebraic computation, Formelmanipulation), z. B. /3/.

Typische Problemklassen, die mit der Computer-Algebra behandelt werden können sind:

- Die Darstellung der Objekte algebraischer Bereiche im Computer, insbesondere die symbolische Vereinfachung (Simplifikation) von möglichen Darstellungen eines Objektes auf eine Standardform.
- Die Konstruktion von Algorithmen, mit denen die Grundoperationen auf den Objekten im Computer ausgeführt werden sollen.
- Die Übersetzung von verschiedenen Darstellungen ein und desselben Bereiches ineinander.
- Die Dekomposition von Objekten in einfache Objekte, bzw. die Frage nach der Dekompositionierbarkeit.
- Das Auffinden gemeinsamer "Teil-" oder "Oberobjekte" gegebener Objekte.
- Die exakte Lösung von Gleichungen oder Ungleichungen.

4.2.2.2 Kausale Verarbeitung (Ursache-Wirkungs-Beziehungen), qualitative Verarbeitung

Wenn ein System- oder Prozeßverhalten nicht mit Hilfe eines exakten formalen mathematischen Modells beschrieben werden kann oder soll (z. B. aus Kosten-, Komplexitätsgründen oder da es sich um einen ersten Schritt im Zuge eines Analyseprozesses handelt) besteht im Rahmen eines symbolischen modellbasierten Verfahrens die Möglichkeit die prozeß- oder systemimmanenten kausalen Zusammenhänge in Form von Ursache-Wirkungs-Beziehungen darzustellen.

Beispielsweise würde ein lineares Systemverhalten, das durch die Gleichung

$$\frac{d}{dx} = a \cdot x + b, \qquad a > 0$$

exakt beschreibbar ist in Form von symbolischen Ursache-Wirkungs-Beziehungen wie folgt beschrieben:

$$\begin{aligned} &\text{Für } x \to -\infty \Rightarrow d/dx \to -\infty, \\ &\text{für } x \to 0 \Rightarrow d/dx \to b, \\ &\text{für } x \to \infty \Rightarrow d/dx \to \infty. \end{aligned}$$

Anhand solcher kausaler Beziehungen können aufgrund des qualitativen Verhaltens einzelner System- oder Prozeßkomponenten die funktionalen Eigenschaften eines System oder Prozesses erschlossen werden (vgl. /5/).

5 Berücksichtigung von explizitem heuristischem Wissen

5.1 Regelorientierte Verarbeitung

Eine nichtklassische Form der Wissensverarbeitung, die auch heuristisches Wissen berücksichtigt ist die Verarbeitung in regelorientierten Systemen. Den Kern regelorientierter Systeme bilden WENN-DANN-Regeln. Die WENN-Teile (*Antezedens*) solcher Regeln beinhalten Kombinationen bekannter Fakten (bzw. allgemeiner Bedingungen), die DANN-Teile (*Konsequenz*) geben neue Fakten (bzw. allgemeiner Aktionen) an, die direkt von den bekannten Fakten (deduktiv) abgeleitet werden können.

Durch einen Ableitungsmechanismus wird ein regelorientiertes System zu einer Art *Deduktionssystem*; allgemeiner spricht man von einem System bestehend aus Bedingungs-Aktionen-Regeln, (/32/).

Wenn alle Bedingungen einer Regel durch die aktuelle Situation erfüllt sind, so sagt man die Regel ist ausgelöst (*triggered*). Wenn die Aktionen ausgeführt werden, so sagt man von der Regel, daß sie gefeuert hat (*fired*). Auslösen bedeutet nicht immer Feuern, denn die Bedingungen mehrerer Regeln können gleichzeitig erfüllt sein. In solchen Fällen muß eine *Konfliktlösungsstrategie* entscheiden, welche Regel tatsächlich feuern soll. Dies geschieht z. B. durch Bewertung der anwendbaren Regeln und deren anschließende Anordnung in einer Prioritätenliste.

Um *Ableitungen über mehrere Schritte* durchführen zu können, werden Ableitungsketten gebildet. Grundsätzlich unterscheidet man zur Bildung und Abarbeitung solcher Ableitungsketten zwei verschiedene Strategien, die *Vorwärts-* und die *Rückwärtsverkettung*. Diese beiden Strategien sind prinzipiell gleichwertig und gleichmächtig. Daher hängt eine Entscheidung für die eine oder andere Strategie sehr stark von den jeweiligen Gegebenheiten ab.

In regelorientierten Systemen wird das Problem und nicht der Lösungsweg beschrieben. Regelorientierte Systeme erleichtern die Beantwortung von Fragen zu ihrem Verhalten, d. h. sie können ihre Vorgehensweise bis zu einem gewissen Grad begründen.

Aus den oben bereits angeführten Gründen besteht ein regelverarbeitendes System, ein sogenanntes **Produktionssystem** im wesentlichen aus drei Komponenten:

- Den Produktionsregeln,
- einem Regelinterpreter zur Auswahl und Auswertung der Produktionsregeln,
- einer Erklärungskomponente.

Dem Vorteil des modularen Aufbaus und damit der leichten Erweiterbarkeit der Regelmenge steht als großer Nachteil deren Unstrukturiertheit gegenüber. Diese Unstrukturiertheit hat zur Auswirkung, daß bei einer großen Anzahl an Produktionsregeln die Regelauswahl und/oder die Regelauswertung schnell zu einer kombinatorischen Explosion führt.

5.2 Objektorientierte Ansätze

Der objektorientierte Ansatz verspricht für die Software-Entwicklung in den neunziger Jahren ähnliche Bedeutung zu erlangen, wie die strukturierte Programmierung im vergangenen Jahrzehnt.

Es werden zum einen technische Konzepte angeboten, die die Erweiterbarkeit und Wiederverwendung von Softwarebausteinen unterstützen, und zum anderen ein Vorgehen beim Entwurf vorgeschlagen, das es gestattet, Struktur und Komplexität des Problembereiches in "natürlicher" Weise auf die Software zu übertragen; einfache Änderungen im Anwendungsgebiet bleiben auch im Softwaresystem einfach.

Der objektorientierte Ansatz verbindet bewährte Prinzipien der Softwareentwicklung wie "abstrakte Datentypen" und "Information Hiding" mit den weniger bekannten Konzepten "Vererbung" und "dynamisches Binden" und kombiniert diese mit einer anwendungsnahen Sicht der Welt zu einem konsistenten und durchgängigen Verfahren, (/27/).

Im objektorientierten Ansatz ist es durch das Klassenkonzept und das Konzept der Vererbung möglich Gemeinsamkeiten, Verallgemeinerungen, aber auch Spezialisierungen von Komponenten mit programmiersprachlichen Mitteln zu beschreiben. Daraus ergibt sich langfristig ein Produktivitätsgewinn durch Einsparung an Entwicklungszeit und -kosten.

Darüber hinaus wird durch die Verwendung von standardisierten und ausgetesteten Software-Bausteinen das zu erstellende Softwareprodukt sicherer und qualitativ hochwertiger sowie der Entwicklungsaufwand geringer.

Die konsequente Anwendung der Konzepte der objektorientierten Programmierung verspricht die Erstellung von klar strukturierten, sicheren, änderbaren, erweiterbaren und leicht wartbaren Systemen, wobei vorhandene Software-Bausteine auf einfache Weise wiederverwendet werden können.

Die Grundlage für ein Verständnis der Objektorientierung und ihres Einsatzes ist ein konsistenter Satz von Basiskonzepten. Diese werden im folgenden kurz vorgestellt und erläutert.

5.2.1 Die Basiskonzepte objektorientierter Programmierung

Bei der objektorientierten Programmierung stehen nicht wie beim herkömmlichen Vorgehen Unterprogramme oder Prozeduren, sondern **Objekte** - auch *Instanzen* genannt - im Vordergrund.

Diese Objekte modellieren konkrete oder abstrakte Einheiten aus dem Anwendungsbereich. Sie bestehen zum einen aus *Daten*, den Instanzvariablen, die den veränderlichen Zustand des Objektes realisieren und zum anderen aus Aktionen, den sogenannten **Methoden**.

Diese beiden Teile werden als Einheit aufgefaßt. Während der Laufzeit werden Methoden ausgeführt; dabei kann ein Objekt Nachrichten an andere Objekte verschicken und dadurch Aktionen beim Empfänger anstoßen. Die Beschreibung eines Objektes ist seine **Klasse**. Sie

enthält die Deklaration der Instanzvariablen sowie die Beschreibung der Methoden und besitzt Typcharakter. Zu jedem Objekt gibt es genau eine Klasse; dagegen beschreibt eine Klasse im allgemeinen mehrere gleichartige Objekte (1-zu-n Beziehung), die sich in den Werten der Daten unterscheiden können.

Die Daten eines Objektes sind von außen nicht sichtbar (Information Hiding), sie können direkt nur durch Methodenaufrufe gelesen oder modifiziert werden.

Die Menge aller Klassen ist durch das Konzept der **Vererbung** hierarchisch strukturiert. Dadurch können Gemeinsamkeiten und Differenzen ähnlicher Klassen knapp und übersichtlich beschrieben sowie Redundanzen vermieden werden. Die Idee dabei ist, eine Klasse UK von einer anderen Klasse OK abzuleiten und zu erweitern oder zu modifizieren. Zur Erzeugung einer Unterklasse wird weder Quelltext noch Objektcode physikalisch kopiert sondern es genügt ein einfacher Verweis.

Der **Polymorphismus**, d. h. das Vorhandensein mehrerer Methoden, mit unterschiedlichen Implementierungen, für eine Instanzvariable, schließlich ermöglicht es, daß eine Instanzvariable zu unterschiedlichen Zeiten auf Objekte verschiedener Klassen verweist, die dasselbe Methodenangebot, aber unterschiedliche Implementierungen besitzen. Als Konsequenz hiervon ergibt sich, daß bei einer polymorphen Instanzvariablen zur Übersetzungszeit nicht entschieden werden kann, welcher Methodenaufruf ausgeführt wird. Diese Frage wird erst zur Laufzeit beantwortet. Die Tatsache, daß somit erst später entschieden wird, welche Anweisung(en) bei dem entsprechenden Methodenaufruf ausgeführt wird (werden), bezeichnet man als **dynamisches Binden**.

5.3 Schemaorientierte Ansätze

Zur Erläuterung des auf den Schweizer Kognitionswissenschaftler Jean Piaget (1896 - 1980) zurückgehenden Schemakonzeptes seien die wesentlichen Eigenschaften der Schemata kurz aufgelistet, (/15/):

- Schemata sind diejenigen kognitiven Strukturen, in denen allgemeines Wissen im Gedächtnis repräsentiert ist. Dies bedeutet, daß sowohl das Wissen über typische Zusammenhänge in einem realen Bereich, als auch das Wissen über häufig wiederkehrende Handlungs- bzw. Ereignisfolgen in Schemata organisiert ist.

- Schemata sind die elementaren Bedeutungs- und Verarbeitungseinheiten des menschlichen Informationsverarbeitungssystems.

- Schemata können ineinander eingebettet sein, d. h. elementare Schemata können in Schemata höherer Ordnung (Hierarchie) eingebettet sein (Schemata über Schemata).

- Schemata weisen Leerstellen auf, da verschiedene Merkmale in Abhängigkeit vom jeweiligen Kontext unterschiedliche Werte annehmen können. Erwartungen spiegeln sich in sogenannten Default-Werten wider.

- Allgemeine Schemata werden durch den Prozeß der Generalisierung aus speziellen Schemata (Instanzen) gebildet. Spezielles Wissen wird aus dem in allgemeinen Schemata enthaltenen Wissen durch Differenzierung d. h. durch Bildung von Subschemata erzeugt.

- Schemata sind aktive, miteinander in Wechselbeziehung stehende Wissensstrukturen, die aktiv am Verständnis einlaufender Informationen beteiligt sind und die Ausführung der Verarbeitungsoperationen steuern.

- Schemata unterliegen einem ständigen Wandel, indem sie sich entsprechend der je gegebenen Informationsverarbeitungs-Anforderung adaptieren und umstrukturieren. Dieser Wandel fällt unter die allgemeineren Vorgänge der **Akkommodation** und der **Assimilation** (vgl. auch /4/ und /12/).

- Schemata haben nicht nur eine Struktur-, sondern auch eine ausgeprägte Prozeßkomponente; sie beinhalten eine ganze Reihe von Kontrollprozessen, sie bewerten ihre Passung bzw. Kongruenz mit einlaufenden Informationen (durch Verwendung einer Meterik, die Gemeinsamkeiten bzw. Unterschiede bewertet), sie rufen evtl. andere Schemata auf etc.

Daß Schemata über eine Struktur- und eine Prozeßkomponente verfügen schlägt sich im Zusammenspiel zwischen datengeleiteten (bottom up) und schemageleiteten (top down) Verarbeitungsprozessen nieder. Einlaufende Informationen aktivieren bestimmte Schemata (bottom up), die aktivierten Schemata führen ihrerseits zu bestimmten Hypothesen bzgl. der zu erwartenden Information (top down). Diese Vorgehensweise kann man sich bei der Lösung der Probleme der Wissensakquisition und -validierung (vgl. Abschnitt 2.2.3) zu Nutze machen. Auf diese Art und Weise kann beispielsweise eine Klassifikation der dargebotenen Situationen anhand deren Attributen erfolgen.

Durch die sehr stark vernetzte Struktur der Schemata, deren Veränderbarkeit durch adaptive, assimilative und akkommodative Prozesse und die repräsentationale Gleichbehandlung von Faktenwissen, prozeduralem Wissen und heuristischem Wissen, ist dieses Schemakonzept sämtlichen bisherigen "klassischen" und "nicht-klassischen" Konzepten sowohl in der Flexibilität, als auch in seiner Mächtigkeit überlegen.

Der konzeptuelle Entwurf eines auf dem Schemakonzept aufbauenden wissensbasierten Systemes, das C^3R-System wird im Abschnitt 8.3 vorgestellt.

5.4 Wahrscheinlichkeitsbetrachtungen und unscharfes Wissen

5.4.1 Wahrscheinlichkeitsbetachtungen

Das Theorem von Bayes ist ein statistischer Ansatz zur computerunterstützten Diagnose. Die Methode operiert mit zwei Typen von Wahrscheinlichkeiten

- Die Wahrscheinlichkeit P(A):
  P(A) bezeichnet die Wahrscheinlichkeit für das Auftreten eines Fehlers oder eines Symptomes A, unabhängig vom Prozeßzustand.

- Die bedingte Wahrscheinlichkeit P(A/B):
  P(A/B) bezeichnet die Wahrscheinlichkeit für das Auftreten eines Fehlers A bei Vorhandensein eines Symptomes B. Ebenso läßt sich damit die Wahrscheinlichkeit für das Auftreten eines Symtomes B bei Vorhandensein des Fehlers A bestimmen.

Als Ergebnis erhält man die möglichen Fehler und die Wahrscheinlichkeit, mit der ein Fehler Ursache der beobachteten Symptome B sein kann.

5.4.2 Fuzzy Logik

Das Arbeiten mit unscharfen logischen Ausdrücken erfolgt unter Verwendung der Fuzzy Logik /33/. Diese Methode gehört zur assoziativen Diagnostik, das heißt ihr Ansatz beruht auf assoziativem Wissen. Im Gegensatz zu der booleschen Klassifikation, die nur die Werte TRUE und FALSE zuläßt, werden hier, bei der Beantwortung der Frage, ob eine Regel erfüllt ist oder nicht, keine scharfen Grenzen gezogen. Eine Aussage, ob eine Regel erfüllt ist wird bei der Fuzzy-Methode mit einem Gewichtsfaktor zwischen 0 und 1 versehen. Diese Zwischenwerte werden entsprechend den Operationen der Fuzzy Logik verrechnet (z. B. Schwerpunktbildung). Man erhält damit eine Abbildung von Werten in einen unscharfen Bereich und über eine Rücktransformation die entsprechenden Stellwerte. Eine weitere Methode sind Modal-Logiken /28/.

5.4.2.1 Beispiel

Für die rechnerunterstütze Prozessführung von Zement-Brennanlagen wurde die Fuzzy Logik schon sehr früh eingesetzt /33/. Der Führung dieses Prozesses wird erschwert, weil nur wenige Zustandsmerkmale direkt gemessen werden können. Mit der herkömmlichen Technik (klare Antwort JA/NEIN, wie Computer im allgemeinen programmiert werden) waren solche Prozesse, insbesondere im Vergleich zu einem erfahrenen Operator, nicht optimal zu führen.

Zur Prozessbeurteilung beobachtet der Operator vier Größen:

(1) Sauerstoffanteil,

(2) freier Kalkanteil in %,

(3) Änderung des Drehmomentes in % und

(4) CO-Anteil in den Abgasen.

Der Operator wendet 40 bis 50 Faustregeln für die Kontrolle der Material- und Brennstoffzufuhr, der Brenngeschwindigkeit sowie den Gasfluß durch den Ofen an. Eine solche Regel lautet zum Beispiel.

```
Wenn Sauerstoffanteil hoch ist "UND" der freie
Kalkanteil sowie das Drehmoment normal ist
-->
"DANN" erniedrige die Brennstoffzufuhr und reduziere
ein wenig die Beschickung.
```

Es gibt Situationen, bei denen mehrere Regeln zutreffen und die Aktionen, die vorgeschrieben werden, in Konflikt zueinander stehen. Der Operateur versucht zu relativierten Aussagen der zutreffenden Regeln bzgl. der Gesamtaussage zu gelangen. Hierzu gewichtet er die Regel entsprechend dem Grad, wie sie zutrifft.

Der entwickelte Fuzzy-Controller verarbeitet eine Reihe von Faustregeln, die in Form von Fuzzy-Termen (fuzzyfizieren) eingegeben sind. Die Regeln werden in Bedingungs- und Aktionsteil gegliedert. Der kleinste Grad, zu welchem ein Teil der aktuelle Bedingung einer Regel erfüllt ist, bestimmt den Grad, zu welchem die Gesamtregel erfüllt ist.

Der Zusammenhang zwischen Gewichtsfaktor und den gemessenen Werten bei den einzelnen Bedingungen ist als Funktion gegeben. Für die Schlußfolgerung werden zuerst anhand der Fakten und den gegebenen Zusammenhängen die Gewichtsfaktoren der einzelnen Regeln ermittelt und mit diesen jeweils die Kurven für die Bestimmung des Eingriffes multipliziert. Von den so erhaltenen Kurven wird der gemeinsame Schwerpunkt ermittelt, dessen Lage die Aussage für die Anpassung liefert (Wert der Stellgröße) (defuzzyfizieren).

6 Berücksichtigung von implizitem Wissen (neuronale Netze)

Getragen von der Idee die Vorgänge bei der menschlichen Informationsverarbeitung zu simulieren, beschäftigt sich die Informatik bereits seit Ende der 50er Jahre damit Beziehungen zwischen dem menschlichen Gehirn und mathematischen Maschinen zu finden.

In den vergangenen zehn Jahren wurden die Aktivitäten auf diesem Gebiet sehr stark vorangetrieben und es entstand, als interdisziplinäres Forschungsgebiet, der **Konnektionismus**. Der Inhalt des Konnektionismus ist die Erforschung und Konstruktion informationsverarbeitender Systeme, die sich aus vielen primitiven, uniformen Einheiten zusammensetzen und deren wesentliches Verarbeitungsprinzip die Kommunikation zwischen diesen Einheiten ist. Ein weiteres Charakteristikum dieser Systeme ist die parallele Verarbeitung von Information innerhalb des Systems durch eine gleichzeitige Aktivität vieler Einheiten (/17/).

Die generelle Zielsetzung des Konnektionismus ist die Modellierung kognitiver Prozesse mittels derartiger Konnektionistischer Modelle, sogenannter **neuronale Netze**.

Die wahrscheinlich treffendste Definition für ein neuronales Netzes stammt von Kohonen /20/:

> Künstliche neuronale Netze sind massiv parallel verbundene Netzwerke aus einfachen (üblicherweise adaptiven) Elementen in hierarchischer Anordnung oder Organisation, die mit der Welt in derselben Art wie biologische Nervensysteme interagieren sollen.

Der Aufbau und die Funktionsweise neuronaler Netze wird im folgenden, wie bei /24/, dargestellt.

6.1 Die Grobstruktur des menschlichen Gehirns

Ein neuronales Netz versucht in gewisser (eingeschränkter) Weise die Funktionsweise des menschlichen Gehirns nachzubilden. Daher ist es sinnvoll, zunächst einen kurzen Überblick über dessen Funktionsweise zu geben.

Die Mächtigkeit des Gehirns basiert auf der großen Anzahl der Neuronen und deren internen Verbindungen zur parallelen Interaktion. Ein **Neuron** ist eine Nervenzelle, mit allen zugehörigen Prozessen.

In den **Zellkern** (**Nukleus**) führen ein oder mehrere **Dendriten**. Diese Prozesse der Nervenzelle leiten Impulse zum **Zellkörper**, der den Nukleus enthält. Das **Axon** ist der Prozeß

einer Nervenzelle, der Impulse vom Zellkörper weg leitet. Bündel von Neuronen oder Nervenfasern bilden **Nervenstrukturen.**

In einer stark vereinfachten Darstellung werden Impulse zwischen einzelnen Neuronen dadurch Übertragen, daß sie durch Nerven von den **Rezeptor Organen** (z. B. Augen und Ohren) zu den **Effektor Organen** (z. B. Muskeln und Drüsen) geleitet werden.

Der Punkt zwischen zwei Neuronen in einem neuronalen Pfad, an dem das Axon des einen Neurons sehr nahe an den Zellkörper oder die Dendriten des anderen Neurons kommt heißt **Synapse.** Die einzige Beziehung zwischen den beiden Neuronen an diesem Punkt besteht in deren Kontakt. Der Impuls, der sich durch das eine Neuron bewegt, löst an dieser Stelle einen Impuls im zweiten Neuron aus.

In Verbindung mit neuronalen Netzen sind die folgenden Vorgänge in den Synapsen wesentlich:

- Die in die Synapsen einlaufenden Signale sind gewichtet, d. h. einige Signale sind stärker als andere. Einige Signale wirken anregend, andere wirken hemmend. Die Auswirkungen aller gewichteter Eingaben werden summiert.
- Falls die Summe größer oder gleich dem Schwellenwert des Neurons ist, so liefert das Neuron eine Ausgabe, d. h. es feuert. Es handelt sich folglich um eine "Alles oder Nichts Situation", denn entweder "feuert" das Neuron oder nicht.
- Durch die Aktivität des Nervensystems (z. B. erhöhte Aufmerksamkeit, Müdigkeit) kann die Übertragungsrate der Signale (positiv und negativ) beeinflußt werden. Diese Fähigkeit, Signale zu regulieren, ist ein Mechanismus zum Lernen.
- Die Schwellenwertfunktionen integrieren die Energie einlaufender Signale im Nervensystem über Raum und Zeit.

6.2 Die Grundelemente neuronaler Netze

6.2.1 Das Prozeßelement

Das **Prozeßelement** oder der **Knoten**, die **Unit** (vgl. Abb. 9), eine Art künstliches Neuron, bildet die Grundeinheit eines neuronalen Netzes. Es erfüllt die folgenden Basisfunktionen:

- Auswertung der gleichzeitig eintreffenden Signale, indem die Stärke jedes einzelnen bestimmt wird.
- Berechnung des gesamten Input-Signals als Kombination der einzelnen Werte.
- Vergleich des gesamten Input-Signals mit einem Schwellenwert.

- Bestimmung der Ausgabe, wobei höchstens ein Ausgabe-Signal erfolgen kann; d. h. es gibt zwar mehrere Eingaben (Inputs), aber nur genau eine Ausgabe (Output).

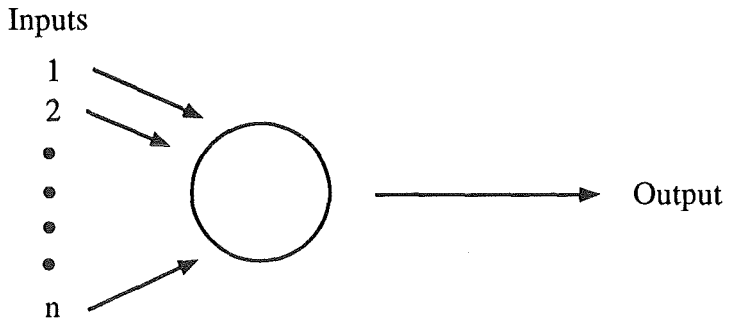


Abbildung 9

6.2.2 Die Funktionen (Prozesse) des Prozeßelements

Um die Relevanz eines jeden Input-Signals im Bezug auf das gesamte Input-Signal beeinflussen zu können, kann jedes Input-Signal mit einem besonderen Gewicht versehen werden (Abb. 10).

Die im folgenden beschriebenen Prozesse der Summationsfunktion, der Aktivierungsfunktion und der Übertragungsfunktion sind es in erster Linie, die die Dynamik eines neuronalen Netzes bestimmen.

Mathematisch kann man die Input-Signale und die Gewichte als Vektoren $(i_1, i_2, ..., i_n)$ und $(w_1, w_2, ..., w_n)$ auffassen. Die Berechnung des gesamten Input-Signals erfolgt mit Hilfe einer **Summationsfunktion**, z. B. die gewichtete Summe, d. h. dem euklidischen Skalarprodukt der beiden Vektoren, was geometrisch als ein Maß für die Ähnlichkeit dieser beiden Vektoren aufgefaßt werden kann.

Wenn die gewichtete Summe der Input-Signale größer als der Schwellenwert ist, dann erzeugt das Prozeßelement ein Signal. Falls das Ergebnis der Summationsfunktion kleiner als der Schwellenwert ist, dann wird kein Signal (oder ein hemmendes Signal) erzeugt; beide Antworten sind signifikant.

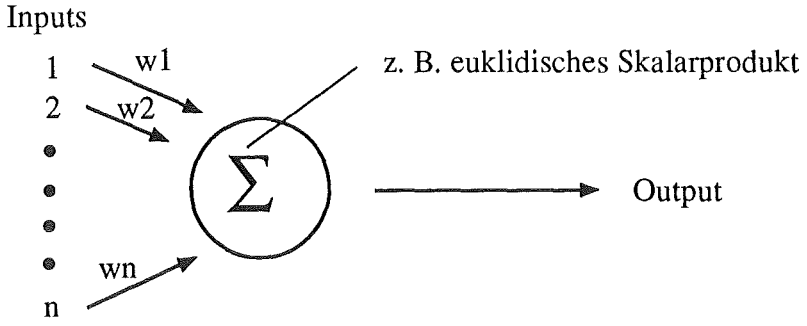


Abbildung 10

Ein weiterer Prozeß im Prozeßelement betrifft dier **Aktivierungsfunktion**. Das Ergebnis der Summationsfunktion kann als Argument für eine Aktivierungsfunktion dienen, bevor es an die Übertragungsfunktion (T in Abb. 11) übergeben wird. Der Sinn der Aktivierungsfunktion besteht darin, daß sie es ermöglicht, daß die Ausgabe in Abhängigkeit von der Zeit variiert. Die Aktivierungen früherer Zeitpunkte geben dem Prozeßelement ein Gedächtnis, mit dem beispielsweise Adaption modelliert werden kann. Allgemein ist die Aktivierungsfunktion von vorhergehenden Aktivierungen und von einem Satz von Parametern abhängig (/19/). In den meisten Fällen wird als Aktivierungsfunktion die Identität verwendet, da die Untersuchungen auf diesem Gebiet noch nicht sehr weit fortgeschritten sind. Den Wert der Aktivierungsfunktion bezeichnet man als **Aktivierungswert**.

Eine wichtige Funktion ist die im allgemeinen nichtlineare **Übertragungsfunktion** T (**Schwellenwert-**, **Output-Funktion**). Diese Funktion ist im allgemeinen nicht-linear, da lineare Funktionen aufgrund der Tatsache, daß ihr Resultat proportional zur Eingabe ist, nur beschränkte Möglichkeiten bieten und sich daher viele Probleme, wie z. B. das exclusive-or-(XOR-) Problem, mit linearen Funktionen nicht lösen lassen (vgl. z. B. /24/ oder /19/).

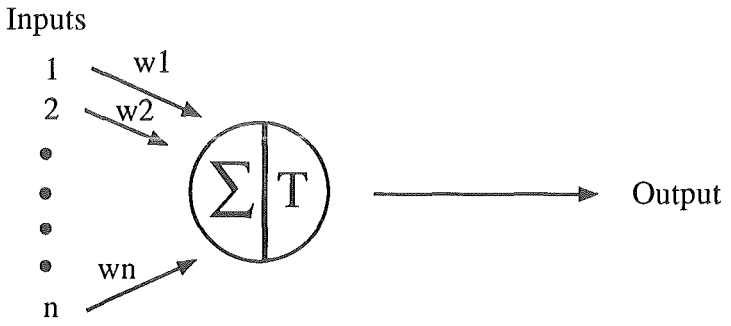


Abbildung 11

Bei den normalerweise üblichen Übertragungsfunktionen (vgl. z. B. /24/ oder /19/) werden in den Anwendungen sigmoide, d. h. S-förmige, Funktionen bevorzugt werden, da sie außer der Nicht-linearität noch über gewisse Differenzierbarkeitseigenschaften verfügen.

Die Anforderungen an eine Übertragungsfunktion sind, daß diese für negative oder zu kleine positive Werte ein definiertes Minimum (entsprechend der Ruhefrequenz beim Neuron) oder 0 liefert und ab einem gewissen Schwellenwert allmählich oder abrupt einen Maximalwert, häufig 1 animmt (/19/).

6.2.2.1 Lernfunktionen

Im Bereich der Lernfunktionen liegt der Vergleich mit dem menschlichen Gehirn wieder nahe. In gleicher Weise wie an den biologischen neuronalen Synapsen können Signale im Prozeßelement hemmend oder stimulierend in bezug auf die Erzeugung einer Ausgabe wirken.

Falls man das Prozeßelement mit einem lokalen Speicher versieht, so kann es Ergebnisse vorangegangener Berechnungen speichern und die im weiteren Verlauf verwendeten Gewichte modifizieren. Diese Fähigkeit, die Gewichte zu verändern, ermöglicht es dem Pro-

zeßelement sein Verhalten in Abhängigkeit von den Eingaben zu verändern, d. h. zu lernen, im Sinne einer Verbesserung der Performanz.

6.2.3 Die Kombination von Prozeßelementen zu einem Layer

Bisher wurden immer nur einzelne Prozeßelemente betrachtet. Der nächste Schritt besteht nun darin, mehrere Prozeßelemente zu einer **Schicht (Layer)** zu kombinieren. Die Input-Signale können zu mehreren Prozeßelementen mit unterschiedlichen Gewichten verbunden sein, was dann zu einer Reihe von Ausgaben führt, je eine pro Prozeßelement (Abb. 12).

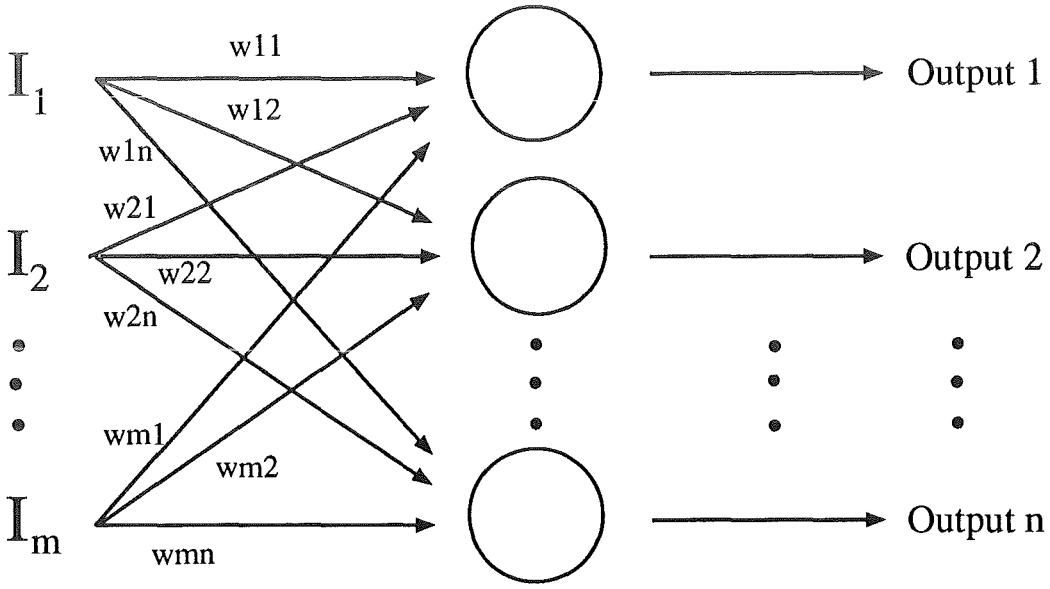


Abbildung 12

6.2.4 Die Kombination mehrerer Layer zu einem mehrschichtigen neuronalen Netz

Setzt man den bisher verfolgten Weg zur Konstruktion eines neuronalen Netzes fort, so muß man nun folgerichtig mehrere Schichten zu einem umfangreicheren neuronalen Netz kombinieren.

Derjenige **Layer** (Schicht oder Ebene), der die Inputs empfängt heißt **Input-Layer**. Seine Aufgabe besteht in der Pufferung der Input-Signale. Der Output des Netzwerkes wird von einem **Output-Layer** erzeugt. Sämtliche anderen Layer heißen, da sie keinen direkten Kontakt zur externen Umwelt haben **Hidden-Layer** (vgl. Abb. 13). In einem neuronalen Netz kann es beliebig viele Hidden-Layer geben.

Es liegt in der Natur der Sache, daß es unzählig viele Möglichkeiten gibt, die Knoten der einzelnen Schichten miteinander zu verbinden. Einige der bekanntesten werden im folgenden kurz beschrieben.

- Ein Netzwerk heißt **vollständig verbunden** (fully connected), wenn jeder Output eines Layer mit jedem Knoten des nächsten Layer verbunden ist.

- Das Ausgabesignal eines Knotens kann jedoch auch als Input-Signal darunterliegender Schichten dienen. In einem hierarchisch strukturierten Netzwerk unterscheidet man die folgenden Varianten:

  - **Feed forward-Netze:**
    Die Prozeßelemente eines bestimmten Layers haben keinen Einfluß auf darunterliegende Layer.

  - **Interaktive Modelle:**
    Es können sowohl Verbindungen nach oben, als auch nach unten vorhanden sein.

  Generell gilt jedoch in hierarchischen Netzwerken, daß Verbindungen nur innerhalb oder zu direkt benachbarten Layern (oberhalb oder unterhalb) erlaubt sind.

- In einem nicht-hierarchisch strukturierten Netzwerk gibt es praktisch keine Ordnung. Jedes Prozeßelement kann mit sich und mit jedem anderen Prozeßelement beliebig in Verbindung treten. Solche Netze werden auch **feedback-Netze** genannt. Netzwerke mit geschlossenen Schleifen heißen **rekursive Systeme**.

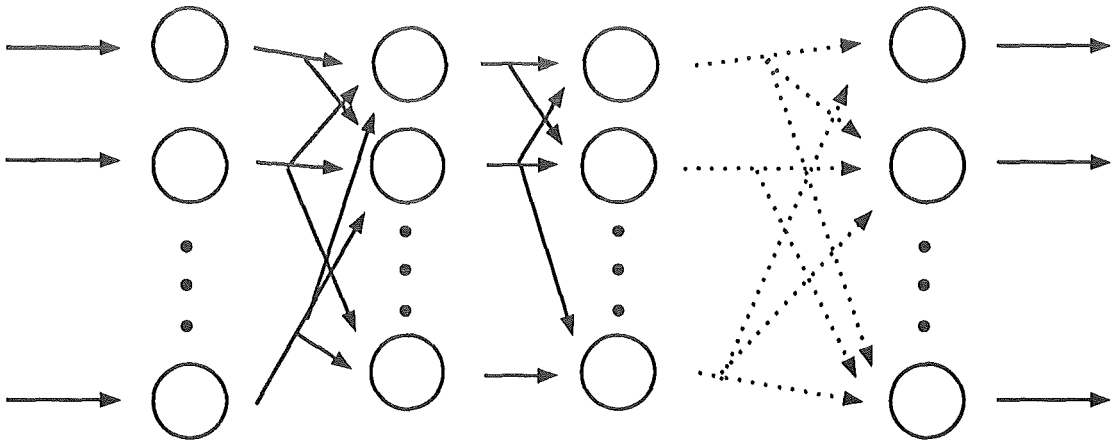


Abbildung 13

Durch das Auftreten von geschlossenen Schleifen (Rekursionen) ergeben sich interessante Möglichkeiten. Man darf jedoch nicht übersehen, daß solche Systeme, sofern sie nicht bestimmten Bedingungen genügen, zum Schwingen neigen. Die Meinungen über die Leistungsfähigkeit hierarchischer gegenüber nicht-hierarchischer Systeme gehen auseinander.

Minsky und Pappert haben gezeigt, daß man mit den bekannten Methoden der Entrekursivierung rekursive Netze, in endlicher Zeit, in entsprechende feed forward-Netze umwandeln kann (/19/).

Allgemein gilt, daß es in einem solchen Netzwerk mehr Verbindungen als Knoten gibt.

6.3 Lernen in neuronalen Netzen

Für das Lernen in neuronalen Netzen ist entscheidend, wie das Ausgabe- oder Reaktionsverhalten des Netzes gebildet wird bzw. geändert werden kann, und demnach auch, wie und wo das Netz etwas speichert, das sein Verhalten bestimmt. Das statische (gespeicherte) Wissen des Netzes liegt seinen Verbindungen (deren Gewichten) und in seinem Aufbau, das dynamische (aktuelle) ist in den Aktivierungswerten enthalten, die, wenn es sich um Prozeßelemente zur Ausgabe handelt, die Reaktion des Netzes auf ein präsentiertes Eingabemuster darstellen.

Das zu Erlernende (allgemein ein Muster) wird in neuronalen Netzen nicht explizit gespeichert. Vielmehr sind die Gewichte zwischen den einzelnen Prozeßelementen, die es erlauben, daß ein bestimmtes Muster immer wieder erzeugt werden kann, gespeichert. Lernen in einem neuronalen Netz bedeutet zumeist, die Gewichte richtig zu trainieren. Man geht dabei meist von einer Initialisierung mit Zufallszahlen aus und wendet dann eine Lernstrategie zur Veränderung der Gewichte des Netzes an. Dazu werden dem Netz wiederholt Muster präsentiert. Es werden keine expliziten Regeln gelernt, sondern die Regeln werden anhand der Daten implizit mitgelernt (generalisiert).

Prinzipiell kann man drei Arten der Veränderung von Gewichten unterscheiden:

(1) Entwicklung neuer Verbindungen,

(2) Abbruch vorhandener Verbindungen,

(3) Veränderung der Gewichte bereits existierender Verbindungen.

Die Varianten (1) und (2) können als Spezialfälle von (3) angesehen werden, wenn dem Verbindungsabbau ein Setzen des entsprechenden Gewichtes auf 0 entspricht und dem Verbindungsaufbau einem Ändern des Gewichtes von 0 auf einen Wert ungleich 0.

Da noch keine Klarheit herrscht, nach welchen Kriterien der Auf- respektive Abbau der Verbindungen erfolgen soll, sind die Varianten (1) und (2) noch recht unerforscht.

Wie bereits dargestellt wird ein Großteil des Wissens in einerm neuronalen Netz verteilt repräsentiert; dies bedeutet, daß die Zustände einzelner Prozeßelemente bzw. ihre Verbindungen nicht mehr oder in nicht mit vertretbarem Aufwand in Symbole überführt werden können, die für einen Menschen verständlich sind. Die Problemlösungseigenschaften des neuronalen Netzes im Hinblick auf das spezielle Ausgangsproblem, für welches das neuronale Netz konstruiert wurde - entstehen nur im Zusammenspiel der einzelnen Prozeßelemente. Der Lösungsweg kann deshalb nicht wie bei einem Expertensystem unter Bezugnahme auf die angewendeten Symbole erklärt werden. Das (Problemlösungs-) Wissen wird auf einer niedrigeren Abstraktionsebene, "unterhalb" der symbolischen Repräsentation dargestellt. Man spricht daher auch von **subsymbolischer** Verarbeitung (/23/).

Eine auffallende Gemeinsamkeit der bekannten Lern-Algorithmen für neuronale Netze ist deren ausschließliches Zugreifen auf lokale Informationen, d. h. auf Informationen benachbarter Prozeßelemente. Außerdem ist interessant, daß sie zumeist ohne übergeordnete Instanz zu implementieren sind, d. h. ohne globale Ablaufsteuerung. Dadurch kann das Lernen in fast allen Fällen parallel und ohne globale Zugriffe vor sich gehen, was sich bei entspre-

chender Hardware sehr günstig auf den zeitlichen Aufwand auswirkt und sicherlich auch eher den Vorgängen beim Lernen des Menschen entspricht.

6.3.1 Verschiedene Formen des Lernens in neuronalen Netzen

Grundsätzlich kann man die verschiedenen Lernansätze nach der Art der Präsentation der zu erlernenden Muster einteilen. Hierzu unterscheidet man:

- Überwachtes Lernen oder Klassifizieren (supervised learning, learning by reinforcement):
  Bei diesem Lernmodus wird die tatsächliche Ausgabe des neuronalen Netzes mit der gewünschten Ausgabe verglichen. Die Gewichte werden dann durch das neuronale Netz verändert, so daß bei der nächsten Iteration eine bessere Übereinstimmung mit dem Muster erreicht wird. Das Ziel sämtlicher Lernalgorithmen besteht in einer nachhaltigen Minimierung des Fehlers zwischen dem gewünschten und dem tatsächlichen Ausgabemuster durch kontinuierliche Modifizierung der Gewichte.

  Beim überwachten Lernen muß das neuronale Netz vor seinem praktischen Einsatz "trainiert" werden. Das Training besteht darin, daß man dem neuronalen Netz Eingabe- und Ausgabedaten, eine Trainingsmenge, präsentiert; d. h. zu jedem gegebenen Eingabemuster präsentiert man dem neuronalen Netz das gewünschte Ausgabemuster.

  Wenn der Lernvorgang abgeschlossen ist, dann werden die Gewichte "eingefroren", d. h. es kann dann nicht weiter gelernt werden und damit findet eine deutliche Unterscheidung zwischen Trainingsphase und Einsatzphase des neuronalen Netzes statt.

  Beispiele überwachter Lernansätze sind:

  - Selbstassoziation (Auto Association),
  - Musterassoziation (Pattern Association),
  - Klasseneinteilung (Classification Paradigm) und
  - Fehlerpropagierung (Error (Back-) Propagation).

- Nicht überwaches Lernen oder Clustering (unsupervised learning, learning by doing):
  Bei diesem Lernmodus benötigen die neuronalen Netze keine externe Beeinflussung, um ihre Gewichte anzupassen. Dafür verfügen sie über eine interne Möglichkeit ihre Performanz zu überwachen. Das neuronale Netz sucht nach Regularitäten oder Tendenzen in den Eingabesignalen und führt Anpassungen bzgl. der Funktion des Netzes durch.

  Selbst wenn dem neuronalen Netz nicht mitgeteilt wird, ob es mit seinen Entscheidungen richtig liegt, so muß es doch über Information verfügen, wie es sich selbst organisieren soll.

Ein Algorithmus zum nicht-überwachten Lernen könnte den Schwerpunkt auf die Kooperation zwischen Clustern von Prozeßelementen legen. In einem solchen Schema würden die Cluster miteinander arbeiten und versuchen, sich gegenseitig zu stimulieen.

Ebenso könnte ein Wettbewerb zwischen den Prozeßelementen die Basis für das Lernen bilden (sogenanntes Wettbewerbslernen). Das Training miteinander in Wettbewerb stehender Cluster könnte die Antwort bestimmter Gruppen auf spezielle Reize verstärken und diese Gruppen miteinander und mit einer angemessenen Antwort verbinden.

6.3.2 Lernregeln

Im folgenden werden einige der bekanntesten Lernregeln für neuronale Netze kurz vorgestellt:

(1) Hebb-Regel:
Dies ist die älteste und am weitesten verbreitete Lernregel. Sie besagt: Wenn die Prozeßelemente a und b zugleich (wiederholt) stark aktiviert sind, so erhöhe die Stärke ihrer Verbindung.

(2) Die Delta-Regel:
Diese häufig verwendete Regel basiert auf der einfachen Idee, die Stärke der Verbindungen kontinuierlich zu modifizieren, um die Differenz (das Delta) zwischen dem gewünschten Ausgabewert und dem momentanen Ausgabewert eines Prozeßelements zu verringern.

Diese Regel wird auch **Widrow-Hoff-Lernregel** oder Methode der kleinsten Fehlerquadrate (least mean square learning rule) genannt.

(3) Gradienten-Abstiegs-Regel (Gradientenverfahren):
Dies ist ein mathematischer Ansatz, um den Fehler zwischen den aktuellen und den gewünschten Ausgaben zu minimieren. Die Gewichte werden um einen Betrag modifiziert, der proportional zum Wert der ersten Ableitung der Fehlerfunktion bzgl. der Gewichte ist.

Obwohl das Konvergenzverhalten sehr langsam ist, wird diese Methode sehr häufig verwendet. Die Delta-Regel ergibt sich als Spezialfall aus dieser Methode.

(4) Kohonens Lernregel:
Diese von Teuvo Kohonen entwickelte Lernregel wurde durch das Lernen in biologischen Systemen inspiriert. Sie wird nur in nicht-überwachten-Lernsituationen angewandt.

Bei dieser Lernregel besteht zwischen den Prozeßelementen ein Wettbewerb, um die Gelegenheit zum Lernen zu erhalten. Das Prozeßelement mit dem

größten Ausgabewert wird zum Sieger erklärt und erhält die Möglichkeit, sowohl seine Mitbewerber zu hemmen, als auch seine Nachbarn zu stimulieren. Nur dem Sieger ist es gestattet eine Ausgabe zu erzeugen und nur der Sieger und dessen Nachbarn dürfen die Gewichte anpassen.

(5) Back Propagation Lernen:
Back-Propagation ist eine Verallgemeinerung der Delta-Regel auf Netzwerke mit beliebig vielen Layer und feed-forward-Verarbeitung. Die prinzipielle Idee ist, daß die Hidden-Prozeßelemente (d. h. jene Prozeßelemente, die weder Input- noch Output-Prozeßelemente sind) eine interne Repräsentation der Musterassoziationen durch Rückwärtspropagieren eines Fehlers vom Output-Layer in Richtung Input-Layer (error-back-propagation) erlernen.

Das Lernverfahren erfolgt in zwei Schritten: Zuerst wird das angelegte Muster in Richtung Output-Layer propagiert, um dort die Reaktion des Netzes auf das präsentierte Muster zu generieren. In der zweiten Phase erfolgt die Gewichtsänderung, abhängig vom Grad der Falschheit der Netzantworten. Die grundlegende Idee des Fehlerrücksendens hat bereits Rosenblatt (1962) formuliert.

Die Ausbreitung durch das Netz erfolgt schichtweise, d. h. die Aktivierungen der Prozeßelemente werden zuerst in jenem Hidden-Layer berechnet, der dem Input-Layer am nächsten liegt. Dann erfolgt die Berechnung der Aktivierung des nächsten, weiter beim Output-Layer gelegenen Layer, bis schließlich der Output-Layer selbst erreicht wurde.

In der zweiten Phase erfolgt die Fehlerbestimmung und die entsprechende Gewichtsveränderung, wiederum schichtweise. Dabei wird beim Output-Layer begonnen, da hier das gewünschte Muster zur Verfügung steht und mit dem tatsächlichen vom Netz produzierten Muster verglichen werden kann. Aus der Differenz dieser beiden Muster wird ein sogenanntes Fehlersignal gebildet, von dem einerseits die Änderung der Gewichte zwischen dem Output-Layer und dessen benachbartem Hidden-Layer und andererseits auch die Berechnung des neuen Fehlersignals für den nächsten Hidden-Layer abhängt.

Ist der Fehler bis zum letzten Hidden-Layer zurückgesendet und wurden dabei alle Gewichtsänderungen vorgenommen, kann wieder ein (neues) Muster angelegt und vorwärts propagiert werden. Wendet man diese Lernprozedur wiederholt an, so wird der Fehler schrittweise verringert. Es sei darauf hingewiesen, daß man andere, bereits erlernte Muster mit diesem Vorgehen zerstören kann. Der Lernvorgang wird, wenn der Fehler entsprechend klein geworden ist, als beendet angesehen.

Da es sich bei der Back Propagation um ein Gradientenverfahren handelt, kann sich das Netz in einem lokalen Minimum verfangen und die gestellte Aufgabe nicht fehlerfrei erlernen. Bei binären Aufgabenstellungen mag das nicht störend sein, da die Entscheidbarkeit (und nicht die Fehlerfreiheit) ausschlaggebend ist.

(6) Grossbergs Lernregel:
Grossbergs Lernregel kombiniert die Hebb-Regel mit dem biologischen Vorgang des Vergessens.

In Grossbergs Modell wird jedes neuronale Netz aus Instars und Outstars gebildet; ein Instar ist ein Prozeßelement das viele Eingaben empfängt, ein Outstar dementsprechend eines, das seine Ausgaben zu vielen anderen Prozeßelementen sendet. Hier erlauben die Verbindungen den Rückruf eines konzentrierten Bildes von einem einzigen Outstar-Knoten; das Muster wird verteilt gespeichert.
Falls ein Knoten sowohl eine hohe Input-, als auch eine hohe Outputaktivität aufweist, so werden die zugehörigen Gewichte signifikant verändert. Falls entweder der gesamte Input oder die Ausgaben kleine Werte annehmen, so werden die Veränderungen der Gewichte ebenfalls gering ausfallen und die Gewichte können auf unwichtigen Verbindungen sogar gegen 0 gehen.

Die Zeit spielt bei Grossbergs Lernmodell eine wesentliche Rolle. Falls ein Eingabereiz weggelassen wird, so wird im Laufe der Zeit, wenn das Vergessen einsetzt, auch die Antwort (in der Ausgabe) auf diesen Reiz nachlassen.

(7) Lernen in Cauchy und Boltzmann Maschinen:
Die hier verwendeten Lernprozeduren entsprechen eher stochastischen-, als deterministischen Lernprozeduren.

Ein Algorithmus für Boltzmann Maschinen kann folgendes beinhalten: In einem Temperaturgleichgewicht sind alle Zustände im neuronalen Netzwerk möglich, wobei deren relative Wahrscheinlichkeiten durch eine Boltzmann-Verteilung gegeben sind. Falls die Wahrscheinlichkeiten der Zustände im Modell denjenigen der Umwelt entsprechen, dann verfügt das neuronale Netz über ein abstraktes Modell der Umwelt.

7 Das Projekt "Innovative Prozeßführung"

Eine Lösung der in der Einleitung skizzierten Problemstellung erfordert die Entwicklung einsetzbarer Werkzeuge, die Methoden aus den unterschiedlichsten wissenschaftlichen Bereichen integrieren. Im Bereich einer "wissensbasierten Prozeßführung" müssen diese Werkzeuge die folgenden Operationen unterstützen:

- Eine direkte oder modellgestütze Messung der Prozeßgrößen (numerische Komponente).
- Die Störungsfrüherkennung als permanente simultane Bewertung aller Prozeßgrößen sowie daraus ableitbarer Größen (Signale) bzgl. ihrer Referenz (Kontext). Als Ergebnis hieraus resultieren die beobachteten Symptome.
- Die Ermittlung der den Symptomen (Symptomatik, Situation) zugeordneten Ursachen in Form von Teil- und Gesamtdiagnosen (dies beinhaltet eine zusätzliche Ableitung von Informationen/Symptomen).
- Die Ableitung von Eingriffen zur Behebung der Störung/Beseitigung der fehlerhaften Situation bzw. Abschwächung derselben (inkl. Folgestörungen).
- Eine automatische Ableitung von kausalen Beziehungen (Wissensakquisition und -validierung).

Diese Operationen sind im Kontext einer verteilten, parallel ablaufenden Anlage mit Rückkopplungen zu sehen. Durch die Dynamik des Systemverhaltens können lokal behebbare Störungen Folgestörungen verursachen, welche über Rückkopplungen wieder eine lokale Störung zur Folge haben können. Eine Ableitung von Gegenmaßnahmen muß die strukturellen Eigenheiten (Teilprozesse, Apparate, Apparatekomponenten) des technischen Pro- zesses über alle Hierarchie-Ebenen hinweg berücksichtigen.

Neben der Entwicklung eines direkt einsetzbaren Werkzeugs muß eine Umgebung entwickelt werden, welche die Modellierung von technischen Prozessen unter dem Gesichtspunkt einer echtzeitorientierten, wissensbasierten Prozeßführung unterstützt.

7.1 Methodische Ansätze

Im Projekt INPRO (Innovative Prozeßführung) wird das Konzept einer wissensbasierten Prozeßführung erarbeitet. Hierzu sind sowohl die Integration von Methoden aus verschiedenen Bereichen, als auch die Erarbeitung bestimmter methodischer Lösungen notwendig.

Die zu betrachtenden methodischen Bereiche sind:

- numerische Verfahren für mathematischen Modelle zur funktionalen Beschreibung von Komponenten,
- symbolische Verfahren für mathematische Funktionen zur on-line Erweiterung um Beeinflußungsgrößen und zur symbolischen Analyse der Ursache-Wirkungs-Beziehungen,
- Darstellung und Verarbeitung von heuristischem Wissen in Form von Regeln und Schemakonzepten,
- realitätsnahe, bedeutungsorientierte Strukturierung von Wissen in jeglicher Form (objekt-, schemaorientierter Ansatz),
- Modellierungsmöglichkeiten für konkrete und abstrakte Objekte und Siutationen,
- automatische Ableitung und Validierung von Wissen und
- Interaktion mit dem Benutzer über objektorientierte graphische Oberflächen.

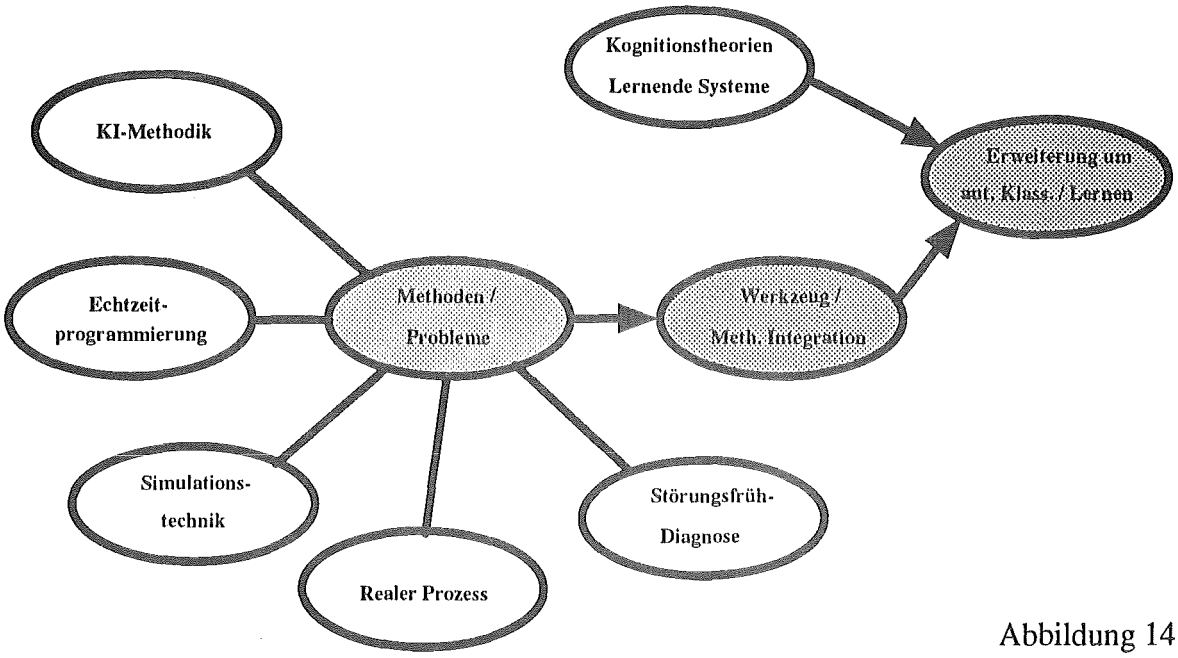


Abbildung 14

Gerade im Bereich der Strukturierung von Wissen nach Bedeutungsmerkmalen oder Prozeßtopologien bei einer autonomen, aber verkoppelten Verarbeitung sind methodische Arbeiten notwendig. Der Datenaustausch bei modellierten Einheiten, die eigenständige Wissensbasen besitzen, erfordert ein anderes Konsistenzprinzip bzgl. der Schlußfolgerung, als in globalen Wissensbasen (ähnlich den Blackboard-Architekturen /9/, aber erweitert um direkte Kommunikationsstrukturen, wie Actor-Systeme /1/). Eine situationsorientierte Ver-

arbeitung erfordert darüber hinaus eine prioritätengesteuerte Ausführung unter Echtzeitbedingungen, wobei Unter- und Obersituationen zu modellieren sind. Anhand der Projekte TEX-I (/18/) und TEX-B (/7/) sind gerade diese Anforderungen eindeutig abzuleiten.

Das Problem der Akquisition von heuristischem Wissen für wissensbasierte Systeme wird über lernende Systeme (selbstlernend) angegangen. Durch die Rückführung auf elementare Strukturen kann dieser Problembereich in seiner Komplexität reduziert und möglicherweise als Formalismus modelliert werden. Hierzu laufen eigenständige Arbeiten, die über das obige Integrationsproblem hinausgehen.

Aus den verschiedenen methodischen Bereichen sind die wesentlichen Verfahren oder Ansätze zu integrieren:

- KI-Methodik:
  Regeln, objektorientierte Strukturierung, Vererbung, qualitative bzw. algebraische Verfahren, neuronale / konnektionisische Ansätze.

- Simulationstechnik-Methodik:
  Numerische Verfahren, Modellierungskonzepte.

- Softwaretechnik / PDV-Methodik:
  Prozeßorientierte Ausführung, Echtzeitfähigkeit durch Prioritäten, Unterbrechungsmöglichkeiten, Kommunikationsprinzipien, Prozeßankopplung, verteilte, modulare Struktur, Modulkonzept mit Abstraktion.

- Störungsfrühdiagnose (SFD) -Methodik:
  Einfache numerische Verfahren, parametrische Diagnosemodelle, Multi-Modell-Diagnose.

- Realer, technischer Prozeß:
  Unbestimmtheiten in der mathematischen Modellierung, Nichtmeßbarkeiten, Situationserkennung, Symptomzuordnung, Heuristiken, Bedienerphilosophien.

Aus dem Bereich der Kognitionswissenschaften sind Strukturen und Verfahren der Schematheorien und Problemlösungsheuristiken (z. B. Teilzielbildung, Interpolation, Analogieschlüsse, Klassenmetriken) zu berücksichtigen.

7.2 Randbedingungen

Die wissensbasierte Prozeßführung soll in erster Linie zur Vermeidung der Bildung von Schadstoffen in einem technischen Prozeß beitragen. Eine Verwirklichung dieses Zieles garantiert auch die Verfügbarkeit des technischen Prozesses und damit die Möglichkeit zu dessen optimaler Führung. Dies hat weitreichende Folgen für die Wirtschaftlichkeit, d. h. für die Ausnutzung bzw. Verwendung von Ressourcen und die Einhaltung vorgegebener

Toleranzen, und für die Umweltverträglichkeit des technischen Prozesses durch die Beherrschung bzw. Vermeidung von Störungen.

Potentielle Anlagen bei denen diese Methodik erfolgreich eingesetzt werden kann sind z. B. Müllverbrennungsanlagen oder auch andere verfahrenstechnische Anlagen. Die Vorgehensweise und die Anforderungen wurden deshalb methodisch an solchen Anlagen ausgerichtet

Die wesentlichen Anforderungen im Bereich der Prozeßführung sind

- die frühzeitige Erkennung von Störungen,
- die Ermittlung der zugrundeliegenden Ursachen,
- Ableitung von Eingriffen zur Behebung bzw. Abschwächung/Begrenzung von Störungen.

Die Störungsfrühdiagnose (SFD) ist eine wesentliche Komponente im Bereich der wissensbasierten Prozeßführung (siehe Kapitel 3). Sie basiert sowohl auf analytischem als auch auf heuristischem Wissen.

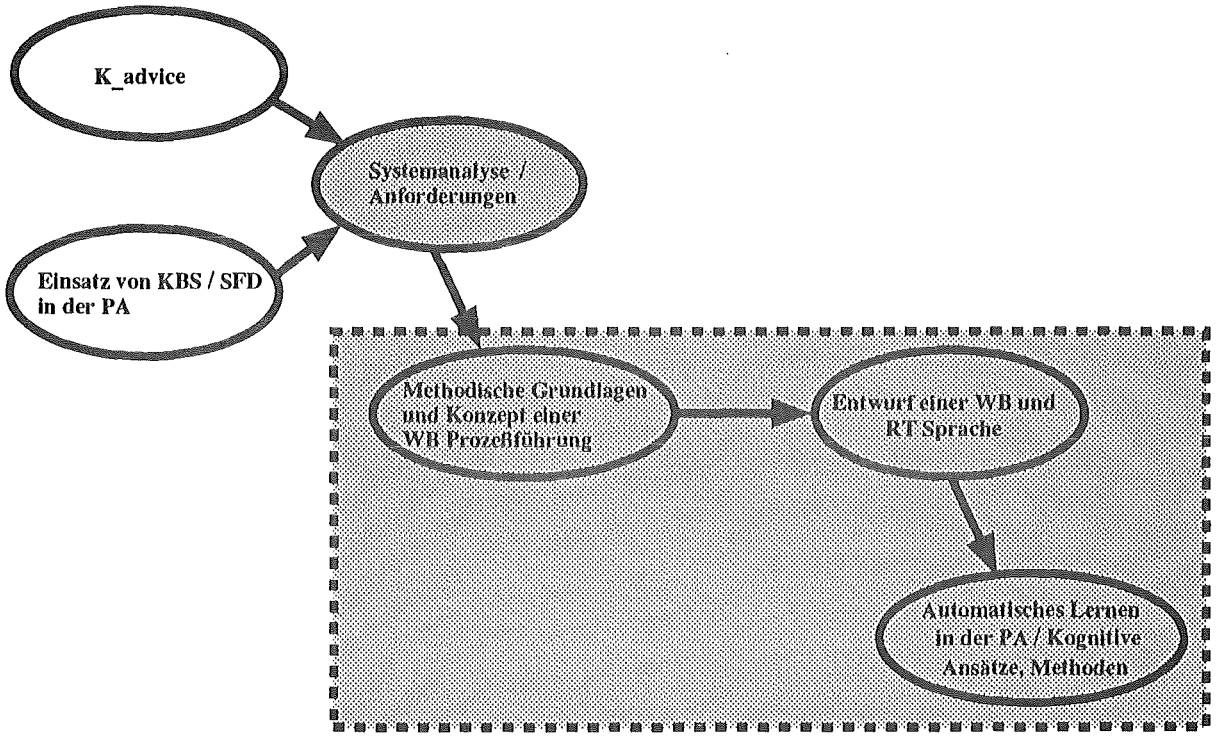


Abbildung 15

(KBS = Wissensbasiertes System; SFD = Störungsfrühdiagnose; PA = Prozeßautomatisierung; WB = wissensbasiert; RT = real-time).

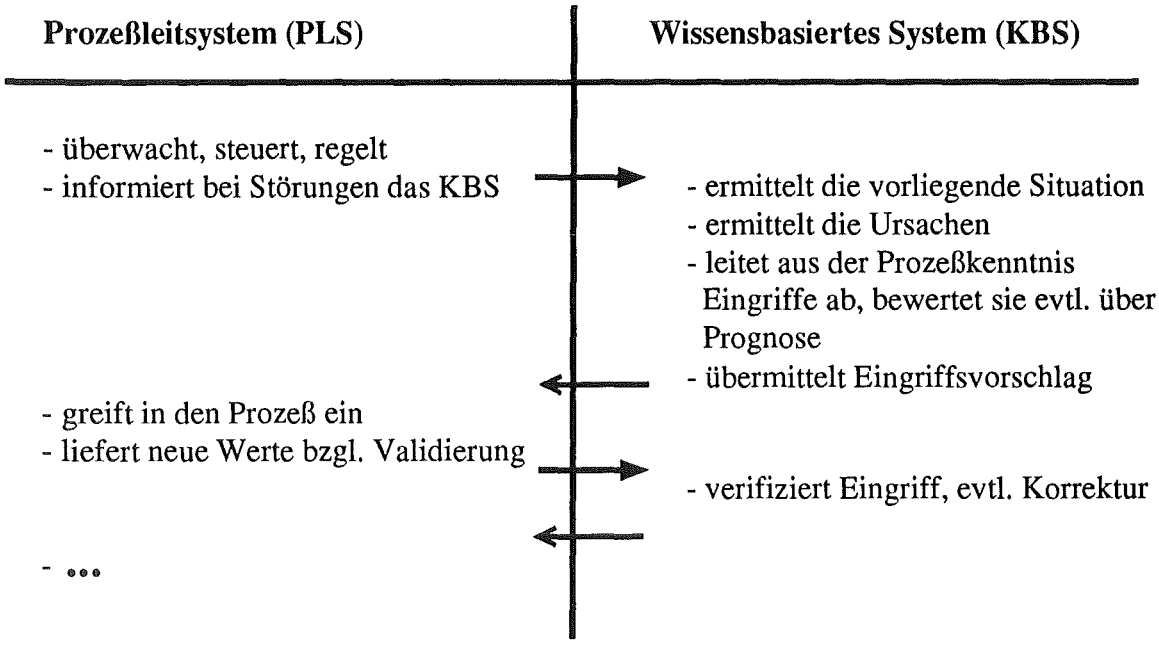
Die Vorgehensweise im Projekt INPRO ist die folgende:

- Ableitung allgemeiner und spezieller Anforderungen mit Überprüfung an einer realen Anlage (siehe /22/),
- Entwicklung einer Methodik der wissensbasierten Prozeßführung (Integration unterschiedlicher Aspekte, siehe /21/),
- Entwicklung eines Werkzeugs zur Umsetzung dieser Methodik und letztlich Erweiterbarkeit bzgl. der automatischen Übernahme von Wissen in Form einer Modifikation der vorhandenen Strukturen.

Der direkte Schwerpunkt der Arbeiten wird auf den ersten beiden Punkten liegen, allerdings sollte das Werkzeug die Aspekte des letzten Punktes berücksichtigen. Der zu bearbeitende Aufgabenbereich, die Methodik und die zu entwickelnden Werkzeuge sollten und müssen dabei aber auf einige wesentliche Strukturen beschränkt bleiben. Der Formalismus zur Darstellung und Verarbeitung von heuristischem Wissen sollte flexibel, aber ausdrucksmächtig und effizient sein. Hierzu bieten sich regelorientierte Methoden an.

Die Reihenfolge der Arbeiten hinsichtlich Komplexität und Bearbeitung sind in der Abb. 15 dargestellt.

Die Leistungsfähigkeit der Prozeßleitsysteme im klassischen Bereich soll nicht neu abgebildet, sondern integriert werden. Die Aufgabenteilung und das Interaktionsmuster kann beim Einsatz eines Expertensystems wie im folgenden Bild dargestellt aussehen. Beim Einsatz neuronaler Netze ergeben sich andere Strukturen und Interaktionsmuster.



8 Werkzeuge

Eine Integration der dargelegten Methoden zur Modellierung von Objekten, Situationen und deren gegenseitige Bezüge muß eine hohe Flexibilität in der Formulierung und Verarbeitung, die Möglichkeit der Vererbung von allgemeinen Eigenschaften, eine prioritätengesteuerte Ausführung und Operationen bzgl. der Klassifizierung, Abstandsberechnung von Instanzen zu Klassen und ähnliches mehr haben. Ein solche Flexibilität und Ausdrucksmächtigkeit kann durch eine interpretativ verarbeitete Sprache mit einer zugehörigen Entwicklungsumgebung erreicht werden. Dabei sind alle sprachlichen Operationen als Interpreterfunktionen über ein Laufzeitsystem zu implementieren. Dadurch ist eine Erweiterung der definierten Situationen oder der mathematischen Funktionen zur Laufzeit möglich.

Das K_advice-System ist als eine Komponente im Gesamtkonzept im wesentlichen fertiggestellt. Es dient primär zur mathematischen Modellierung dynamischer Prozeßkomponenten (Entwicklung und Test), sowie zur leittechnisch integrierten und verteilten Echtzeitsimulation (Beobachter) bzw. zur Prognose (Vorhersage) des dynamischen Verhaltens. Der Schwerpunkt des K_advice-Systems liegt mehr im klassischen numerischen Bereich, der Schwerpunkt der im Rahmen des Projektes INPRO neu zu entwickelnden Komponente dagegen mehr im wissensbasierten Bereich.

8.1 Das K_advice-System

8.1.1 Allgemeines

Die Modellierung komplexer dynamischer und stark verkoppelter Systeme erfordert eine umfangreiche Rechnerunterstützung. Der Gesamtvorgang der Modellierung und Simulation (model life cycle) kann dabei in Phasen unterteilt werden, welche dem Wissensstand einzelner Benutzergruppen entsprechen. Ausgehend von der mathematischen Beschreibung von elementaren Vorgängen werden Modellbausteine als Basiskomponenten/Schablonen umfangreicher Modellkomplexe definiert. Somit ist ein modularer und hierarchischer Aufbau komplexer Modelle möglich. Dieser Aufbau ist durch eine graphische Oberfläche optimal unterstützbar.

Basismodelle und benutzerdefinierte Modellkomplexe werden in einer Bibliothek abgespeichert. Parametervariationen und direkte Modellvergleiche sind leicht durchführbar. Simulationsergebnisse werden über einen Dialogprozeß on-line dargestellt, Eingriffe sind direkt möglich. Die Ergebnisse weden in Form von Zeitreihen über ein Managementsystem verwaltet und können später beliebig analysiert werden.

Das K_advice-System, /14/, ist eine interaktive, graphisch orientierte Modellierungs- und Simulationsumgebung. Die Operationen orientieren sich an den verschiedenen Phasen im Zyklus einer Simulation.

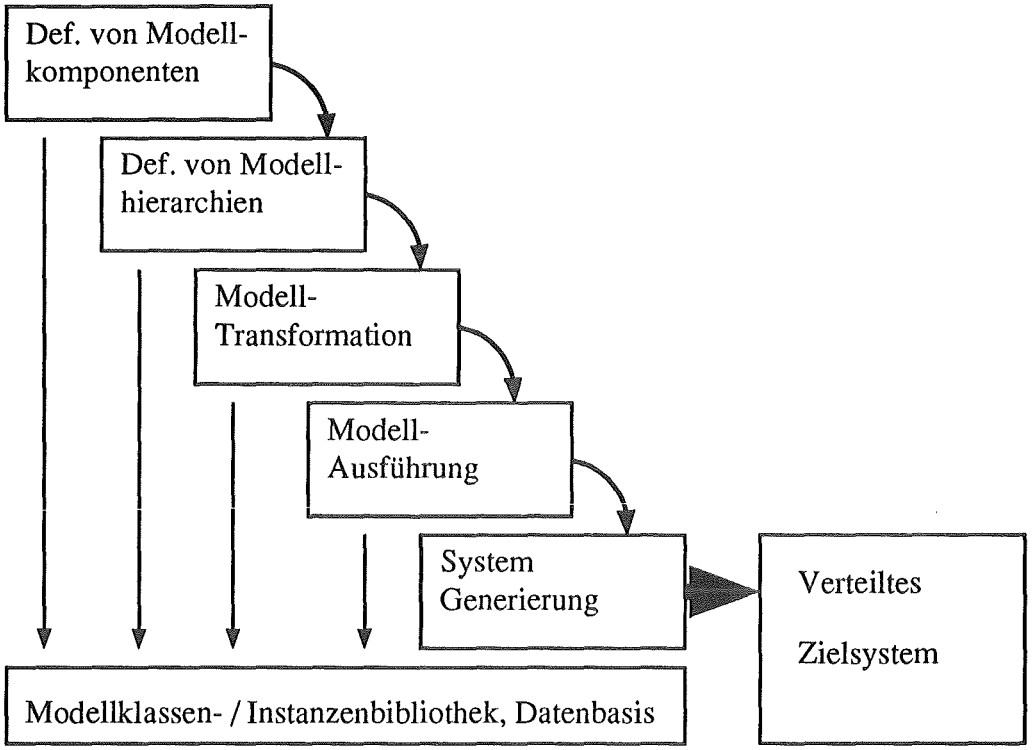


Abbildung 16

8.1.2 Die Graphische Oberfläche GAP / XGAP

Die Operationen zur Modellierung und Simulation eines Systems sind umfangreich und zum Teil kompliziert. Eine sprachorientierte Schnittstelle bietet dem Benutzer zu wenig Unterstützung. Für das K_advice-System wurde eine graphisch interaktive Oberfläche geschaffen, die alle notwendigen Interaktionselemente besitzt.

Alle Auswahlvorgänge werden per Maus bezogen entweder auf graphische Modellkomponenten oder auf Menüs ausgeführt. Kommandoeingaben erfolgen über die Auswahl von Kommandos aus einer Kommandoleiste. Verbindungen zwischen Modellkomponenten erfolgen über die Selektion von Ausgangs- und Eingangselement und Angabe der Umlenkpunkte der Verbindungslinie. Die Modellierung erfolgt hierarchisch, hierzu werden Rahmenmodelle (frames) zur Aggregation von Modellkomponenten (Basismodell oder Modellhierarchie) angeboten.

8.1.3 Modellierung

Die Modellierung erfolgt auf zwei Ebenen. Auf der mathematischen Ebene werden die Größen und ihre Beziehungen in Form von Gleichungen (physikalisch/chemisch) eingegeben. Die Kopplungen werden über Ein- und Ausgangsströme definiert. Die Benutzereingabe wird in eine interne deskriptive Form gebracht und in einer Modellbank abgespeichert. Die

in dieser Ebene definierten Modelle dienen als Schablonen für echte Modellteile (Instanzen). Unterschieden wird in diskrete und kontinuierliche Modelle, bei kontinuierlichen Modellen in die Art der Gleichungen (ODE, PDE) und dann noch einmal in explizite und implizite Gleichungen. Eine Matrizen-orientierte Beschreibung erfolgt nicht, da nichtlineare Systeme nicht in Matrixform darstellbar sind und lineare Systeme oft nur schwach besetzte Matrizen besitzen. Momentan wird die Modellierung kontinuierlicher Systeme auf der Basis expliziter Differentialgleichungen mit explitziten Parameterfunktionen voll unterstützt. Eine Erweiterung auf die gesamten Modellklassen ist geplant. Für die definierten Schablonen wird auf Benutzerwunsch Ada-Quelltext generiert, der automatisch in das System eingebunden wird. Danach stehen die neuen Modellschablonen als ausführbare Funktionen zur Verfügung.

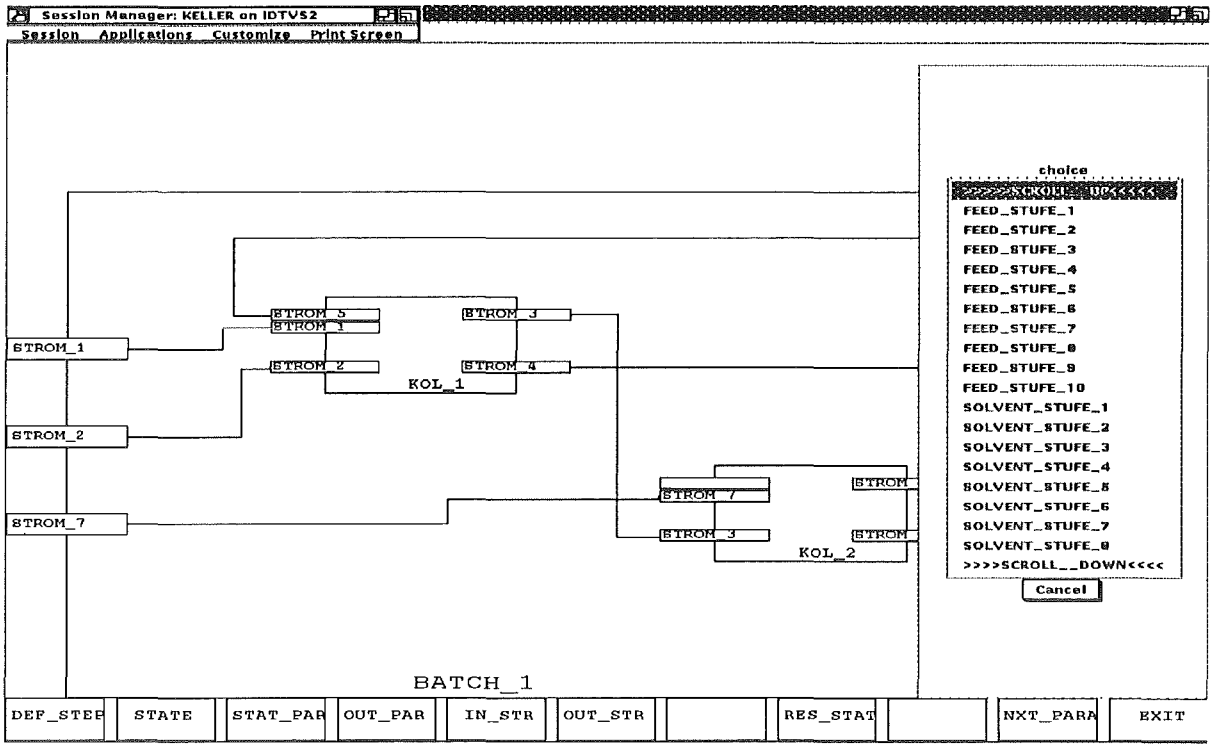


Abbildung 17

Die nächste Ebene dient zur Erzeugung von umfangreichen Modellkomplexen auf der Basis der definierten Schablonen (Modellkomponenten). Diese Modellierung wird als graphisch interaktive Konfigurierung bezeichnet, da sie fast vollständig mit Maus und Menü erfolgt. Modellkomponenten werden über eine Menüauswahl erzeugt, plaziert und mit anderen verbunden. Die Verbindung erfolgt über die Selektion der entsprechenden Ein- und Ausgangselemente. Mehr als eine Modellkomponente wird immer von einem Rahmen umfaßt, der die nach außen gehenden Verbindungen in die nächste Hierarchiestufe transportiert. Damit ist eine flexible und modulare Modellierung möglich.

8.1.4 Simulationslaufzeitsystem

Das Laufzeitsystem für die Simulationsexperimente wird jeweils dynamisch über Ada-Tasks erzeugt. Bei der Simulation kann in virtuelle und Echtzeit unterschieden werden. Für die Echtzeitsimulation wird ein RK4-Verfahren eingesetzt, das bei veränderlichen Eingangsgrößen die Zwischenschritte explizit mit den jeweils neuesten Eingangswerten durchführt.

Ein Gesamtmodell wird auf der Basis vorübersetzter Basismodellfunktionen gerechnet. Pro Basiskomponente des Gesamtmodells wird eine Gruppe von drei Tasks, eine **input**-, eine **output**- und eine **compute**-Task erzeugt. Die compute-Task rechnet die Basismodellkomponente entsprechend den zeitlichen Randbedingungen in Echtzeit durch. Die output-Task übernimmt die Ergebnisse und stellt sie den verkoppelten Modellkomponenten zur Verfügung. Die input-Task berechnet zum aktuellen Integrationszeitpunkt die notwendigen Eingangsdaten. Hierzu übernimmt sie von den output-Tasks die Werte in einen internen Puffer und extrapoliert auf den jeweiligen Integrationszeitpunkt. Aufgrund der Kommunikationsverzögerung sind die Ergebnisse der Teilmodelle nur verzögert verfügbar, das Verfahren ist aber für Systeme mit Tiefpaßverhalten stabil.

8.1.5 Analysekomponente

Prozeßdaten in Form von Meßreihen/Zeitreihen können in das K_advice-System eingelesen und archiviert werden. Die Analysekomponente erlaubt die Auswahl, Verarbeitung und Darstellung beliebiger zeitlicher Ausschnitte dieser Zeitreihen in Kombination mit intern erzeugten Daten (Simulationsergebnissen). Filterverfahren können zur Glättung angewandt werden. Zur besseren Darstellung können Zeitreihen als Kurven beliebig gedehnt und auch gegeneinander verschoben werden (Abb. 18).

8.1.6 Erweiterungen / Übertragbarkeit

Das K_advice-System kann um diskrete Modelle und andere mathematische Modelle erweitert werden. Die Struktur ist hierfür vorbereitet. Im deskriptiven Teil müssen die Eigenschaften definiert und vom Benutzer gesetzt werden. Das Laufzeitsystem muß um die entsprechenden Funktionen erweitert werden. Im Rahmen der momentanen Restrukturierung werden Grundfunktionen hierzu implementiert. Für spezielle Problemstellungen bei fixiertem Modell, bei denen nur Parametervariationen notwendig sind, kann ein anwendungsspezifisches Zielsystem generiert werden. Diese Komponente ist jedoch nur teilweise implementiert.

Das K_advice-System ist in Ada, die graphische Schnittstelle auf der Basis von GKS, Level 2c oder X-Windows/Motif implementiert. Eine Übetragung auf Unix-Plattformen ist einfach durchzuführen. Da auf PC-Ebene (Intel 386) leistungsfähige und validierte Ada-Compiler verfügbar sind, ist eine Portierung auf PC-Systeme möglich. Eine Anpassung der Schnittstelle auf die jeweilige graphische Plattform (MS-Windows, OS2-Presentation Manager, usw.) ist jedoch nötig.

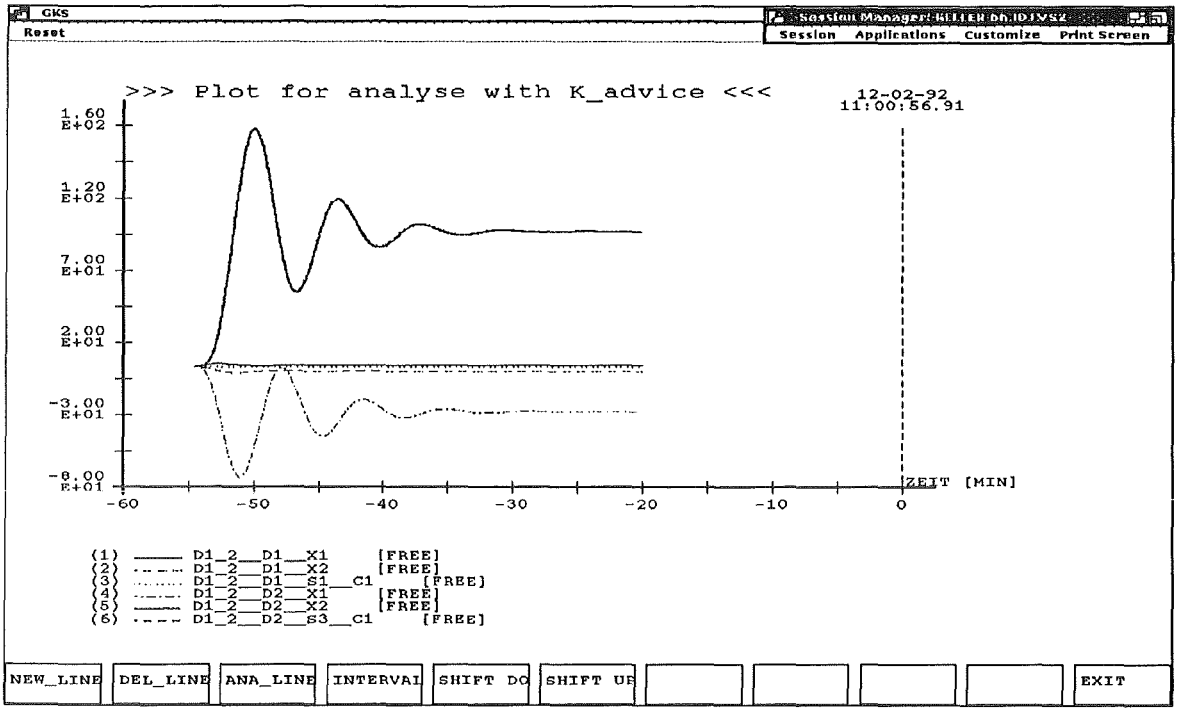


Abbildung 18

8.2 Sprache 1

8.2.1 Allgemeines

Im Gegensatz zum methodischen Zugang beim Entwurf von prozeduralen Programmen fehlt dieser bei wissensbasierten Programmsystemen. Die Strukturierungsmittel einer prozeduralen Sprache wie Ada finden sich in wissensbasierten Systemen aufgrund der fehlenden Sprachkonstrukte nicht immer wieder. Das methodische Vorgehen beim Entwurf ist ebenfalls ein anderes (z. B. die Klassendefinition bei objektorientierten Sprachen).

Ein weiteres Manko sind die fehlenden Sprachkonstrukte für Echtzeitanweisungen und für die Prozeßein-/ausgabe. Eine prozeßorientierte Verarbeitung mit nebenläufiger Ausführung, unter Verwendung von Kommunikationsmechanismen sowie Prioritäten, findet ebenfalls nicht statt.

Ein weiterer Gesichtspunkt stellt die verteilte Ausführung von prozeßleittechnischen Aufgaben dar. Eine wissensbasierte Verarbeitung ist sicherlich eine höherwertige Funktion, die örtlich verteilte Charakteristik des realen Prozesses mit seiner inhärenten Parallelität und den aus der Prozeßdynamik sich ergebenden Echtzeitanforderungen, müssen sich in einem

zugeordneten Programmsystem widerspiegeln. Sprachkonstrukte zur verteilten Ausführung (Migration, Kommunikation) sollten verfügbar sein.

Bei bestimmten modellbasierten Methoden können während dem Betrieb Inkonsistenzen zwischen den im Modell getroffenen Annahmen (mathematische Formulierung) und den tatsächlichen, im Prozeß vorliegenden Gegebenheiten auftreten. Im mathematischen Modell fehlende Terme, sollten zur Laufzeit einfügbar sein. Dies erfordert sowohl die bisher vorhandene numerische Verarbeitung von mathematischen Modellen, als auch deren symbolische und term-orientierte Verarbeitung.

Technische Apparate gleicher Funktionalität ähneln sich in ihrem internen Aufbau und in ihren Eigenschaften. Der Unterschied besteht in der speziellen Erweiterung von Eigenschaften oder der Funktionalität. Es bietet sich daher an, objektorientierte Klassifikations- und Vererbungsmechanismen zu verwenden. Für einfache Objekte (einfacher Typ) kann das prozedurale Schema beibehalten werden. Objekte höheren Abstraktionsgrades sollten als autonome, abstrakte Klasseninstanzen prozeßorientiert ausgeführt werden. Dies ermöglicht eine nebenläufige, prioritäts- und zeitgesteuerte Ausführung.

Das heuristische Wissen eines Bedieners sollte in Form von Regeln dargestellte werden. Dies hat gegenüber anderen Wissensdarstellungs- und Verarbeitungsformen die Vorteile einer höheren Anschaulichkeit, einer einfacheren Umsetzbarkeit, sowie die Möglichkeit einer einfachen Integration von Kontrollwissen über die Angabe einer direkten Nachfolgeregel (siehe z. B. /28/). Außerdem können Regelmengen inkrementell erweitert werden.

8.2.2 Sprachkonzept

Der Sprachentwurf geht von einer objektorientierten Beschreibung der technischen Prozesse und des zur Führung notwendigen Wissens aus. Die Definition von Klassen auf der Basis von elementaren Typen mit der Möglichkeit von Spezialisierungen (ersetzen von Klassenteilen) und der allgemeinen Vererbung (Erweiterung von Klassen) stellen die Grundelemente zur Beschreibung (Programmierung) dar. Die erzeugten Objekte werden nebenläufig ausgeführt und ermöglichen eine physikalische oder logische Parallelität. Durch die Vergabe von Prioritäten können Objekte eine unterschiedliche Gewichtung bzgl. der Ausführung erhalten und damit der unterschiedlichen Dringlichkeit von z. B. Situationsbewertungen entsprechen. Objekte implementieren algorithmische und heuristisches Wissen. Heuristisches Wissen wird in Form von Regeln dargestellt und verarbeitet. Durch die objektorientierte Struktur entstehen einzelne, gekappselte Regelmengen. Die Schlußfolgerungen über der Regelmenge eines Objektes sind durch die Kommunikationsbeziehungen nach außen mit der Ausführung anderer Objekte gekoppelt. Eine streng monotone Ableitung ist über externe Beziehungen hinweg nicht immer möglich. Der Ableitungspfad als Folge von einzelnen Regeln muß in Teile ohne externe Beziehungen gegliedert werden.

Das Modell des Regelinterpreters berücksichtigt externe Referenzen und die nebenläufige (prozeßorientierte) Ausführung von Objekten. Im Sinne einer Reaktion auf Prozeßereignisse werden die Regeln datengetrieben/zielorientiert als notwendige Handlungssequenz abgear-

beitet. Ihre Verknüpfung ist vorwärtsverkettet. Neben einfachen Anweisungen können im Aktionsteil einer Regel auch Funktions-/Prozeduraufrufe oder Kommunikationsoperationen auftreten. Kontrollwissen ist in Form von expliziten Folgeregeln möglich. Unsicheres Schließen kann über die Klassenoperationen realisiert werden.

8.2.2.1 Regeldarstellung

Ein erster, unter einem pragmatischen Gesichtspunkt sinnvoller Ansatz zur Darstellung extrahierten Wissens eines Bedieners ist die Regelform. Im allgemeinen handelt es sich um Handlungen, die in bestimmten Situationen durchzuführen sind. Regeln werden mit Bedingungs- und Aktionsteil wie folgt formuliert:

```
<rule_spec>

-> RULE rule_IDENT;

<bool_expr>;

ACTION <rule_statement> {;<rule_statement>};

[SUCCESSOR rule_IDENT {,rule_IDENT}];
       /* rule_IDENT nicht rekursiv */

END RULE;
```

Eine Regel besteht aus ihrem Namen (rule_IDENT), einem logischen Ausdruck (<bool_expr>) als Bedingungsteil und dem Aktionsteil. Der Aktionsteil kann eine oder mehrere Aktion(en) (<rule_statement>) enthalten. Optional ist die Angabe einer oder mehreren Nachfolgeregeln (rule_IDENT) möglich. Hierdurch kann Wissen um den Ablauf von bestimmten Handlungsfolgen als Kontrollwissen eingebracht werden. Der Name der Folgeregel darf nicht identisch mit dem der Regel sein.

8.2.2.2 Regelausführung

Die Beschreibung von Wissen in Regelform ist statisch. In einer prozeßleittechnischen Umgebung müssen zur Ausführung der Regeln bestimmte Randbedingungen erfüllt sein. Für den Regelinterpreter gelten daher folgende Forderungen:

- Kontrollwissen soll in Form von Folgeregeln eingebracht werden. Durch die Angabe von Folgeregeln soll steuernd auf den Regelinterpreter eingewirkt werden.
- Regelverarbeitung soll vorwärtsverkettend erfolgen. Es soll die Möglichkeit einer Rückwärtsverkettung in der Struktur vorgesehen werden. Für Steuerungszwecke

kommt hauptsächlich die Vorwärtsverkettung in Frage, da die "Ist-Zustände" (Daten) meist bekannt sind, auf die durch eine steuernde Aktion reagiert werden soll.

- Der Regelinterpreter muß reentrant aufrufbar sein, da die Ausführung der Objekte prozeßorientiert erfolgt, um konkurrierende Situationen mit eventuell unterschiedlicher Dringlichkeit korrekt bearbeiten zu können (Kontext sichern).

Festlegungen für die Lösungsstrategie des Regelinterpreters:

- Alle Regeln der Konfliktmenge werden ausgeführt, sofern sie nicht durch die Auswirkungen externer Referenzen vor ihrer Ausführung wieder ungültig werden.
- Für Prozeßsteuerungszwecke reicht nicht eine Lösung (eine Aktion), sondern es müssen alle möglichen Lösungen gefunden (Aktionen ausgeführt) werden.
- Die Ableitungen sollen
  - aktuell und
  - spezifisch sein.

Das heißt für die Konfliktlösungsstrategie:

- aktuell:
  Die Regeln mit den neusten Daten sollen bevorzugt abgearbeitet werden.
- spezifisch:
  Die Regeln mit den spezifischsten Bedingungen, d. h. die Bedingungen mit den meisten Termen, sollen bevorzugt abgearbeitet werden.

8.2.2.3 Konfliktlösungsstrategie

Die Reihenfolge der Regeln kann anhand der folgenden Kriterien bestimmt werden:

(1) explizit angegebene Folgeregeln
Wenn mehrere Folgeregeln angegeben sind, so wird die Reihenfolge wie vorgegeben verwendet. Wenn die erste Folgeregel nicht feuert (d. h. nicht Teil der Konfliktmenge ist), wird die zweite versucht u.s.w..

(2) Alter der Daten
Wenn die Regeln, die auf den neuesten Daten beruhen, bevorzugt abgearbeitet werden, ist gewährleistet, daß ein begonnener Ableitungspfad zielstrebig weiter verfolgt wird, bis er keine neuen Erkenntnisse mehr liefert. Hier werden Daten bevorzugt, die sich erst kürzlich ergeben haben: Schlußfolgerungen, neu eingelesene Daten etc..

(3) Komplexität (Spezifität) der Regeln
Wenn mit Hilfe der Kriterien (1) und (2) noch keine Reihenfolge festgelegt werden kann, werden die Regeln mit dem komplexeren Bedingungsteil bevorzugt, da man davon ausgehen kann, daß diese Regeln die spezifischeren sind. Als Auswahlkriterium dient hier einfach die Anzahl der Bedingungsterme.

(4) Externe vor interne Referenzen
Daten, die aus externen Referenzen folgen, spiegeln die Realität besser wider, als solche, die auf internen Referenzen beruhen. Daten aus internen Referenzen sind Daten, die auf ältere Daten zurückzuführen sind.

Diese Kriterien sind in der genannten Reihenfolge anzuwenden. Sollte damit keine eindeutige Reihenfolge bestimmt werden können, werden die Regeln in der Reihenfolge ihres Auftretens (nach ihrer Nummer "RNr") abgearbeitet. Dies bedeuted, daß die Regeln eine gewisse Priorität durch ihre Anordnung erhalten können. Dies ist aber nur bedingt möglich, da Regeln, die zur Laufzeit neu erzeugt werden, nur hinten angehängt werden, und man keinen Einfluß mehr auf ihre Reihenfolge nehmen kann.

8.2.3 Beispiel "Kompressor"

Um die Fomrulierungsmöglichkeiten mit Regeln darszustellen, wird ein Kompressor mit zwei Druckbehältern (Druck: D1 und D2), einer Pumpe (P) mit Kontrolleuchte (L), sowie drei Ventilen (V1, V2, V3) als Beispiel benutzt. V1 kontrolliert den Druckausgleich zwischen den Druckbehältern. V2 wird als Überdruckventil für den zweiten Druckbehälter verwendet. Das Ventil V3 wird über einen Schalter (S) betätigt. Es soll sich aber nur dann öffnen lassen, wenn der Druck D2 in den zulässigen Grenzen liegt.

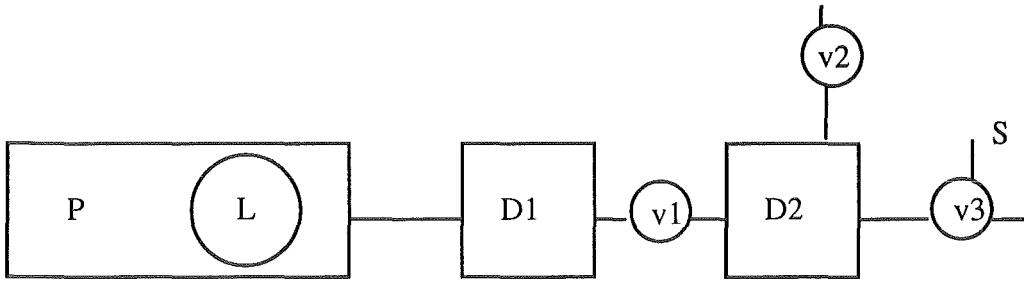


Abbildung 19

8.2.3.1 Objekt Pumpe P

```
RULE R1;
D1 < D1_min und status = aus;      // Druck zu niedrig
ACTION
einschalten(P);      // Prozedur zum Einschalten der Pumpe
einschalten(L);      // Prozedur zum Einschalten der Lampe
EXIT;      // beenden
END_RULE;

RULE R2;
D1 > D1_max und status = ein;      // Druck zu hoch
ACTION
ausschalten(P);      // Prozedur zum Ausschalten der Pumpe
ausschalten(L);      // Prozedur zum Ausschalten der Lampe
EXIT;      // beenden
END_RULE;
```

8.2.3.2 Objekt Ventil V1

```
RULE R1;
D1 > D2;      // Druck in Kessel 1 höher als in Kessel 2
ACTION
status := enable;      // Öffnen ermöglichen
SUCCESSOR R4;      // ggf. Ventil öffnen
END_RULE;

RULE R2;
D2 > D1;      // Druck im Kessel 2 höher als in Kessel 1
ACTION
status := disable;      // Öffnen unterbinden
SUCCESSOR R5;      // ggf. Ventil schließen
END_RULE;

RULE R3;
D2 > D2_max;      // Überdruck in Kessel 2
ACTION
status := disable;      // Öffnen unterbinden
SUCCESSOR R5;      // ggf. Ventil schließen
END_RULE;

RULE R4;
(status = enable) and (D2 < D2_max);
ACTION
oeffnen(V1);      // Prozedur zum Öffnen des Ventils
EXIT;      // beenden
END_RULE;
```

```
RULE R5;
status = disable;
ACTION
schliessen(V1);     // Prozedur zum Schließen des Ventils
EXIT;     // beenden
END_RULE;
```

8.2.3.3 Objekt Ventil V2

```
RULE R1;
D2 > D2_max;     // Überdruck in Kessel 2
ACTION
oeffnen(V2);     // Prozedur zum Öffnen des Ventils
EXIT;     // beenden
END_RULE;

RULE R2;
D2 < D2_max;     // kein Überdruck in Kessel 2
ACTION
schliessen(V2);     // Schließen des Ventils
EXIT;     // beenden
END_RULE;
```

8.2.3.4 Objekt Ventil V3

```
RULE R1;
(D2 < D2_max) and (D2 > D2_min);     // Soll-Druck erreicht
ACTION
status := enable;     // Öffnen ermöglichen
END_RULE;

RULE R2;
(status = enable) and (S = auf);     // Öffnen möglich und Schalter auf Auf
ACTION
oeffne(V3);     // Prozedur zum Öffnen des Ventils
EXIT;     // beenden
END_RULE;

RULE R3;
(D2 > D2_max) or (D2 < D2_min);     // Soll-Druck nicht erreicht
ACTION
status := disable;     // Öffnen unterbinden
SUCCESSOR R4;     // ggf. Ventil schließen
END_RULE;
```

```
RULE R4;
(status = disable) or ((status = enable) and (S = zu));
ACTION
schliessen(V3);     // Ventil schließen
EXIT;     // beenden
END_RULE;
```

Die Ausführung der oben formulierten Regelmengen erfolgt für jedes Objekt getrennt nach den definierten Randbedingungen/Anforderungen. Ein Informationsaustausch zwischen den Regelmengen (Objekten) erfolgt über Kommunikationsoperationen.

8.3 Das C^3R-System zur automatischen Wissenserfassung

8.3.1 Allgemeines

Unter bestimmten Randbedingungen, wie z. B. schlechter Erfaßbarkeit der Beeinflußungsgrößen (z. B. die Zusammensetzung des Mülls bei der Müllverbrennung), Nichtmeßbarkeit interner Zustände, fehlender Kenntnis der physikalischen/chemischen Zusammenhänge oder Parameter, ist eine Reduktion der zu erkennenden Beziehungen auf elementare kausale Abhängikeiten an Stelle theoretischer mathematischer Modelle, der einzige Weg zu einer rechnergestützen Prozeßführung.

Vorliegende und im Betrieb optimierte heuristischer Modelle mehrerer Operateure beinhalten umfangreiches und somit für die Prozeßführung wichtiges Expertenwissen, das standardisiert und eingesetzt werden sollte. Hierdurch können technische Prozesse stärker im Bereich des optimalen Zustandes gefahren werden.

Die heuristischen Beziehungen/Modelle können über automatische Lernverfahren im Betrieb abgeleitet und dabei gleichzeitig validiert werden. Die Validierung ist hierbei auch eine Standardisierung und damit ein sicherheitstechnisch hoch einzuschätzender Prozeß. Um die Erklärbarkeit symbolischer Verfahren, aber auch die Lernfähigkeit neuronaler Systeme bzgl. kausalen Abhängikeiten einsetzen zu können, sind beide Fähigkeiten in einem System zu integrieren. Das C^3R-System (***C***hilden's ***c***ognitive learning behavior for ***c***ausal ***r***easoning about dynamic systems) ist hierzu ein im IDT entwickelter Ansatz /31/. Die **Sprache 1** stellt dabei die grundlegenden Konstrukte und Methoden bereit.

8.3.2 Das Ziel des C^{3}R-Systems

Das **C^{3}R-System** stellt einen Ansatz zur automatischen Ableitung von Bedienerwissen (heuristische Modelle) und zur Ableitung von Bedienstrategien. Im C^{3}R-System werden die Fähigkeiten neuronaler Netze (Lernfähigkeit, Musterklassifikation) auf symbolische Art und Weise nachgebildet, um dadurch die Nachteile neuronaler Netze (Nicht-explizierbarkeit des Wissens, Unterscheidung in Trainings- und Verarbeitungsphase) zu vermeiden und die Vorteile symbolischer Verarbeitungsmechanismen (Explizierbarkeit des Wissens und inkrementelle Lernfähigkeit) zu nutzen. Diese Nachbildung ermöglicht es kausale Zusammenhänge aus dem Verhalten des technischen Prozesses abzuleiten.

Folgende Schritte bzw. Operationen sind in Bearbeitung:

- Automatisches Lernen von gegenseitigen Abhängigkeiten zwischen Prozeßgrößen,
- Strukturierung der Beziehungen nach dem Grad der Kopplung und der zeitlichen Differenz zwischen Wertefolgen,
- Lernen der Übergangsbeziehungen in Form von wertefolgen und deren Abstraktion,
- Erkennen von Sollwertabweichungen,
- Berechnen der Störungsursachen,
- Berechnen der einer Störung zuordenbaren exogenen Größen (beeinflußbare Einwirkungen),
- Berechnen der notwendigen Zustandsänderungen zur Ausregulierung der Störungsauswirkungen/-ursachen,
- Meldung von Eingriffsvorschlag an den Bediener,
- Berücksichtigung von gestörten Komponenten als Randbedingung bei der Berechnung der Eingriffsvorschläge,
- Parallelisierung der Situationsbearbeitung für mehrere Störungen.

Die gegenseitigen Abhängigkeiten von Prozeßgrößen können anhand der Meßreihen automatisch gelernt und entsprechend dem Grad der Kopplung strukturiert werden. Anhand dieser strukturierten Prozeßgrößen sind lokale Übergangsbeziehungen ableitbar. Die exogenen Größen werden anhand dem Kopplungsgrad berechnet. Aufgrund der Kenntnis des regulären Prozeßverlaufes (Sollwerte) können Störungen erkannt und auf dieser Grundlage die Störungsursachen und die zu beeinflußenden exogene Größen berechnet werden. Nach der Bestimmung der notwendigen Zustandsübergänge, können dem Operateur on-line Vorschläge für Handlungsfolgen zur Beseitigung von Störungen und zur Prozeßführung innerhalb eines vorgegebenen Toleranzbereiches gegeben werden. Die für das Gesamtsystem beschriebene Vorgehensweise kann evtl. auch auf gestörte Komponenten übertragen werden, indem man deren Randbedingungen als nicht änderbare Zustände und Kopplungsbeziehungen berücksichtigt.

Zur Realisierung der bisher dargestellten Ziele in einer verteilten Prozeßleitsystemumgebung sind parallele Problemlösungs- bzw. Situations-Verarbeitungsprozesse vorzusehen. Das Lernen erfolgt anhand der aktuellen Meßwerte und stellt einen inkrementellen Prozeß dar. Durch das inkrementelle Lernen ergibt sich automatisch eine on-line-Validierung des vorhandenen, bereits abgeleiteten Wissens. Die Aufnahme von neuem Wissen, bzw. die Änderung von vorhandenem Wissen erfolgt inkrementell, aber mit einer Trägheit über die Berücksichtigung der Historie (Verrechnung alter Werte). Stochastische Einflüße werden somit neben Änderungen unterhalb der dem System zugrundeliegenden Eigenzeit (Auflösung) gefiltert.

8.3.3 Das Konzept des C^3R-Systems

Die Ableitung kausaler Zusammenhänge erfolgt durch eine heuristische Vorgehensweise und ist daher nicht mathematisch exakt. Kausal gekoppelte Merkmale (Attribute) werden entsprechend dem Grad der Kopplung zu Gruppen, sogenannten **Clustern** zusammengefaßt.

Diese Cluster werden nach einem hierarchischen Prinzip mit Hilfe mengentheoretischer Operatoren zu konstruktiven Kategorien (Kategorien 1-ter Ordnung) weiterverarbeitet. Anhand des Kopplungsgrades werden aus diesen konstruktiven Kategorien rekursiv semantische Kategorien höherer Ordnung gebildet. Die Semantik spiegelt sich im Kopplungsgrad wider. Diese Operationen sind im C^3R-System auf der untersten, einer **symbolischen Systemebene** anzusiedeln, die ein **neuronales Verhalten** nachbildet.

Die somit entstandene Kategorisierungshierarchie entwickelt sich unter Berücksichtigung der Basisrelationen zum Klassenaufbau, zur Klassenvererbung, zu Attributabstand/Analogie und zur Zustandsänderung zu einer stark vernetzten hierarchischen Struktur. Durch die Berücksichtigung dieser Relationen wird diese flexibel und dynamisch. Sie paßt sich inkrementell, on-line an den technischen Prozeß an. Die Basisrelationen ermöglichen bei der Informationsverarbeitung beispielsweise den unmittelbaren Übergang einer auf Vererbung basierenden Vorgehensweise zu einer attributgesteuerten und vice versa. Auf dieser stark vernetzten hierarchischen Struktur operiert man auf Klassen, deren Instanzen, Attributen sowie deren Operationen. Sie stellt die mittlere, **kategorische Ebene** des C^3R-Systems dar. Durch eine Situationsklassifikation wird es möglich die für die Zustandsübergänge verantwortlichen Operatoren abzuleiten.

Die oberste Ebene des C^3R-Systems, die **Problemverarbeitungsebene**, umfaßt die Prozesse zur Verarbeitung der definierten, d. h. vorgegebenen Probleme. Es wird hier versucht zu einem gegebenen Problem, als Teil der realen Welt (des Objektverbundes), unter Einbeziehung vorhandener heuristischer Prozeduren Lösungen zu finden. Hierzu agiert das System auf einem kontext-sensitiven Ausschnitt der Struktur der kategorischen Ebene, einer sogenannten **Sicht**. Auf der Basis dieser Sicht kann z. B. anhand eines vorgegebenen (Ausgangs-) Zustandes, eines Zielzustandes und einem Repertoire an elementaren Problemlösungsmethoden eine Operatorfolge abgeleitet werden, die den gewünschten Übergang des Ausgangs- in den Zielzustand bewältigt. Um eine Bewertung des durch die Anwendung von Operatoren fortschreitenden Problemlösungsvorganges bzw. Lernprozesses vornehmen zu können werden alle durchgeführten Operationen in bezug auf ihre Eignung und Effizienz protokolliert und (neuronal adaptiv) auf die Struktur zurückgeführt.

Analogieschlüsse und damit das häufig anzutreffende Lernen durch Analogie /25/ werden auf der Grundlage benachbarter Klassen (auf der gleichen Ebene oder eine Ebene darüber) durchgeführt. Das Lernen findet im Grunde auf allen drei Systemebenen statt. Im Bezug auf die unterste Ebene handelt es sich hierbei um eine kontextbezogene Mustererkennung; erkannt wird ein elementares Muster bzgl. der (durch den Kontext) erwarteten Superklasse. Während es sich beim Lernen von Klassifikationen um den Aufbau einer Klassenhierarchie handelt, betrifft das Problemlösen den Nachweis der Evidenz von Methoden im Kontext.

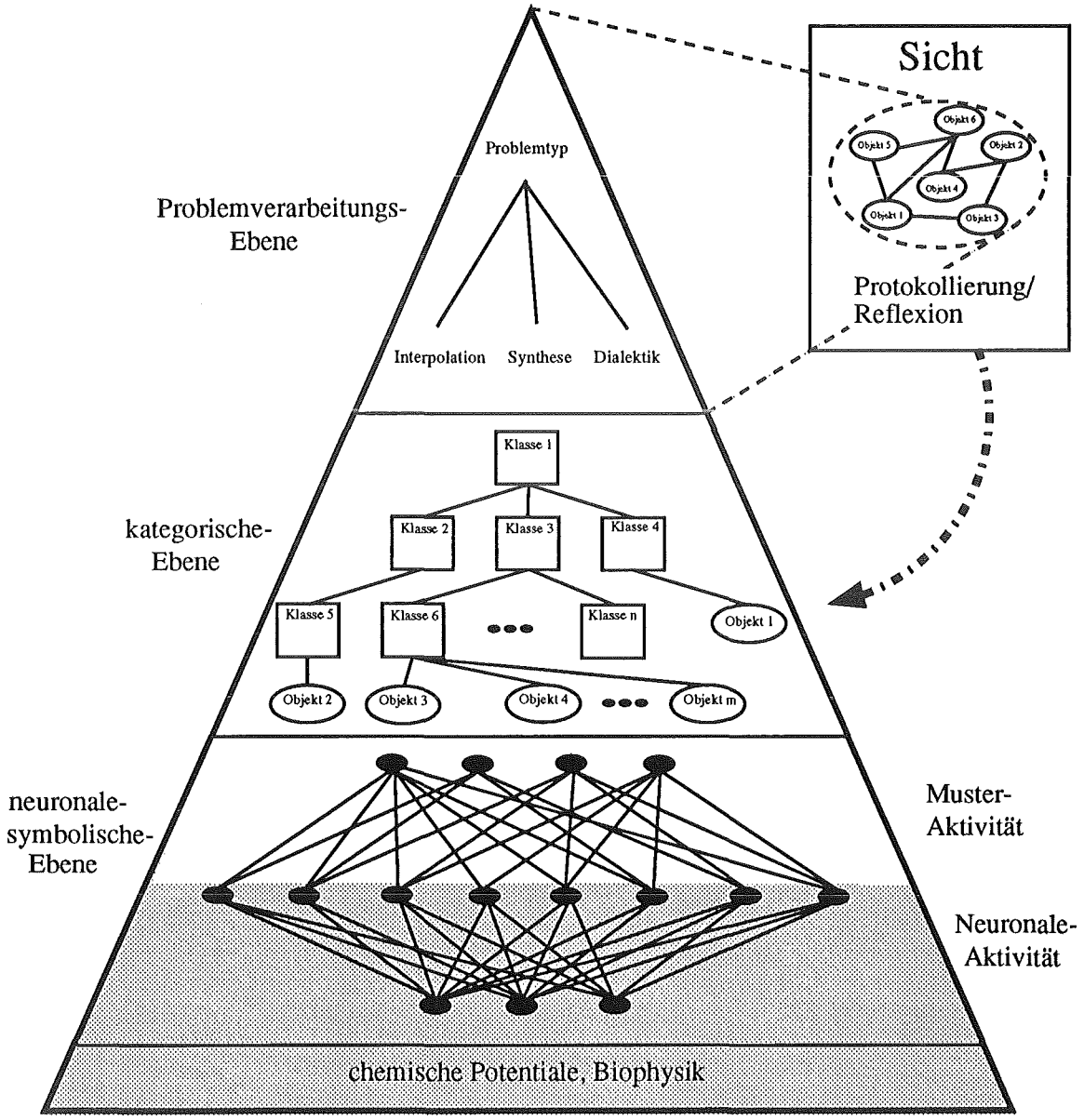


Abbildung 20

Generell basiert das Lernen auf den Informationen, die der technische Prozeß und der Bediener (Experte) liefern (gemessene Werte, Bedienerhandlungen etc.) und auf dem, dem System vorgegebenen Grundwissen.

Das vorgestellte C^3R-System stellt einen Ansatz dar, der das Ergebnis einer am neuronalen Modell orientierten Vorgehensweise zur Mustererkennung mit einer hierarchischen, stark vernetzten Darstellungsform, die auf dem **Konzept kognitiver Schemata** /15/ beruht und als Grundlage für Problemlösungsprozesse dient, verbindet.

Bei kognitiven Schemata handelt es sich um kognitive Strukturen, in denen allgemeines Wissen, d. h. sowohl deklaratives, als auch prozedurales Wissen repräsentiert ist. Kognitive Schemata sind aktive, miteinander in Wechselwirkung stehende Wissensstrukturen, die aktiv am Verständnis und der Anforderung einlaufender Informationen beteiligt sind und die Ausführung der Verarbeitungsoperationen steuern. Da es das Konzept kognitiver Schemata ermöglicht, daß Schemata über Schemata gebildet werden und daß das Wissen unterschiedliche, dynamische Interpretationen, durch die Sichten zuläßt, besticht der vorgestellte Ansatz insbesondere durch seine Flexibilität und Mächtigkeit.

Auf der Basis dieses Konzeptes wird es möglich sein, Operateurwissen automatisch abzuleiten und auf der extrahierten Struktur permanent zu validieren. Das abgeleitete Wissen wird dem Operateur über die Problemlösungsebene zur Führung des technischen Prozesses, insbesondere bei Störungen, zur Verfügung gestellt.

8.3.4 Stand/Planung

Im Rahmen des Projektes INPRO wurde das in Kapitel 7 dargestellte Konzept als Erweiterung der Konzepte und Methoden "klassischer" wissensbasierter Systeme erarbeitet (/15/, /30/).

Der Systementwurf sieht die folgenden drei Ebenen vor:

- Ebene kausaler Abhängigkeiten mit inkrementeller Clusterung,
- Aufbau der semantischen, kategorischen Struktur mit Transformationsbeziehungen,
- Bearbeitung von Situationen, Ableitung von Bedienstrategien.

Eine Präzisierung der inkrementellen Clusterungsansätze auf der Basis elementarer numerischer Methoden ist bereits erfolgt und es wird die Validierung dieser Ansätze anhand ausgewählter Beispiele angestrebt. Im Rahmen dieser Validierung erfolgt die Implementierung dieser Ansätze zur Clusterung, um dann die weiteren Arbeiten zur mittleren Ebene auf dieser Basis fortsetzen zu können (Beginn: ca. Sommer 1992).

9 Literaturüberblick, geordnet nach relevanten Fachbereichen

In diesem Kapitel wird ein Überblick über von uns ausgewählte und als relevant befundene Literatur aus den Bereichen

- Echtzeit-Anwendungen,
- Expertensysteme,
- Fuzzy Logik,
- Künstliche Intelligenz (allgemein),
- Maschinelles Lernen,
- neuronale Netze,
- Projekt INPRO

gegeben. Die Literaturhinweise erheben keineswegs einen Anspruch auf Vollständigkeit, sie sollen lediglich Hinweise für grundlegende und weiterführende Studien liefern.

9.1 Echtzeit-Anwendungen

/1/ Howard M. Forster, Simon R. Meadowcroft,
"The application of on-line expert systems to supervisory process control",
Measurement + Control, Vol. 21, Juli/August, 1988.

/2/ B. Freyermuth, R. Isermann, D. Neumann,
"Wissensbasierte On-Line-Fehlerdiagnose in der Fertigung",
Fabrik der Zukunft, Band 5, S. 2 - 5, 1990.

/3/ H. Früchtenicht et al.,
"Technische Expertensysteme: Wissensrepräsentation und Schlußfolgerungsverfahren",
Oldenbourg Verlag, München, 1988.

/4/ R. Isermann,
"Identifikation dynamischer Systeme",
Band I, Springer Verlag, Berlin, 1988.

/5/ R. Isermann,
"Wissensbasierte Fehlerdiagnose technischer Prozesse",
Automatisierungstechnik at, Heft 11, 1988.

/6/ M. Kerndlmaier,
"Technische Expertensysteme für Prozeßführung und Diagnose",
Oldenbourg Verlag, München, Wien, 1989.

/7/ Thomas J. Laffey, Preston A. Cox, James L. Schmidt, Simon M. Kao, Jackson Y. Read,
"Real-time knowledge-based systems",
AI Magazine, Spring 1988.

/8/ Thomas J. Laffey,
"The real-time expert",
Byte, Januar 1991.

/9/ Andreas Lübbert,
"Trends beim Einsatz von Realzeitrechnern für Forschung und Entwicklung in der Technischen Chemie",
Chem.-Ing.-Tech, 62, Nr. 6, S. 474 - 483, VCH Verlagsgesellschaft mbH, 1990.

/10/ Magdi S. Mahmoud, Shawki Z. Eid, Ahmed A. Abou-Elsoud,
"A real-time expert control system for dynamical processes",
IEEE Transactions on Systems, Man and Cybernetics, Vol. 19,
No. 5, September/Oktober, 1989.

/11/ R. L. Moore, L. B. Hawkinson, M. Levin, A. Hofmann, B. L. Matthews, M. H. David,
"An expert system for real-time process control",
In: D. Siram, R. A. Adey (Eds.),
"Knowledge based expert systems for engineering: Classification, education and control",
Proceedings Cambridge, Ma., 4. - 7. 8. 1987, Southhampton, 1987.

/12/ E. Mühlenfeld,
"Automatisierung und Diagnose von Prozessen ohne mathematisches Modell",
Automatisierungstechnik at, Heft 9, 1989.

/13/ Tilo Pfeifer, Robert Grob, Rudolf Schmidt,
"Expertensysteme für die SPC",
QZ 36, Carl Hanser Verlag München, S. 432 - 436, 1991.

/14/ Peter Raulefs,
"Computational architectures for computer-integrated engineering an manufacturing: An artificial intelligence perspective",
In: D. Metzing (Ed.),
"GWAI-89, 13th German Workshop on Artificial Intelligence",
Springer Verlag, 1989.

/15/ R. Soltysiak,
"HEPROX, eine Expertensystemshell für Prozeßführungsaufgaben",
Automatisierungstechnische Praxis atp, Heft 2, 1989.

/16/ U. Sill, P. Elzer, H. Roghmanns,
"Informationsaufbereitung und -darstellung zur Störungserkennung und -behebung",
In: "Rechnergestützte Prozeß- und Betriebsführung im Kraftwerk",
ETG-Fachbericht 28, VDE-Verlag GmbH, Berlin, 1989.

/17/ H. B. Verbruggen, P. M. Bruijn, A. K. Krijgsman,
"Towards the practical application of knowledge-based systems in real-time control",
Interkama '89, Proceedings.

9.2 Expertensysteme

/1/ Javier Aracil, Anibal Ollero, Alfonso Garcia-Cerezo,
"Stability for the global Analysis of expert control systems",
IEEE Transactions on Systems, Man and Cybernetics, Vol. 19,
No. 5, September/Oktober, 1989.

/2/ P. Bathelt, M. Kerndlmaier, T. Zink,
"Expertensysteme für die technische Diagnose",
In: M. Thoma, G. Schmidt (Eds.)
Interkama '89, Springer Fachberichte Messen Steuern Regeln 14, 1986.

/3/ B. Bieker,
"Wissenserwerb für eine einfache Experten-Regelung",
Automatisierungstechnische Praxis atp, Heft 9, 1986.

/4/ Werner Karbach, Marc Linster,
"Wissensakquisition für Expertensysteme - Techniken, Modelle, Softwarewerkzeuge",
Carl Hanser Verlag, München, Wien, 1990.

/5/ I. Kupka,
"Theoretische Grundlagen der Expertensysteme",
Automatisierungstechnik at, Heft 3, 1988.

/6/ Karl Kurbel, Wolfram Pietsch,
"Expertensystem-Projekte: Entwicklungsmethodik, Organisation und Management",
Informatik-Spektrum 12, S. 133 - 146, Springer Verlag, 1989.

/7/ Frank Puppe,
"Einführung in Expertensysteme",
Springer Verlag, Berlin, 1988.

/8/ Frank Puppe,
"Problemlösungsmethoden in Expertensystemen",
Springer Verlag, Berlin, 1990.

9.3 Fuzzy Logik

/1/ Constantin von Altrock,
Über den Daumen gepeilt - Fuzzy Logic: Scharfe Theorie der unscharfen Mengen",
c't, Heft 3, 1991.

/2/ Hartwig Hetzheim, Günter Hommel,
"Fuzzy Logic für die Automatisierungstechnik?",
Automatisierungstechnische Praxis atp, Heft 10, 1991.

/3/ George J. Klir, Tina A. Folger,
"Fuzzy sets, uncertainty, and information",
Prentice-Hall International Editions, 1988.

/4/ Chuen Chien Lee,
Fuzzy logic in control systems: Fuzzy logic controller - part I",
IEEE Transactions on Systems, Man and Cybernetics, Vol. 20, No. 2,
März/April, 1990.

/5/ Chuen Chien Lee,
Fuzzy logic in control systems: Fuzzy logic controller - part II",
IEEE Transactions on Systems, Man and Cybernetics, Vol. 20, No. 2,
März/April, 1990.

/6/ E. H. Mamdani, B. S. Sembi,
"Process control using fuzzy logic",
In: P. P. Wang et al. (Eds.),
"Fuzzy Sets: Theory and applications to policy analysis and information systems",
Proceedings Durham, N. C. 28. - 30. 6. 1980, S. 249 - 265, 1980.

/7/ Floor van der Rhee, Hans R. van Nauta Lemke, Jaap G. Dijkman,
"Knowledge based fuzzy control of systems",
IEEE Transactions on Automatic Control, Vol. 35, No. 2, Februar 1990.#

/8/ Lotfi A. Zadeh,
"Fuzzy sets",
Information and Control, 8, S. 338 - 353, 1965.

/9/ Lotfi A. Zadeh,
"Making computers think like people",
IEEE Spectrum, August 1984.

/10/ Lotfi A. Zadeh,
"Outline of a new approach to the analysis of complex systems
and decision process",
IEEE Transactions on Systems, Man and Cybernetics, Vol. 3, No. 1, Januar 1973.

/11/ H.-J. Zimmermann,
"Fuzzy sets theory - and its applications",
2. Auflage, Kluwer-Nijhoff Publishing, 1990.

9.4 Künstliche Intelligenz (allgemein)

/1/ G. Agha, C. Hewitt,
"Concurrent programming using actors-exploiting large-scale parallelism",
In: N. Maheshwari (Ed.), 5th Conference on Foundations of Software Techniology
and theoretical Computer Science, Springer, S. 19, 1985.

/2/ Hans-Werner Güsgen, Manfred Fidelak,
"Programmieren mit Constraints",
Informationstechnik, it 30, 6, 1988.

/3/ B. Hayes-Roth,
"A blackboard architecture for control",
AI Journal 26, S. 251 - 321, 1985.

/4/ Frederick Hayes-Roth
"Rule-based systems",
Communications of the ACM, Vol. 28, No. 9, September 1985.

/5/ B. Faltings,
"Wissensrepräsentation und qualitatives Schließen",
Informationstechnik, it 31, 2, S. 113 - 119, 1989.

/6/ Kenneth D. Forbus,
"Qualitative process theory",
Artificial Intelligence 24, S. 85 - 168, 1984.

/7/ Michael R. Genesereth, Matthew L. Ginsberg,
"Logic programming",
Communications of the ACM, Vol. 28, No. 9, September 1985.

/8/ Ch. Habel,
"Repräsentation von Wissen",
Informatik Spektrum 13, S. 126 - 136, 1990.

/9/ Wolfgang Kreuzer, Bruce McKenzie,
"Programming for artificial intelligence - methods, tools and applications",
Addison-Wesley, Reading, Mass., 1991.

/10/ Roger Penrose,
"Computerdenken",
Spektrum der Wissenschaft Verlagsgesellschaft mbH, Heidelberg, 1991.

/11/ P. Schlüter, A. Behdjati, P. Fleischer, S. Bagdon,
"Objektorientierte Software-Entwicklung: Konzepte und Terminologie",
Siemens AG, Zentralabteilung Forschung und Entwicklung, München.

/12/ Cosima Schmauch,
"Wissensrepräsentation, Grundkurs",
In: Thomas Christaller (Ed.)
5. KIFS 87, Interne Fachberichte 202,
Springer Verlag, Berlin, S. 103 - 156, 1987.

/13/ Patrick Henry Winston,
"Künstliche Intelligenz",
Addison-Wesley, Reading, Mass., 1987.

9.5 Maschinelles Lernen

/1/ Jaime G. Carbonell (Ed.),
"Machine learning - paradigms and methods",
MIT-Press, Cambridge, Mass., 1990.

/2/ Yves Kodratoff,
"Introduction to Machine Learning",
Pitman, 1989.

/3/ Yves Kodratoff, Ryszard S. Michalski (Eds.),
"Machine Learning - an artificial intelligence approach",
Volume III, Morgan Kaufmann Publishers, Inc. Los Altos, Ca., 1990.

/4/ Ryszard S. Michalski, Jaime G. Carbonell, Tom M. Michell (Eds.),
"Machine Learning - an artificial intelligence approach",
Springer Verlag, Berlin, 1983.

/5/ Ryszard S. Michalski, Jaime G. Carbonell, Tom M. Michell (Eds.),
"Machine Learning - an artificial intelligence approach",
Volume II, Morgan Kaufmann Publishers, Inc. Los Altos, Ca., 1986.

/6/ Valerie L. Shalin, Edward J. Wisniewski, Keith R. Levi,
"A formal analysis of machine learning systems for knowledge acquisition",
International Journal Man and Machine Studues, no. 29, S. 429 - 446, 1988.

/7/ Sholom M. Weiss, Casimir A. Kulikowski,
"Computer systems that learn",
Morgan Kaufmann Publishers, Inc. Los Altos, Ca., 1991.

9.6 Neuronale Netze

/1/ William F. Allman,
"Menschliches Denken - Künstliche Intelligenz",
Droemer Knaur, München, 1990.

/2/ Robert Hecht-Nielsen,
"Neurocomputing: Picking the human brain",
IEEE Spectrum, Vol. 25, No. 3, März 1988.

/3/ Christel Kemke,
"Der neue Konnektionismus, ein Überblick",
Informatik Spektrum 11, S. 143 - 162, 1988.

/4/ Monika Köhle,
"Neuronale Netze",
Springer Verlag, Wien, 1990.

/5/ Teuvo Kohonen,
"An introduction to neural computing",
Neural Networks, Vol. 1, S. 3 - 16, 1988.

/6/ Karl Kurbel, Wolfram Pietsch,
"Eine Beurteilung konnektionistischer Modelle auf der Grundlage
ausgewählter Anwendungsprobleme und Vorschläge zur Erweiterung",
Wirtschaftsinformatik, Heft 5 Oktober 1991, S. 355 - 364, Vieweg Verlag, 1991.

/7/ Richard P. Lippmann,
"An introduction to computing with neural nets",
IEEE ASSP Magazine, April 1987.

/8/ Marylin McCord-Nelson, W. T. Illingworth,
"A practical guide to neural nets",
Addison-Wesley Publishing Company, Inc. N. Y., 1990.

/9/ Helge Ritter, Thomas Martinetz, Klaus Schulten,
"Neuronale Netze",
2. Auflage, Addison-Wesley, 1991.

/10/ Kajiro Watanabe, Ichiro Matsuura, Masahio Abe, Makoto Kubota,
"Incipient fault diagnosis of chemical processes via artificial neural networks",
AIChE Journal, Vol. 35, No. 11, S. 1803 - 1812, November 1989.

9.7 Projekt INPRO

/1/ H. B. Keller,
"Unterstützung der Prozeßführung im nuklear-chemischen Bereich durch den Einsatz der Simulationstechnik",
3. Symposium Simulationstechnik, Bad Münster am Stein Ebernburg. Fachberichte Simulation, Berlin: Springer 1985

/2/ A. Jaeschke, H. B. Keller, H. Orth,
"Prozeßinformationssysteme in der Wiederaufarbeitung",
Fachtagung Deutsches Atomforum e. V., Bonn 1987. Berichtsband "Mensch und Chip", Forumverlag, Bonn, 1987.

/3/ Hubert B. Keller,
"Algorithmische und problemstrukturelle Parallelität - Ansätze zur verteilten Simulation komplexer Systeme",
4. Symposium Simulationstechnik, Zürich. Fachberichte Simulation, Springer, Berlin 1987

/4/ Hubert B. Keller,
"Das Modellierungs- und Simulationssystem K_advice",
Vortrag auf dem ASIM-Workshop in Celle.

/5/ Hubert B. Keller,
"Distributed computing - a concept for distributed simulation in Ada",
European Simulation Symposium, Edinburgh, Scotland, San Diego, SCSi, 1989.

/6/ Hubert B. Keller,
"Echtzeitsimulation zur Prozeßführung komplexer Systeme",
Monographie. Fachberichte Simulation, Springer, Berlin, 1988.

/7/ Hubert B. Keller,
"Informationskonzepte für die Simulation",
Einladung zum Vortrag durch die Deutsche Forschungsanstalt für Luft- und Raumfahrt e. V., Oberpfaffenhofen.

/8/ Hubert B. Keller,
"Objektorientierte Benutzerschnittstellen zur Modellierung und Analyse komplexer ökologischer Systeme",
6. Symposium Informatik für den Umweltschutz München, 1991,
Informatik Fachberichte, Springer, Berlin, 1991.

/9/ Hubert B. Keller,
"Parallele Simulationsmethoden zur Ausführung komplexer Modelle",
Jahrestagung der Gesellschaft für Informatik, Vernetzte und
komplexe Informatik-Systeme, Hamburg 1988,
Fachberichte Informatik, Springer, Berlin, 1988.

/10/ Hubert B. Keller,
"Unterstützung der Prozeßführung im nuklear-chemischen Bereich durch den Einsatz der Simulationstechnik",
3. Symposium Simulationstechnik, Bad Münster am Stein Ebernburg. Fachberichte Simulation, Springer Berlin, 1985.

/11/ Hubert B. Keller,
"Verteilte/modulare Echtzeitsimulation komplexer Systeme",
5. Symposium Simulationstechnik Aachen, 1988,
Fachberichte Simulationstechnik, Springer, Berlin, 1988.

/12/ Hubert B. Keller, M. Gauges,
"Das XGAP-System - Ein Werkzeug zum Aufbau interaktiver graphischer Schnittstellen",
KfK-Bericht 4868, Kernforschungszentrum Karlsruhe GmbH, 1991.

/13/ Hubert B. Keller, M. Gauges,
"Generierung objektorientierter Benutzerschnittstellen zur Unterstützung der Simulation komplexer Systeme",
7. Symposium Simulationstechnik Hagen, 1991, Fortschritte Simulationstechnik,
Vieweg, Wiesbaden, 1991.

/14/ Hubert B. Keller, Thomas Weinberger,
"Lernmodelle und Wissensverarbeitung",
KfK-Bericht 5002, Kernforschungszentrum Karlsruhe GmbH 1992.

/15/ Hubert B. Keller, Thomas Weinberger,
"Simulation of children's learning behavior",
Eurosim Simulation Congress, Capri 1992,
Amsterdam: Elsevier, 1992 (erscheint)

10 Begriffe

Im folgenden werden einige wichtige Begriffe kurz erläutert. Weitergehende Begriffe werden zum Zeitpunkt ihrer Einführung im Text erläutert, falls die Gefahr einer Falschinterpretation oder Mehrdeutigkeit besteht.

Signal :=
stellt die Erscheinungsform des Wertes einer Prozeßgröße dar, seine Interpretation liefert den Wert (Wandlung).

Prozeßgröße :=
stellt eine den Prozeß charakterisierende Größe dar, kann direkt oder modellgestützt gemessen sein.

modellgestützte Messung :=
Ableitung nicht direkt meßbarer Meßwerte über ein quantitatives mathematisches Modell (Beobachter).

Symptom :=
beobachtbare Abweichung einer (Prozeß-) Größe von einer Referenz, Hinweis (Charakeristika) eines bestimmten oder mehrerer Fehler (Ursache), kann mit Störung gleichgesetzt werden, oft mehrdeutig.

Symptomatik :=
gesamtes Erscheinungsbild eines Fehlers, Symptomkomplex

Situation :=
relevante Menge der vorliegenden Symptome, die evident für den Zustand des Prozeßbereiches sind.

Kontext :=
Umfeld, bzgl. dem eine Bewertung eines Signals erfolgt, Ziel ist die evidente Symptomatik.

Ursache, Fehler :=
Ausgangspunkt in der kausalen Abhängigkeit, der zu den beobachteten Symptomen führt, als primärer Fehler einem Defekt oder Ausfall entsprechend.

Störung :=
Fehler bzgl. einer Vorgabe, Abweichung von einer Referenz, schwächer als Fehler im Sinne von Ausfall oder Defekt, fehlerhafte Funktionserfüllung, lokal durch Defekt bedingt oder als Folgestörung.

Ausfall, Defekt :=
Funktionsfähigkeit einer Komponente ist fehlerhaft, Defekt verursacht lokale Störungen.

Merkmal :=
Subsummierung aller Größen/Fakten, die den Prozeß charakterisieren, die Gesamtheit aller Merkmale bilden einen Merkmalsraum.

Muster :=
wertemäßige Ausprägung von Merkmalen einer Teilmenge des Merkmalraums.

Interpretation, Bewertung :=
Vergleich eines Merkmals mit seinem Kontext.

Diagnose :=
Rückführung eines Symptoms oder einer Symptomatik auf die zugrundeliegende Ursachen, eine Diagnose kann in Teilschritten erfolgen.

Hypothese :=
Aufstellung einer Annahme bzgl. der die vorliegende Symptomatik erklärende Ursache, wird durch heranziehen weiterer Merkmale (Werte, Muster) verifiziert.

Generierung :=
zielorientiertes Vorgehen; erstellen eines Plans, um ausgehend von der aktuellen Situation zu einer Zielvorgabe zu kommen, z. B. Ableitung eines Eingriffs zur Ausregulierung einer Störung.

Wissen :=
Tatsachen und die Beziehungen zwischen Tatsachen, Unterscheidung in analytisches und heuristisches Wissen, Gegenstand der Erkenntnistheorie (konzeptionelle Ebene).

analytisches Wissen :=
Wissen, das die auf der betrachteten Untersuchungsebene vorkommenden Phänomene durch Rückführung auf unterliegende Tatsachen und Beziehungen erklärt, Vereinfachungen und Abstraktion von unwesentlichen Phänomen ist immer gegeben (Problem der analytischen oder theoretischen Modellierung), axiomatische Systeme, geschlossene formale Systeme/Grundprinzipien z. B. Physik.

heuristisches Wissen :=
Wissen, das die beobachteten Phänomene in Form von aus der Erfahrung gewonnenen Beziehungen erklärt, liefert keine tiefere Einsicht und kann nicht auf unterliegende Strukturen zurückgeführt werden, kann als Blackbox Denken bezeichnet werden, hat statistischen Charakter, Detaillierung kann von wenn .. dann .. -Beziehungen bis zu Ergebnissen von Identifikationsverfahren reichen, Wissen über die Anwendung von Lösungsverfahren.

heuristische Prozedur :=
Verfahren zur Reduktion des Problemraums, Regel zur Behandlung einer Problemklasse (nicht erschöpfend), allgemeine (trial and error) und spezielle (Teilzielbildung), elementare Transformationsregel.

heuristische Struktur :=
Gesamtheit aller heuristischen Prozeduren, Teil der kognitiven Struktur des Menschen, beinhaltet Strategien zur Findung von Lösungsverfahren (rekursive Definition, Meta-), Navigation.

Faktenwissen :=

Wissen um die Eigenschaften von Objekten (abstrakte Einheit, charakterisiert durch ihre Eigenschaften).

Prozedurales Wissen :=

Wissen um die Verarbeitung von Fakten, Transformationsoperatoren für Objekte.

Tiefenwissen :=

Allgemeine Theorien und allgemeine Heuristiken, Weltmodell (Epistemik).

Oberflächenwissen :=

Fachbezogene Heuristiken und Spezialtheorien.

Wissensdarstellung und -Verarbeitung (enge Fassung im Sinne von rechnerorientiert) :=

Formen der Repräsentation von Wissen auf einem Rechner und deren zugehörige Verarbeitungsmodelle, Sprache als Formulierungsmedium und der die Sprache interpretierende Automat, die Erkenntnistheorie als ganzes stellt das allgemeinste Modell der Wissensdarstellung und -verarbeitung (Mensch) dar.

Lernen :=

Mit der globalen Zielvorgabe einer optimalen bzw. optimierten Prozeßführung definieren wir Lernen als mittelbare oder unmittelbare Verbesserung der Performanz des Systems durch eine (adaptive) assimilative oder akkommodative Veränderung der Repräsentation des Wissens.

Bezüglich dem Begriffskomplex Störung, Fehler, Ausfall wird auf /29/, Seite 38, verwiesen.

Referenzen

/1/ G. Agha, C. Hewitt,
"Concurrent programming using actors-exploiting large-scale parallelism",
In: N. Maheshwari (Ed.), 5th Conference on Foundations of Software Techniology and theoretical Computer Science, S. 19, Springer, 1985.

/2/ K. Brammer, G. Siffling,
"Stochastische Grundlagen des Kalman-Filters. Wahrscheinlichkeitsrechnung und Zufallsprozesse",
Oldenbourg Verlag, München, 1975.

/3/ Bruno Buchberger, Bernhard Kutzler,
"Computer-Algebra für Ingenieure",
Aus: Buchberger et al., "Rechnerorientierte Verfahren", B. G. Teubner Verlag, Stuttgart, S. 11 - 69, 1986.

/4/ Dietrich Dörner,
"Problemlösen als Informationsverarbeitung",
Kohlhammer Standards Psychologie, dritte Auflage, 1987.

/5/ Kenneth D. Forbus,
"Qualitative process theory",
Artificial Intelligence 24, S. 85 - 168, 1984.

/6/ B. Freyermuth, R. Isermann, D. Neumann,
"Wissensbasierte On-Line-Fehlerdiagnose in der Fertigung",
Fabrik der Zukunft, Band 5, S. 2 - 5, 1990.

/7/ H. Früchtenicht et al.,
"Technische Expertensysteme: Wissensrepräsentation und Schlußfolgerungsverfahren",
Oldenbourg Verlag, München, 1988.

/8/ Hans-Werner Güsgen, Manfred Fidelak,
"Programmieren mit Constraints",
Informationstechnik, it 30, 6, 1988.

/9/ B. Hayes-Roth,
"A blackboard architecture for control",
AI Journal 26, S. 251 - 321, 1985.

/10/ R. Isermann,
"Wissensbasierte Fehlerdiagnose technischer Prozesse",
Automatisierungstechnik at, 36. Jahrgang, Heft 11, 1988.

/11/ R. Isermann, Bernd Freyermuth, Dieter Neumann,
"Beispiele für die Fehlerdiagnose mittels Parameterschätzung",
at 37 (9/1989), S. 336-343.

/12/ Konrad Joerger,
"Einführung in die Lernpsychologie",
Herderbücherei, Pädagogik, 13. Auflage, 1989.

/13/ Werner Karbach, Marc Linster,
"Wissensakquisition für Expertensysteme"
(Techniken, Modelle, Softwarewerkzeuge),
Carl Hanser Verlag, München, Wien, 1990.

/14/ Hubert B. Keller,
"Echtzeitsimulation zur Prozeßführung komplexer Systeme."
Fachberichte Simulation, Bd. 11, Berlin: Springer, 1988

/15/ Hubert B. Keller, Thomas Weinberger,
"Lernmodelle und Wissensverarbeitung",
KfK-Bericht 5002, Kernforschungszentrum Karlsruhe, März 1992.

/16/ Hubert B. Keller, Thomas Weinberger, Eugen Kugele,
"Wissensbasierte Systeme - Darstellungs- und Verarbeitungsmodelle",
KfK-Bericht, Kernforschungszentrum Karlsruhe, in Vorbereitung.

/17/ Christel Kemke,
"Der neue Konnektionismus, ein Überblick",
Informatik Spektrum 11, S. 143 - 162, 1988.

/18/ M. Kerndlmaier et al.,
"Technische Expertensysteme für Prozeßführung und Diagnose",
Oldenbourg Verlag, München, 1989.

/19/ Monika Köhle,
"Neuronale Netze",
Springer Verlag, Wien, New York, 1990.

/20/ Teuvo Kohonen,
"An Introduction to neural computing",
Neural Networks, Vol. 1, S. 3 - 16, 1988.

/21/ Eugen Kugele, Hubert B. Keller,
Interner Bericht, KfK-IDT, August 1990.

/22/ Eugen Kugele, Hubert B. Keller,
Interner Bericht, KfK-IDT, Dezember 1990.

/23/ Karl Kurbel, Wolfram Pietsch,
"Eine Beurteilung konnektionistischer Modelle auf der Grundlage ausgewählter Anwendungsprobleme und Vorschläge zur Erweiterung",
Wirtschaftsinformatik, Heft 5 Oktober 1991, S. 355 - 364, Vieweg Verlag, 1991.

/24/ Marylin McCord-Nelson, W. T. Illingworth,
"A practical guide to neural nets",
Addison-Wesley Publishing Company, Inc. N. Y., 1990.

/25/ Ryszard Michalski, Jaime G. Carbonell, Tom M. Mitchell,
"Machine Learning - An Artificial Intelligence Approach",
Springer Verlag, 1983.

/26/ Frank Puppe,
"Einführung in Expertensysteme",
Studienreihe Informatik, Springer Verlag, 1988.

/27/ P. Schlüter, A. Behdjati, P. Fleischer, S. Bagdon,
"Objektorientierte Software-Entwicklung: Konzepte und Terminologie",
Siemens AG, Zentralabteilung Forschung und Entwicklung, München.

/28/ Cosima Schmauch,
"Wissensrepräsentation, Grundkurs",
In: Thomas Christaller (Ed.),
5. KIFS '87, Interne Fachberichte 202,
Springer Verlag, S. 103 - 156, 1987.

/29/ W. Schneider-Fresenius (Ed.),
"Technische Fehlerfrühdiagnose-Einrichtungen",
Oldenbourg Verlag, München, 1985.

/30/ Thomas Weinberger,
Interner Bericht, KfK-IDT, September 1991.

/31/ Thomas Weinberger,
"Maschinelles Lernen - "klassische", konnektionistische und kognitive Konzepte",
KfK-Bericht, Kernforschungszentrum Karlsruhe, in Vorbereitung.

/32/ P. H. Winston,
"Künstliche Intelligenz",
Addison-Wesley, New York, 1987.

/33/ Lotfi A. Zadeh,
"Making computers think like people",
IEEE Spectrum, August 1984.