Zeitschrift: Bulletin des Schweizerischen Elektrotechnischen Vereins, des Verbandes Schweizerischer Elektrizitätsunternehmen = Bulletin de l'Association suisse des électriciens, de l'Association des entreprises électriques suisses

Herausgeber: Schweizerischer Elektrotechnischer Verein ; Verband Schweizerischer Elektrizitätsunternehmen

Band: 81 (1990)

Heft: 3

Artikel: Einsatz von wissensbasierten Systemen bei der Überwachung und Diagnose komplexer Anlagen

Autor: Sugaya, Hiro

DOI: https://doi.org/10.5169/seals-903071

Nutzungsbedingungen

Die ETH-Bibliothek ist die Anbieterin der digitalisierten Zeitschriften auf E-Periodica. Sie besitzt keine Urheberrechte an den Zeitschriften und ist nicht verantwortlich für deren Inhalte. Die Rechte liegen in der Regel bei den Herausgebern beziehungsweise den externen Rechteinhabern. Das Veröffentlichen von Bildern in Print- und Online-Publikationen sowie auf Social Media-Kanälen oder Webseiten ist nur mit vorheriger Genehmigung der Rechteinhaber erlaubt. Mehr erfahren

Conditions d'utilisation

L'ETH Library est le fournisseur des revues numérisées. Elle ne détient aucun droit d'auteur sur les revues et n'est pas responsable de leur contenu. En règle générale, les droits sont détenus par les éditeurs ou les détenteurs de droits externes. La reproduction d'images dans des publications imprimées ou en ligne ainsi que sur des canaux de médias sociaux ou des sites web n'est autorisée qu'avec l'accord préalable des détenteurs des droits. En savoir plus

Terms of use

The ETH Library is the provider of the digitised journals. It does not own any copyrights to the journals and is not responsible for their content. The rights usually lie with the publishers or the external rights holders. Publishing images in print and online publications, as well as on social media channels or websites, is only permitted with the prior consent of the rights holders. Find out more

Download PDF: 10.07.2025

ETH-Bibliothek Zürich, E-Periodica, https://www.e-periodica.ch

Einsatz von wissensbasierten Systemen bei der Überwachung und Diagnose komplexer Anlagen

Hiro Sugaya

Ein wichtiges Einsatzgebiet für wissensbasierte Systeme bei Industrieanwendungen ist die Überwachung und Diagnose von komplexen Anlagen. Ein schwieriger Aspekt dieser Anwendung ist die Formalisierung des Diagnosewissens: «Ist etwas nicht in Ordnung, so ist es nützlich zu wissen, wie es eigentlich richtig funktionieren sollte». Entscheidend ist die Kenntnis der Struktur und der Funktion einzelner Anlagekomponenten. Das Vorgehen bei der Realisierung solcher Systeme wird anhand einiger Beispiele erläutert.

Un domaine important pour l'application des systèmes experts dans l'industrie est la surveillance et le diagnostic d'installations complexes. Un aspect difficile est de formuler le savoir diagnostique: «Si quelque chose ne marche pas, il est utile de savoir quel eût été le fonctionnement correct». La connaissance de la structure et de la fonction des différents objets d'installation est fondamentale. La réalisation de tels systèmes est illustrée à l'aide de quelques exemples.

Adresse des Autors
Hiro Sugaya, Dr. sc. techn. ETH Informatiker, ABB Forschungszentrum, Gruppe Wissensbasierte Systeme, 5405 Baden-Dättwil

Die Zustandsüberwachung und die Diagnose von komplexen Anlagen ist eine schwierige und wichtige Aufgabe mit grossen wirtschaftlichen Effekten. Ein sicherer Betrieb bei Kraftwerken, eine optimale Führung bei elektrischen Schaltstationen oder eine effiziente Störungsdiagnose bei rotierenden Maschinen sind Beispiele dieser Anwendung. Um diese Aufgabe lösen zu können, benötigt man interdisziplinäre Kenntnisse aus verschiedenen Bereichen. Die Fortschritte in der Informatikforschung zeigen praktisch anwendbare Lösungen dieser Aufgabe in Form von *technischen Expertensystemen* [1, 2], bei denen das Diagnosewissen formalisiert und auf dem Computer symbolisch verarbeitet werden kann, damit logische Schlüsse gezogen werden können. Bereits realisierte Systeme sind unter anderen die Diagnose für Dampfturbinen und Generatoren (Gen-Aid [3]), die Zustandüberwachung des Kreislaufwassers in thermischen Kraftwerken [4] und die Schwingungsüberwachung bei Rotoren [5].

Der Zustand einer komplexen Anlage kann durch den Vergleich der gemessenen charakteristischen Grössen (elektrische Leistung, thermischer Wirkungsgrad, Schalterstellung) und der Anlagenparameter mit den Sollwerten überprüft werden. Abweichungen vom Normalzustand können durch verschiedene Ursachen begründet sein, wie z.B. Änderungen in der Betriebsführung, unterschiedliche Umgebungsbedingungen, Verschmutzung oder Fehler bei Komponenten. Hier geht man davon aus, dass die Anlagenparameter oder das Modell korrekt sind, und dass deshalb die Störungsursache im realen Prozess liegen muss. Diese Annahme ist zwar theoretisch nicht haltbar, in der Praxis jedoch stark genug, um treffende Aussagen machen zu können. Das Bild 1 stellt das Prinzip der Überwachung bei einer Turbogruppe schematisch dar.

Der nächste Schritt ist die Analyse der Einflüsse von Störungen auf den Prozess, falls Ergebnisse vom Normalzustand abweichen. Hier spielt die Kenntnis der Struktur und der Funktion einzelner Diagnoseobjekte, auf welche die Anlage modelliert werden kann, eine wesentliche Rolle. In bisherigen Systemen werden Messdaten in eine Prozessdatenbank gespeichert, in Verfahrensdaten umgerechnet und dem Operator in Form von Diagrammen, Tabellen oder Prozessbildern mit eingeblendeten Werten präsentiert. Diese Information dient dem Operator dazu, den Anlagenzustand zu beurtei-

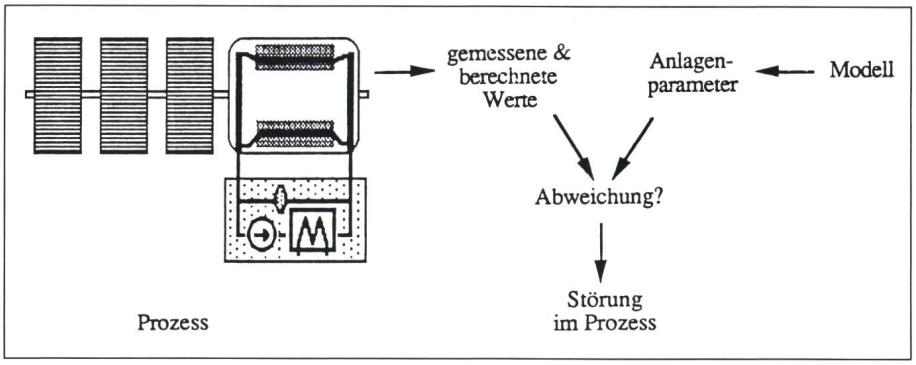

Bild 1 Zustandsüberwachung einer Turbogruppe

len und nötigenfalls Massnahmen einzuleiten, wie z.B. die Reinigung der verschmutzten Komponenten oder die Reparatur eines defekten Teils. Im neuen Ansatz (d.h. Einsatz von technischen Expertensystemen) wird gerade dieses Diagnosewissen auf dem Rechner erfasst und automatisch verarbeitet.

Die wichtigsten Eigenschaften wissensbasierter Systeme sind:

- Der Formalismus, durch den das Bereichswissen repräsentiert wird,
- die Inferenzprozeduren, die das so erfasste Wissen verarbeiten und Lösungen finden,
- die Erklärungskomponente, welche die hergeleitete Schlussfolgerung verständlich macht.

Diese Art vom Rechnereinsatz heisst Symbolverarbeitung. Sie ist ein Bestandteil von wissensbasierten Systemen, ist jedoch nur ein kleiner Teil des gesamten Umfanges. Nebst der Symbolverarbeitung stellt das Bild 2 vier weitere Systemkomponenten anschaulich dar. Charakteristisch ist die Art der Komponten, wie man sie bereits bei bisherigen leittechnischen Systemen findet. Dies deutet darauf hin, dass sich wissensbasierte Systeme in jetziger Rechnerumgebung gut integrieren und erweitern lassen.

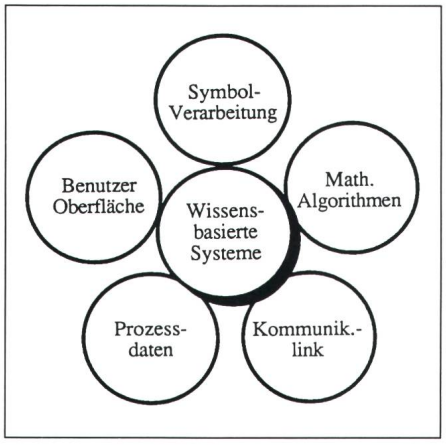

Bild 2 Komponenten eines wissensbasierten Systems

Der Grundsatz der Diagnose ist die Modellierung von Objekten in der realen Welt. Im nächsten Kapitel werden zwei *Diagnoseansätze,* der *fehlerbasierte* und *modellbasierte,* einander gegenübergestellt. Anschliessend werden Vor- und Nachteile detailliert behandelt. Im darauffolgenden Kapitel wird eine Methode beschrieben, bei der *Diagnoseregeln* aus dem Funktions-Struktur-Modell systematisch hergeleitet werden können. Mit dieser Methode ist es nun möglich, die Vorteile beider Ansätze zu kombinieren. Für die Wissensrepräsentation ist es wichtig, einen geeigneten Formalismus zu wählen. Zur Unterstützung des Formalismus und der Weiterverarbeitung des erfassten Diagnosewissens wurde in VIP ein Werkzeug bereitgestellt (im anschliessenden Kapitel beschrieben). VIP ist eine auf Prolog basierende Programmierumgebung zur effizienten Realisierung von wissensbasierten Systemen [6]. In einem weiteren Kapitel wird das Vorgehen bei der *Planung und Einführung* wissensbasierter Systeme aus der Erfahrung erläutert.

Diagnose: fehlerbasiert oder modellbasiert

Expertensysteme für die Diagnose lassen sich in zwei Klassen einteilen. Die einen beruhen auf einer umfangreichen Sammlung von Regeln, welche Erfahrungswissen über die Verknüpfung von Symptomen mit Diagnoseaussagen repräsentieren. Fehlerbäume, Fehlerbuch oder «Dictionary» und Wenn-dann-Regeln sind solche Beispiele. Die andern stützen sich auf Modelle des Diagnoseobjekts, welche über Information betreffend Struktur und Funktion verfügen.

In Systemen der ersten Klasse werden Symptome analysiert, um mögliche Störungsursachen zu erschliessen. Ein diesem *fehlerbasierten* Ansatz zugrundeliegendes Problem ist eine unendlich grosse Anzahl von Symptomen, die sich kaum im voraus vollständig aufzählen lassen. Diese Problematik kann anhand möglicher Fehlerarten eines Geräts (Bild 3) wie folgt präzisiert werden. F_0 ist ein ausgezeichneter Zustand des Geräts und fehlerfrei. $F_1, F_2, ..., F_n, F_{n+1}, ...$ sind alle Fehlerarten, wobei $F_1, F_2, ..., F_n$ bekannte Fehler sind und $F_{n+1}, ...$ unbekannte. Im allgemeinen richten sich Engineeringdisziplinen nach korrekten Funktionsweisen technischer Anlagen. Daher ist, wenn etwas nicht in Ordnung ist, in erster Linie wichtig zu wissen, wie die Anlage *richtig* funktionieren soll. Genau aus dieser Überlegung hat der neue modellbasierte Ansatz in letzter Zeit eine grosse Bedeutung gewonnen.

Systeme der zweiten Klasse sind *modellbasierte* Diagnosesysteme, die sich primär mit der korrekten Funktionsweise von Komponenten befassen [1, 7]. Die Voraussetzung dafür ist eine

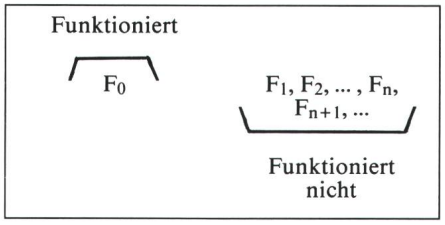

Bild 3 Bekannte und unbekannte Fehler

Modellbeschreibung über die Struktur und Funktion des Systems. Mit diesem Ansatz definiert man ein Symptom bei *Differenzen zwischen beobachtetem und erwartetem Verhalten* (siehe Bild 1). Das Funktionsmodell ermöglicht die Berechnung (Simulation) des erwarteten Verhaltens beim zu untersuchenden System. Liegt ein Fehler im System vor, so ermöglicht die strukturelle Beschreibung zusammen mit der Funktionsbeschreibung die Eingrenzung möglicher Störungsursachen.

Die modellbasierte Diagnose hat verschiedene Vorteile gegenüber den traditionellen Ansätzen mit Fehlerbäumen oder Testprogrammen. Sie ermöglicht den Aufbau eines bereichsunabhängigen Diagnosesystems, da sie nur die Modellbeschreibung voraussetzt und kein explizites Fehlermodell benötigt. Gerade aus diesem Grund ist es möglich, auch unvorhergesehene Fehler zu erkennen. Auch wenn eine Störung weitere Störungen auslöst, kann dieser Ansatz nach einer korrekten Testanordnung primäre und sekundäre Störungen unterscheiden. Zudem kann dieser Ansatz auch mehrere Fehler behandeln [8]. Der modellbasierte Ansatz hat aber auch Nachteile. Es lohnt sich nicht, diesen Ansatz anzuwenden, wenn die möglichen Fehlerursachen leicht aufgezählt werden können oder wenn das Experiment (oder die Erfahrungen) in der realen Welt viel einfacher ist als das Modellieren und Simulieren.

Modellierung kausaler Zusammenhänge

Zur Fehlerdiagnose elektromechanischer Systeme verwendet ein Bereichspezialist seine umfangreichen Kenntnisse über Anlagenstruktur und Komponenten, physikalische Gesetze sowie Erfahrungen. Funktionsschemata mit entsprechender Beschreibung dienen als Grundlage zur Modellierung des zu diagnostizierenden Systems. Das Bild 4 stellt einen Kühlwasserkreislauf bei einem Generator ver-

einfacht dar. Während eines normalen Betriebs kann der Zustand des Generators in verschiedener Weise überwacht werden: durch die Temperaturentwicklung, die Durchflussmenge, den Druck usw. Steigt zum Beispiel die Temperatur des Generators – was kann die Ursache dieser Störung sein? Die Überlastung des Generators? Eine Wasserflussstörung durch die fehlerhafte Pumpe (elektrisch oder mechanisch) oder eine eventuelle Verstopfung bei einer der hydraulischen Komponenten? Wie kann ein Fachspezialist bei der Fehlersuche systematisch vorgehen? Die Komponentenstruktur, Funktion und Heuristiken (empirische Verknüpfungen) sind die drei wichtigsten Elemente des kausalen Modells, die nachfolgend detailliert beschrieben werden.

Komponentenstruktur

Wenn wir alle Komponenten des zu untersuchenden Systems durch einheitliche Blöcke darstellen und diese über Informationspfade miteinander verbinden, so erhalten wir ein Modell des Diagnoseobjekts, das nur strukturelle Informationen enthält. Das Ein- oder Ausgabeverhalten jedes Blockes kann dann durch eine Funktion definiert werden. Die Korrektheit einer Komponente kann nun durch logische Formeln ausgedrückt werden. Wenn eine Komponente X die Eingaben I_1, ..., I_n und Ausgaben O_1, ..., O_m hat, so erhalten wir die logische Aussage

$<O_1$ is correct$>$ and
... and $<O_m$ is correct$>$
If $<X$ is correct$>$ and
$<I_1$ is correct$>$ and
... and $<I_n$ is correct$>$

Die Eingabe I_i (i = 1, 2, ..., n) entspricht wiederum der Ausgabe einer anderen Komponente, und die Komponente X hängt ebenfalls von den darin enthaltenen Subkomponenten ab. Wenn eine Komponente defekt ist, steht deren Beschreibung zu ihrem wirklichen Verhalten im Widerspruch. Zur Fehlerdiagnose werden mit Hilfe des Symptoms alle verdächtigen Komponenten bestimmt. Die entsprechenden Ursachen können durch die Negation der obigen Formel hergeleitet werden. Dies führt zu den Inferenzregeln j = 1, 2, ..., m der Form

$<X$ is faulty$>$ or $<I_1$ is faulty$>$ or
... or $<I_n$ is faulty$>$
If $<O_j$ is faulty$>$

Für das obige Beispiel erhalten wir die Aussagen

<generator_outlet is faulty>
If <cooling_circuit is faulty>

<generator_inlet is faulty> or
<generator is faulty>
If <generator_outlet is faulty>

Diese durch die Struktur bedingten kausalen Abhängigkeiten können in einem Diagnosegraphen abgebildet werden (Bild 5). Jeder Knoten im Graphen entspricht einem möglichen Fehler, und zwar der Endknoten dem atomaren Fehler und der Zwischenknoten der abgeleiteten Wirkung (Symptom). Wenn der Graph von abgeleiteten nach den atomaren Fehlern interpretiert wird, so werden die Ursachen deduziert.

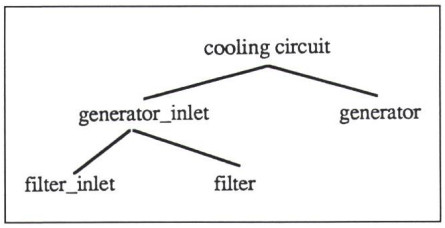

Bild 5 Diagnosegraph

Diese Deduktionsregeln unterscheiden zwei Arten von möglichen Ursachen für Symptome: die Inkorrektheiten bei Eingaben und diejenigen bei Komponenten selbst. Um z.B. die Korrektheit des Generators überprüfen zu können, muss dessen Einlasswasser auf eine mögliche Störung untersucht werden, weil der Generator allein nicht getestet werden kann. Dies führt zur nötigen Anordnung von Diagnoseschritten. Es ist klar, dass alle relevanten Komponenten auf diese Art untersucht werden müssen. Nun kann die Menge der verdächtigen Komponenten durch deren *funktionelle* Verknüpfungen weiter reduziert werden [9].

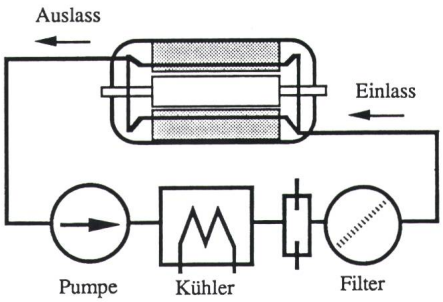

Bild 4 Vereinfachter Kühlwasserkreislauf bei einem Generator

Funktion

Funktionsbeschreibungen können für zwei unterschiedliche Zwecke benützt werden: erstens zur Feststellung eines Symptoms und zweitens zur Eingrenzung möglicher Ursachen, die nur funktionell relevant sind (Die Menge der so erhaltenen verdächtigen Komponenten ist in der Regel kleiner als diejenige, die durch strukturelle Abhängigkeiten hergeleitet wird). Die zweite Art des Schliessens setzt Kenntnisse über die Komponentenmodelle voraus, um kausale Wirkungen zu simulieren. In stationärem Zustand zum Beispiel lässt die Pumpe das Kühlwasser in einer bestimmten Menge mit entsprechendem Beförderungsdruck zirkulieren. Diese kausale Simulation lässt sich wie folgt formalisieren:

Flow rate of generator's outlet water is normal by pump.

Die Inferenzregel für die Diagnose lautet dann

If Output is X by Y and Observation ist not X then Y may be faulty.

Die Grundidee des funktionellen Schliessens ist die, dass jede Komponente eine vordefinierte Zielfunktion hat, deren Wirkung bei der Ausgabe beobachtet werden kann. Das heisst, jede Komponente ist gezielt für eine bestimmte Aufgabe verantwortlich [10].

Das funktionelle Schliessen ist im allgemeinen unvollständig für die Eingrenzung möglicher Ursachen. Die Präzisierung der durch die Negation erhaltenen Aussage ist nicht trivial, da die Ausgabe, wenn sie fehlerhaft ist, unendlich viele verschiedene Werte haben kann. Das funktionelle Schliessen kann eine schnelle Suche nach den defekten Komponenten ermöglichen, ansonsten aber müssen strukturell abhängige Komponenten untersucht werden.

Heuristiken

Die empirische Verknüpfung eines Symptoms mit Ursachen ist eine oft einfache, aber effiziente Lösung für die Fehlerdiagnose von elektromechanischen Systemen. Eine Pumpe z.B., die sporadisch eine ungleiche Durchflussmenge befördert, ist schwer zu modellieren und das Verhalten durch eine kausale Simulation schwer zu analysieren. Es ist viel einfacher, das Verhalten der fehlerhaften Pumpe durch ein reales Experiment zu simulieren und sie mit dem beobachteten

```
cooling_circuit:
        name = '<Generator's cooling circuit disturbance>'
      & type = derived
      & causes = electrical_power_supply or
            water_flow or water_temperature or conductivity.

water_flow:
        name = '<Cooling water_flow disturbance>'
      & type = derived
      & preconditions = not (electrical_power_supply)
      & test = (signal (main_water_flow, Q)  Q≠ normal)
      & causes = water_shortage or pump_problem or hydraulic_devices.

water_shortage:
        name = '<Cooling water shortage>'
      & type = atomic
      & test = signal (tank_level, very_low)
      & action = 'Fill in the cooling water through the makeup valve'.
```

Bild 6 Fehlerobjekte im Frame-Formalismus

Symptom zu verknüpfen. Ausserdem wiederholen sich die beobachteten Fehler im Bereich elektromechanischer Systeme oft von der Art her. Diese Beobachtung führt zur Inferenzregel der Form

Symptom is *X* by *Y*.

Hier ist zu beachten, dass die kausale Simulation eigentlich nur auf korrektes Verhalten der Komponente anwendbar ist, da das Modell keine Fehlerbeschreibung enthält.

Zusammenfassend beschreibt das kausale Modell die Abhängigkeiten zwischen Symptomen und deren möglichen Ursachen durch die strukturellen Verbindungen und die Funktionsweise von Komponenten. Das so erstellte Diagnoseobjekt beinhaltet nicht nur die Information über die Ursachen, sondern auch die Testfunktion zur Festlegung des Symptoms, Erklärungskomponente sowie die Reparaturanweisung. Empirische Anordnungen sind kurze Wege, die Symptome unmittelbar mit Ursachen zu verknüpfen, und können ebenfalls durch das kausale Modell beschrieben werden. Diese Methode ist generell auf beliebigen elektromechanischen Geräten anwendbar (z.B. auf die Fehlerdiagnose von analogen Schaltkreisen [11]).

Formalismus und Werkzeuge

Wir haben oben die drei wichtigsten Elemente des kausalen Modells für die Diagnose beschrieben. Der Repräsentationsformalismus soll nun sowohl die Formulierung des modellbasierten und empirischen Diagnosewissens, als auch deren Interpretation unterstützen. Im folgenden werden zuerst der Formalismus und dann die entsprechenden Werkzeugfunktionen beschrieben.

Formalismus

Die oben beschriebenen Inferenzregeln eignen sich zur Repräsentation kausaler Zusammenhänge zwischen Komponenten. Sie eignen sich jedoch weniger zur Beschreibung statischer Eigenschaften von Komponenten wie z.B. verschiedener Bedingungen zur Fehleranalyse, Funktionsbeschreibungen, Erklärungen und Reparaturanweisungen. Der einfache Frame-Formalismus (sog. Property-Lists) genügt, um solche Eigenschaften einheitlich darzustellen (Bild 6).

Ein Fehler wird durch ein Frameobjekt repräsentiert, das einem Knoten des Diagnosegraphen entspricht. Die Eigenschaften eines Fehlers sowie die kausalen Zusammenhänge werden durch Slots (Properties) beschrieben. Zum Beispiel der *name*-Slot gibt eine detaillierte Beschreibung eines Fehlerobjekts. Der *type*-Slot gibt an, ob es ein atomarer oder abgeleiteter Fehler ist. Wenn es ein atomarer Fehler ist, so wird eine Reparaturanweisung in einem *action*-Slot angegeben. Ein *test*-Slot beschreibt die Testfunktion und der *causes*-Slot die möglichen Ursachen.

Als Beispiel wird der Fehler des Kühlsystems mittels des Generatorauslasswassers beschrieben, und zwar anhand der elektrischen Speisung, der Pumpe, der Durchflussmenge, der Temperatur oder der Leitfähigkeit. Die elektrische Störung wurde aufgrund heuristischer Überlegungen in Betracht gezogen, während die drei anderen Ursachen aus der strukturellen Abhängigkeit hergeleitet wurden. Die Störung *water_flow* wird dann untersucht, wenn die Vorbedingungen erfüllt sind, d.h. die elektrische Speisung nicht defekt ist. Dann erwarten wir die Meldung *water_flow* ist normal, sonst gestört. Falls gestört, so sind der Wassermangel, die Pumpe oder andere hydraulische Komponenten die möglichen Ursachen. Die Ursache *water_shortage* ist eine heuristische Überlegung, die anderen beschreiben strukturelle Abhängigkeiten. Das funktionelle Schliessen schreibt vor, bei gestörtem Wasserdurchfluss die Pumpe zuallererst zu untersuchen, da diese funktionell dafür verantwortlich ist.

Werkzeuge

Im wesentlichen wurden zwei Werkzeuge zur Erfassung des Diagnosewissens und dessen Verarbeitung bereitgestellt. Mittels eines Grapheneditors und eines Formeditors werden Fehlerbeschreibungen in die Frame-Struktur eingegeben. Durch ein Parser wird die Beschreibung nachher in die interne Datenstruktur aufgenommen.

Der Diagnoseprozess verarbeitet nun den Graphen von der Wurzel (allgemeinstes Diagnoseobjekt) nach den Blättern (atomares Diagnoseobjekt). Für jeden Knoten des Graphen wird die Störung zu einem der drei möglichen Zustände deduziert: «existiert», «kann nicht existieren» oder «kann nicht beseitigt werden». Genauer geht der Diagnoseprozess in folgenden drei Schritten vor sich:

1. Auswertung der Vorbedingungen vor der Feststellung eines Symptoms. Sie werden benutzt, um die Menge gestörter Komponenten zwischen primären und sekundären Ursachen aufzuteilen. Bei der Wasserflussstörung zum Beispiel wird zuerst die elektrische Speisung untersucht, da sie eine Menge sekundärer Störungen verursachen kann.
2. Feststellung der Ergebnisabweichung (bzw. des Symptoms) bei einem Diagnoseobjekt durch eine entsprechende Testfunktion.
3. Suche nach den Ursachen entweder durch zusätzliche Messungen und Beobachtungen oder durch einfaches Absuchen aller Varianten (exhaustive search).

Die Vorbedingungen und die Testfunktion sind optionell. Wenn sie feh-

len, dann werden die Ursachen bedingungslos weiter untersucht. Eine weitere wichtige Werkzeugfunktion ist die Konsistenzerhaltung bei der Modellbeschreibung, d.h. ein Symptom wurde erkannt, aber keine mögliche Ursache kann festgestellt werden. In diesem Fall ist die Modellbeschreibung unvollständig, und die möglichen Ursachen müssen erst noch festgestellt werden.

Entwicklungsmethode

Bei der Planung und Einführung wissensbasierter Systeme für energietechnische Anwendungen werden Projektmanager öfters mit folgenden Fragen konfrontiert:

- Kann Knowledge-Engineering von durchschnittlichen Informatikingenieuren gemacht werden?
- Brauchen wir den «high-priced» Knowledge-Ingenieur?
- Brauchen wir neue Software?
- Brauchen wir eine neue System-Entwurfsmethode?

Die kürzlich durchgeführte Untersuchung von Weitzel [12] gibt zu den Fragen die folgenden Antworten: Ja, Nein, Ja und Ja. Diese Haltung stimmt ebenfalls mit unserer Erfahrung überein. Wie im Bild 2 dargestellt, ist Symbolverarbeitung ein kleiner Teil des ganzen Systems. Die konventionellen Methoden und Techniken des Software-Engineerings können weiterhin übernommen werden. Der herkömmliche Systemanalytiker kann solche Aufgaben zusammen mit einem Bereichspezialisten wahrnehmen. Shells und Umgebung für Wissensverarbeitung ermöglichen dem Analytiker durch Fast-prototyping, sich auf das Problem statt auf das Programmieren zu konzentrieren. Die dazu nötige Entwicklungsmethode wird nachfolgend kurz beschrieben.

Vorgehen

Einer zu bildenden Arbeitsgruppe sollte mindestens ein Informatiker (mit KI-Erfahrung von Vorteil) und ein Bereichspezialist angehören. Das wissensbasierte System kann dann in folgenden drei Phasen realisiert werden: 1. Vorstudie, 2. Entwicklung eines Demonstrationssystems und 3. Entwicklung des Zielsystems. In der Vorstudie werden die Machbarkeit (Ist die entsprechende Technologie anwendungsreif?), Durchführbarkeit (Einsatz von Rechner, Messgeräten und Überwachungseinrichtungen) und Wirtschaftlichkeit (Rentiert der Aufwand?) überprüft. Wenn diese drei Kriterien positiv beurteilt werden, kann die Arbeitsgruppe mit der Entwicklung des Demonstrationssystems beginnen.

Nebst den KI-Sprachen Lisp und Prolog gibt es auf dem Markt relativ viele Shells und Umgebungen. Wichtig ist, dass die Prinzipien des Zielsystems anhand einiger Beispiele demonstriert werden. Die vom Bereichspezialisten gewählten Beispiele sollten deshalb charakteristische Anwendungseigenschaften sowohl in der Breite (oberflächlich) wie auch in der Tiefe (Komplexität) beinhalten. Weiter soll ein geeigneter Formalismus zur Wissensrepräsentation – wie z.B. weiter oben beschrieben – festgelegt werden. Bei fortgeschrittenen Systemen wird der Frame-Formalismus mit methodischer Unterstützung zur problemspezifischen Beschreibung verwendet. Mit Vorsicht zu benützen ist der Wenn-dann-Formalismus, wie z.B.

```
If    diff_pressure of winding is
      greater (110) and
      flow_rate is in_range (90,110)
Then  flow_restriction =
      over_max_pressure.
```

Obwohl der Umgang mit diesem Formalismus relativ einfach ist, wird trotzdem nur schwer erkannt, wie diese Regeln zusammenwirken. Nebst fehlender methodischer Unterstützung führt dies weiter zum Problem, dass es mit zunehmender Anzahl von Regeln äusserst schwierig sein wird, deren Konsistenz aufrechtzuerhalten. Der Frame-Formalismus ermöglicht eine uniforme und anschauliche Repräsentation des Diagnosewissens.

Nach der Realisierung des Demonstrationssystems gibt es folgende vier Möglichkeiten:

- Das System ist bereits einsetzbar (selten).
- Das System steht vor neuen Problemen (zu schwierig, nicht wirtschaftlich).
- Das System hat alle Ziele erreicht und das Zielsystem soll nun weiter entwickelt werden
- Das System hat alle Ziele erreicht, soll jedoch wegen der Effizienz in einer prozeduralen Sprache neu implementiert werden.

Der Aufwand für die Weiterentwicklung ist in der Regel um einige Faktoren grösser als die Entwicklung eines Demonstrationssystems.

Erfahrungen

Das Vorgehen bei der Prototypentwicklung wird nachfolgend an konkreten Beispielen aus Anwendungen in thermischen Kraftwerken [13, 14], Schaltstationen von Energieversorgungsnetzen und elektrischen Lokomotiven kurz erläutert. Der erste Schritt bei der Realisierung eines Prototyps ist die Auswahl charakteristischer Beispiele aus dem Anwendungsbereich und ihre kurze Beschreibung durch Tabellen, Texte, Entscheidungsbäume oder deren beliebige Kombinationen. In den meisten Fällen sind Bereichspezialisten mit der Formalisierung ihrer Kenntnisse nicht vertraut. Es ist die Aufgabe von Informatikern, entsprechende Methoden und Techniken anzuwenden und einen für die Problemlösung geeigneten Beschreibungsformalismus festzulegen. Dazu müssen sie die vom Bereichspezialisten ausgearbeiteten Beispiele genau verstehen. Umgekehrt soll der Formalismus vom Bereichspezialisten verstanden und auch akzeptiert werden. Das Bild 6 zeigt ein Beispiel eines solchen Formalismus zur Beschreibung der Diagnosekenntnisse bei einem Kraftwerk.

Das zweite Beispiel ist die Modellierung von Hochspannungsschaltern für die Überprüfung von Verriegelungsbedingungen (Bild 7). Das Feld *field1* besteht aus vier Schaltern *q0, q1, q2* und

```
field1:                                  field_ type_ a:
    name            ='Field 1'           name                 ='Type A'
    & ako           = field_type_a       & ako                = field_type
    & left          = none               & switch_type(q0)    = circuit_breaker
    & right         = field2             & switch_type(q1)    = busbar_isolator
    & switch(q0)    = open               & switch_type(q2)    = busbar_isolator
    &switch (q1)    = open               & switch_type(q51)   = earthing_switch.
    & switch(q2)    = open
    & switch(q51)   = open.
```

Bild 7 Schalterbeschreibung

g51 vom Typ Leistungsschalter, Isolator, Isolator und Erdungsschalter. Die anlagenspezifische Zustandsinformation, die Schalterstellung oder die Beziehung zu den benachbarten Feldern sind im Objekt *field1* dargestellt. Hingegen wird die Information über die Art des Feldes durch die *ako*-Beziehung vom Objekt *field_type_a* vererbt. Aufgrund dieser Beschreibung überprüft eine in Prolog geschriebene Prozedur *interlocking_control (SWITCH, FIELD, ACTION, MODE)* die Verriegelungsbedingungen je nach Schalterstellung, Schalterarten und Aktion (Ein- oder Ausschalten).

Das dritte Beispiel ist die Störungsanalyse bei elektrischen Lokomotiven. Bei dieser Anwendung erstreckt sich die Beschreibung kausaler Zusammenhänge von der Funktions-Strukturanalyse bis zum zeitlichen Verhalten der Lokomotive (Bild 8): Der Slot *predecessors* beschreibt z.B. mögliche Ursachen der Störung *dg1_dauer* und der Slot *effect* deren Folgewirkung. Aufgrund dieser Hinweise kann der Lokomotivführer die Entspannungstaste betätigen, und das Drehgestell wird durch das Antriebsleitsystem vom Netz abgetrennt.

Die Formalisierung des Anwendungswissens mit entsprechendem Formalismus ist überhaupt die wichtigste Aufgabe, deren Qualität für die erfolgreiche Durchführung des Projekts eine entscheidende Rolle spielt.

```
dg1_dauer:
        name = '<Dauerstörung im Drehgestell DG-1>'
      & type = atomic
      & predecessors = speisung_alg1 or fahrleit_sp_flg1
      & advice = 'Entspannungstaste betätigen'
      & effect = 'DG-1 wird abgetrennt'.
```

Bild 8 Störungsanalyse bei Lokomotiven

Zusammenfassung

Mit zunehmender Komplexität von Energiesystemen ist der Einsatz von wissensbasierten Systemen zur Überwachung und Diagnose sicherlich von grossem Interesse. Moderne Anlagen mit leittechnischen Einrichtungen bieten eine ideale Voraussetzung für die Einführung solcher Systeme. Die Zustandsüberwachung hat, nebst der frühzeitigen Störungserkennung, den wirtschaftlichen Vorteil, dass die Lebensdauer der Anlagen voll ausgenützt werden kann. Die Berücksichtigung von Sicherheitsaspekten ist jedoch unabdingbar.

Der Autor dankt Frau S. Hitzig-Sander, den Herren Dr. J. Kriz, Dr. G. Lindberg, C. Muller, D. Vögtli, R. Siegenthaler und B. Wiederkehr für die vielen Anregungen und die zum Teil enge Zusammenarbeit.

Literatur

[1] *D. G. Bobrow:* Qualitative reasoning about physical systems. Artificial Intelligence 24(1984)1.

[2] *P. Wallich:* Software «doctor» prescribes remedies. IEEE Spectrum 23(1986)10, p. 43...48.

[3] *A. J. Gonzales* et *R. L. Osborne:* Un «système expert» pour le diagnostic en temps réel des turbo-alternateurs en service. Rapport CIGRE No. 11-07, 1986.

[4] *M. Gallanti* a. o.: Representing procedural knowledge in expert systems: An application to process control. Proceedings of the Ninth International Joint Conference on Artificial Intelligence, 18...23 August 1985, Los Angeles/California; vol. 1, p. 345...352.

[5] An expert system for on-line machinery diagnostics. EPRI Report NP-3652. Palo Alto/California, Electric Power Research Institute, 1984.

[6] *C. Muller* and *S. Sander:* Versatile interactive Prolog for graphics displays: User's manual. Baden, ABB-Research Center, 1988.

[7] *R. Davis* and *W. C. Hamscher:* Model-based reasoning: Troubleshooting. A.I. Memo No. 1059. Cambridge/Massachusetts, Massachusetts Institute of Technology (MIT), 1988.

[8] *J. de Kleer* and *B. C. Williams:* Diagnosing multiple faults. Artificial Intelligence 32(1987)1, p. 97...130.

[9] *M. R. Genesereth:* The use of design descriptions in automated diagnosis. Artificial Intelligence 24(1984)1-3, p. 411...436.

[11] *J. Kriz* and *H. Sugaya:* Knowledge-based testing and diagnosis of analog circuit boards. 16th Annual International Symposium on Fault Tolerant Computing Systems, Vienna/Austria, 1...4 July 1986; p. 378...383.

[12] *J. R. Weitzel* and *L. Kerschberg:* Developing knowledge-based systems: Reorganizing the system development life cycle. Communications of the ACM 32(1989)4, p. 482...488.

[13] *G. Lindberg:* On-line condition monitoring of power station components using expert systems. EPRI Conference on Expert Systems Applications for the Electric Power Industry, June 1989, Orlando.

[14] *H. Sugaya:* Überwachung und Diagnose bei einer Turbogruppe. ETG-Informationstagung «Expertensysteme in der elektrischen Energieversorgung», 27. Juni 1989, Zürich. Zürich, SEV, 1989; S. 79...90.