Zeitschrift: Technische Mitteilungen / Schweizerische Post-, Telefon- und Telegrafenbetriebe = Bulletin technique / Entreprise des postes, téléphones et télégraphes suisses = Bollettino tecnico / Azienda delle poste, dei telefoni e dei telegrafi svizzeri

Herausgeber: Schweizerische Post-, Telefon- und Telegrafenbetriebe

Band: 70 (1992)

Heft: 2

Artikel: Wissensbasierte Systeme im Netzwerkmanagement : Teil 2: Erfahrungsbericht

Autor: Liver, Beat / Prim, André

DOI: https://doi.org/10.5169/seals-873971

Nutzungsbedingungen

Die ETH-Bibliothek ist die Anbieterin der digitalisierten Zeitschriften. 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. Siehe Rechtliche Hinweise.

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. Voir Informations légales.

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. See Legal notice.

Download PDF: 29.03.2025

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

Wissensbasierte Systeme im Netzwerkmanagement

Teil 2: Erfahrungsbericht

Beat LIVER und André PRIM, Bern

1 Das Forschungsprojekt

11 Aufgabenstellung

In diesem zweiten Teil (der erste ist in den «Technischen Mitteilungen PTT» Nr. 1/1991 erschienen) werden das Forschungsprojekt «Wissensbasierte Systeme im Netzwerkmanagement» und dessen Ergebnis vorgestellt. [1].

Das Projekt bezweckt in erster Linie den Aufbau von Kenntnissen im Bereich wissensbasierter Systeme, damit diese beurteilt und entwickelt werden können. Dazu wurde eine Prototypanwendung aus dem Bereich des Netzwerkmanagements, *Mail-Diagnostic,* mit Unterstützung der Firma *Synlogic AG* in Binningen erstellt. Das Projektziel ist die Entwicklung eines Fehlerdiagnose-Systems für Decnet-Netze und VMS-Rechner von *DEC,* dessen genauer Umfang noch zu definieren ist. In der Projektgruppe ist ein Experte für die Fehlerdiagnose in VMS-Rechnern und Decnet vorhanden. Damit ist neben der Angemessenheit und technischen Machbarkeit auch die umfeldbezogene Machbarkeit für ein wissensbasiertes System in dieser Domäne gegeben. Es wird keine Rechtfertigung der Auswahl des Anwendungsgebietes auf Grund wirtschaftlicher Überlegungen durchgeführt. Die Tatsache, dass die Fehlererkennung und -behebung eine typische Aufgabe im Fernmeldedepartement der PTT-Betriebe ist, rechtfertigt diese Wahl. Die recht schwierige Kosten-Nutzen-Analyse ist aber durchzuführen, wenn ein produktives System entwickelt werden soll.

Für die Präzisierung der Aufgabenstellung ist eine genaue Charakterisierung der Domäne nötig. Dazu wird der ganze Problemraum beschrieben und strukturiert. Die Domäne Computernetze lässt sich vor allem bezüglich der geographischen Ausdehnung, der Art der Protokolle und Anwendungen gliedern. Im Projekt wird die Domäne durch die vorhandene Expertise auf Decnet Phase IV und VMS-Rechner eingeschränkt. Als Applikation wurde die «elektronische Post» gewählt. Die Anwender von Mail-Diagnostic sind Systembetreuer von VMS-Rechnern, die nicht Netzwerkspezialisten sind.

Die weitere *Eingrenzung (Scoping)* führt zum Ergebnis, dass Fehler von VMS-Mail, beruhend auf VMS Version 5.4 und Decnet Phase IV für lokale Netze diagnostiziert werden sollen. Gruppen von VAX-Rechnern (Cluster), Bedienungsprobleme von VMS-Mail und transiente Effekte werden nicht berücksichtigt. Im weiteren sollen Fehler nur soweit genau lokalisiert werden, als vom Systembetreuer korrigierbare Komponenten betroffen sind. Das heisst, dass zum Beispiel Fehler in der Netzwerksoftware nicht erkannt werden.

Die Eingrenzung der Aufgabenstellung ist sehr zeitaufwendig, da alle Schlüsselmitarbeiter des Projekts zu einem gemeinsamen Verständnis der Domäne und des geplanten Systems kommen müssen.

12 Das Ergebnis

Ergebnis des Projekts ist im wesentlichen ein Wissensmodell nach KADS und dessen Implementation, das wissensbasierte System Mail-Diagnostic, mit der Expertensystem-Entwicklungsumgebung Smeci (ein Produkt der Firma *Ilog SA).* KADS ist eine recht neue Methode, die immer häufiger für die Entwicklung wissensbasierter Systeme verwendet wird [2, 3]. Das entwickelte wissensbasierte System ist ein funktionales Modul für die Fehlerüberwachung, das in ein Netzwerkmanagementsystem integriert werden könnte. Die Wissensbasis beruht auf etwa 20 Fehlerfällen.

Ausgehend von einer Fallbeschreibung, die aus der Adresse des Senders und Empfängers und den Fehlermeldungen besteht, wird die Fehlerursache ermittelt. Die Fallbeschreibung erhält der Systembetreuer eines Rechners meistens telefonisch von einem Benützer. Mail-Diagnostic fragt den Systembetreuer nach weiteren Fakten, die durch Systembefehle oder Telefonate mit anderen Systembetreuern ermittelt werden. Die Befehle oder Fragen werden vom System vorgegeben. Das System liefert die diagnostizierte Fehlerursache und eine Anweisung oder zumindest Hinweise, wie der Fehler zu beheben ist.

Das funktionale Modell *(Fig. 1)* gibt einen Überblick über das vorhandene System und mögliche Erweiterungen, die durch punktierte Linien dargestellt sind.

Schwerpunkt des Forschungsprojekts bildet die Modellierung der Expertise in einer Form, die unabhängig von der Implementation ist. Damit soll eine in gewissem Sinne modulare und wartbare Wissensbasis geschaffen werden. Die Erfahrungen deuten darauf hin, dass KADS ein Ansatz zur Lösung dieser Probleme ist.

13 Ausblick

Der Umfang der Wissensbasis könnte einerseits um Wissen über zusätzliche Anwendungen wie X.400 und

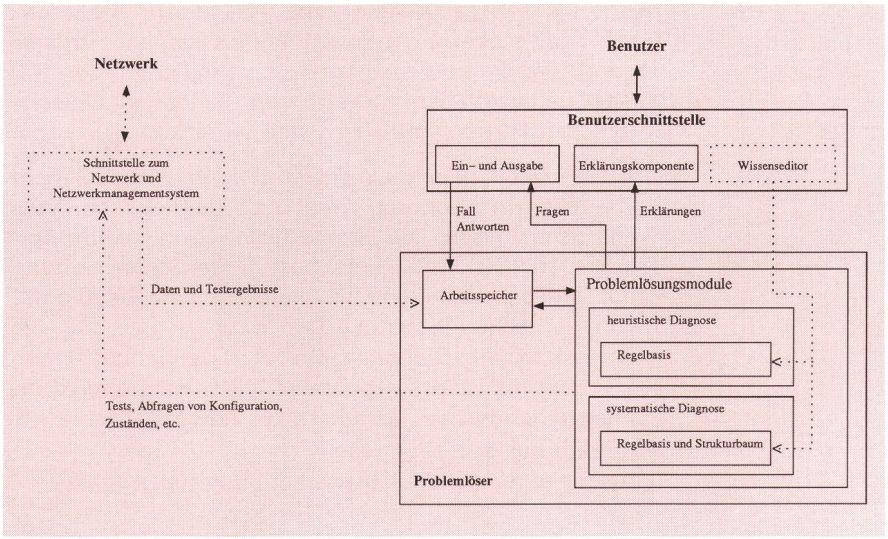

Fig. 1 Funktionales Schema des Prototyps Mail-Diagnostic

Filetransfer, weitere Betriebssysteme oder andere Netzprotokolle wie X.25, ISDN, FDDI usw. erweitert werden. Anderseits könnte auch eine vertiefte Expertise eingebracht werden. Dies hiesse mehr Komponenten, Parameter und dazugehörige Heuristiken und Tests zu modellieren.

Über eine Schnittstelle des wissensbasierten Systems zu einem Netzwerkmanagementsystem könnten viele der nötigen Informationen automatisch beschafft werden. So könnte Mail-Diagnostic in die Implementation von DEC der Enterprise Management Architecture (EMA), DECMCC, integriert werden.

Ein Wissenseditor, mit dem der Experte die Wissensbasis selbständig erweitern oder anpassen kann, wäre wünschenswert. Damit kann der recht grosse Aufwand für Aufbau und Wartung von Wissensbasen verkleinert werden. Das vorhandene konzeptuelle Modell wäre dazu ein guter Ausgangspunkt. Dabei sollte soweit möglich maschinelles Lernen eingesetzt werden.

Der Bereich des Netzwerkmanagements ist sehr dynamisch. Daher ist die kostengünstige Wartung und Erweiterung der Wissensbasis für den praktischen Einsatz wissensbasierter Systeme von entscheidender Bedeutung. In einem Folgeprojekt ist geplant, die Verbesserung der Wartbarkeit und Anpassbarkeit zu studieren. Unter anderem soll die Verwirklichung des erwähnten Wissenseditors untersucht werden.

2 Entwicklung der Modelle nach KADS

Die Modelle wurden nach der Vorgehensweise KADS, die im ersten Teil beschrieben ist, entwickelt. *Figur 2* zeigt die Beziehungen zwischen den Entwicklungstätigkeiten, den Modellen und weiteren Dokumenten. Modelle und Dokumente werden durch Rechtecke dargestellt (z. B. die Modelle *verbale Daten, konzeptuelles Modell, Design* und *Code).* Ovale symbolisieren die Entwicklungstätigkeiten. Von den Unterlagen weist ein Pfeil auf die Tätigkeiten, für die sie benötigt werden. Die von einer Tätigkeit wegführenden Pfeile zeigen auf die Ergebnisse der Tätigkeit. Jedes der folgenden Unterkapitel beschreibt eines der vier Modelle und dessen Entstehung. Dabei wird nur der für wissensbasierte Systeme spezifische Aspekt der Modellierung und Implementierung einer Expertise dargestellt. Auf die Überprüfung, Validation und Evaluation kann im Rahmen dieses Artikels nicht näher eingegangen werden.

21 Die Analyse

Eine Liste von typischen Fällen von Diagnoseproblemen wird erstellt. Der Wissensingenieur (Knowledge Engineer) befragt den Experten, wie er diese Fälle lösen

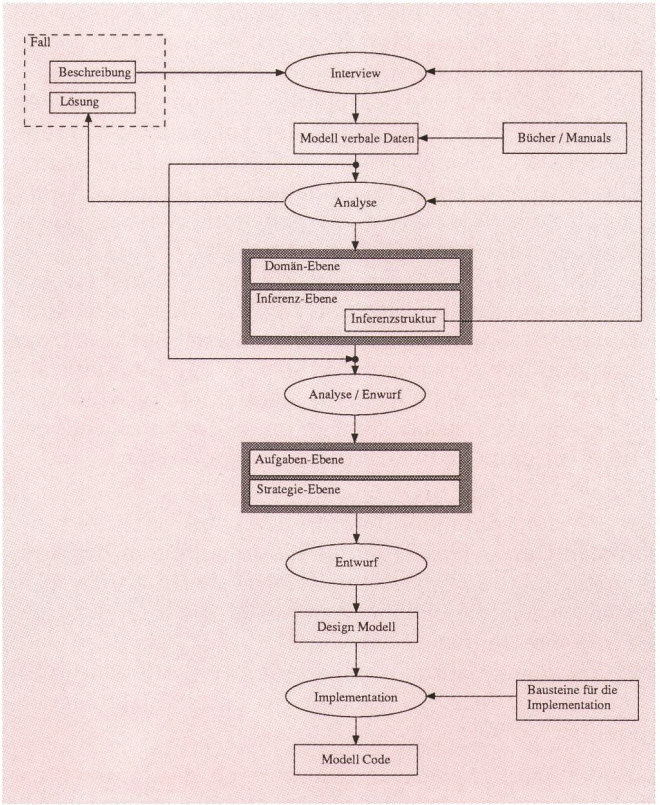

Fig. 2 Entwicklung von Mail-Diagnostic mit KADS

Konzeptuelles Modell

Beschreibung
Sender : Müller_T auf Rechner A
Empfänger : Meier_U auf Rechner B

Fehlermeldung:
%MAIL-E-loglink, error creating network link to node B
-SSYSTEM-F-UNREACHABLE, remote node is not currently reachable

Lösung
1. Mail vom Operator von A zum Operator von B; Resultat: gleiche Fehlermeldung
2. Mail vom Operator von A zum Operator eines Referenzrechners; Resultat: gleiche Fehlermeldung
   Vermutung: Ein DECnet Problem auf A
3. Remote Login von A auf B (SET HOST); Resultat: funktioniert nicht
4. Wieviele Rechner des Netzes "sieht" der Rechenr A?; Resultat: keine
5. Was ist der Status des Executors von DECnet?; Resultat: ON
6. Was ist der Status der Line?; Resultat: SYNCHRONIZING (Problem)
7. Ist mauelles auf 'ON' setzen des Status der Line möglich?; Resultat: Nein
   Fehlerursache: Die physische Verbindung von A mit dem Netz ist unterbrochen

Fig. 3 Ein Fall aus dem Projekt

würde. Die Beschreibung der Problemlösung, Begriffserklärungen, Bemerkungen usw. werden notiert. Zusammen mit Fachwissen, das in Büchern und Handbüchern enthalten ist, bilden diese Befragungsprotokolle das Modell *verbale Daten.* Aus diesem wird nun das *konzeptuelle Modell* erstellt. Es besteht aus den folgenden vier Ebenen:

- Die *Domän-Ebene* umfasst alle nötigen Konzepte der Domäne und deren Relationen.
- In der *Inferenzebene* werden alle *Inferenzen* – die Basisfunktionen für Schlussfolgerungen – beschrieben. *Die Eingabe-* und *Ausgabeargumente* der Inferenzen heissen *Metaklassen.* Diese bezeichnen Mengen von Konzepten der Domäne, die eine bestimmte *Rolle* in einer Schlussfolgerung spielen. Hypothese, Ursache usw. sind Rollen. Dadurch werden also für die Argumente der Inferenzen die möglichen Konzepte und deren Rollen definiert. Die Inferenzen und Metaklassen bilden zusammen die *Inferenzstruktur.* Aus dieser ist ersichtlich, welche Metaklassen eine Inferenz als Ausgabe- bzw. Eingabeargumente verwenden.
- Die *Aufgabenebene* besteht aus Aufgaben, mit denen bestimmte Ziele erreicht werden sollen. Eine Aufgabe besteht aus elementareren Aufgaben und Inferenzen. Die Anwendung der Teilaufgaben und Inferenzen wird durch *Kontrollanweisungen* gesteuert. In dieser Ebene wird also der Kontrollfluss definiert.
- In der *Strategieebene* werden jene Aufgaben beschrieben, deren Zielerreichung von der jeweiligen konkreten Situation abhängt. In dieser Ebene ist der Kontrollfluss dynamisch, da er von der Situation abhängt, in der eine Aufgabe ausgeführt werden soll. Der statische Kontrollfluss der Aufgabenebene genügt für diese Art von Aufgaben nicht mehr.

Der Wissensingenieur interpretiert und analysiert die verbalen Daten. Das Ergebnis dieser Arbeit ist das konzeptuelle Modell. Im ersten Analyseschritt (Analyse in Fig. 2) werden die Lösungen der Fälle, die Inferenzstruktur und die Domän-Ebene ermittelt. Für jeden Fall wird das Lösungsverfahren des Experten beschrieben. In *Figur 3* ist eine Fallbeschreibung aus dem Projekt wiedergegeben.

Die vom Experten verwendeten Inferenzen werden identifiziert, und eine Inferenzstruktur wird konstruiert. Für diese Konstruktion steht dem Wissensingenieur eine Bibliothek von typischen Inferenzstrukturen zur Verfügung, die sehr allgemein sind. Aus dieser Bibliothek wählt er eine oder mehrere Strukturen aus, ändert diese ab und fügt sie neu zusammen. Im Projekt wurden folgende Schlussfolgerungsverfahren, zu denen typische Inferenzstrukturen vorhanden sind, ermittelt:

- *Heuristische oder assoziative Diagnose:* Dabei wird von einer Beobachtung direkt auf eine mögliche Ursache geschlossen. Die direkte Assoziation ist zum Beispiel möglich, wenn eine bestimmte Fehlermeldung mit grosser Wahrscheinlichkeit auf eine bestimmte Fehlerursache hindeutet.
- *Systematische Diagnose:* Sie wird verwendet, wenn keine heuristische Diagnose möglich ist. Dabei kommen folgende zwei Verfahren zur Anwendung:
- strukturbasierte Diagnose
- Verfolgen von Ursachen-Wirkungs-Ketten.

Die *strukturbasierte Diagnose* wird in der vorhandenen Expertise angewandt, wenn die beiden anderen Verfahren nicht möglich sind. Dazu wird das System in Komponenten und Subkomponenten aufgegliedert. So kann zum Beispiel das Mailsystem in die Subkomponenten «VMS-Mail», «Decnet» und «Benützer» unterteilt werden. Dabei entsteht ein *Strukturbaum (Fig. 4).* In diesem bezeichnet «PB-Localization» das Mailsystem. Der Strukturbaum stellt den vollständigen modellierten Suchraum des Diagnosesystems dar. Der Suchraum kann auch als Hypothesenbaum betrachtet werden. Bei dieser Betrachtungsweise ist die Komponente «Mailsystem» äquivalent der Hypothese «das Mailsystem ist fehlerhaft», und die Subkomponenten sind spezifischere Hypothesen, wie etwa «Decnet ist fehlerhaft». Die Problemlösungskomponente für die strukturbasierte Diagnose versucht nun, ausgehend von der am wenigsten spezifischen Hypothese, diese zu widerlegen oder zu bestätigen. Eine Hypothese wird entweder durch die Bestätigung einer spezifischeren Hypothese oder durch einen Test bestätigt.

Aus den obigen typischen Inferenzstrukturen wurde nun die vollständige Inferenzstruktur *(Fig. 5)* der Expertise des Projektes konstruiert. Die Kästchen stellen Inferenzen dar. Die benannten Pfeile sind die Metaklassen. Die folgende Liste enthält die Beschreibung der Inferenzen

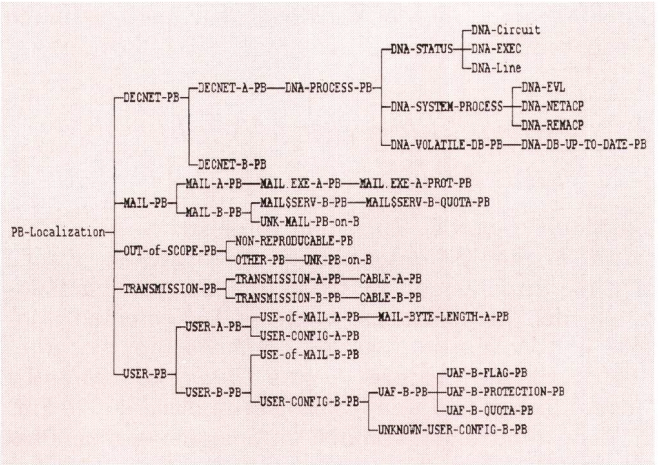

Fig. 4 Vereinfachter Strukturbaum
PB steht für Problem

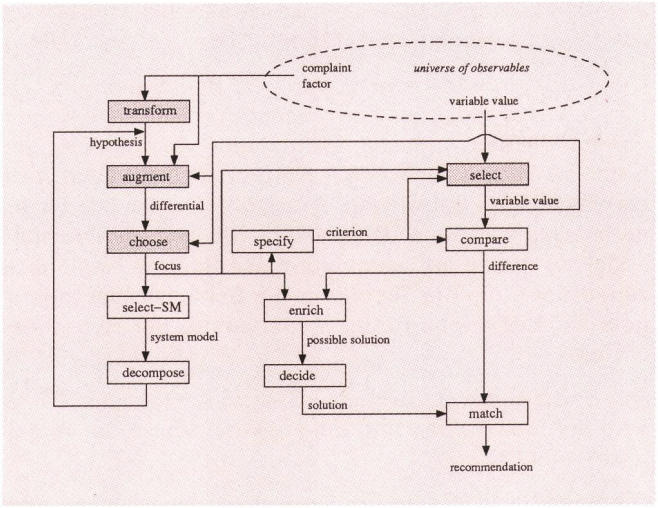

Fig. 5 Vollständige Inferenzstruktur des wissensbasierten Systems Mail-Diagnostic

und ihrer Metaklassen, die im weiteren verwendet werden:

- *Transform* erzeugt für die «complaints» die «hypothesis». Ein *complaint* ist eine Fehlermeldung. Eine *hypothesis* ist eine mögliche Fehlerquelle. Die Wurzel des Strukturbaums ist eine «hypothesis», wenn die heuristische Diagnose nicht angewandt werden kann.
- *Augment* erzeugt für die «complaints» die «differentials». Ein *differential* ist ein möglicher Ausgangspunkt für eine Diagnose. Also erzeugt «augment» auf Grund der möglichen Fehlerquellen die Ausgangspunkte für die Diagnose. Dabei wird entweder ein «complaint» direkt mit einer Hypothese assoziiert, oder es werden gemäss der auf der Struktur basierten systematischen Diagnose die Subkomponenten der Wurzel des Strukturbaums ausgewählt.
- *Choose* ermittelt aus den möglichen «differentials» den «focus». Der *focus* ist der erfolgversprechendste Ausgangspunkt für die Diagnose. Dies heisst also, dass aus einer Anzahl von Hypothesen die wahrscheinlichste ausgewählt wird.
- *Select* assoziiert zu einem «criterion» unter Berücksichtigung des «focus» direkt die «variable values», die für die Prüfung, ob das «criterion» erfüllt ist, notwendig sind. Ein *criterion* ist ein Test, mit dem eine Tatsache festgestellt werden kann. Dabei werden die aktuellen Werte bestimmter Parameter, die *variable values,* von Komponenten des Netzwerks mit den Soll-Werten verglichen. «Select» «weiss», welche Werte für ein «criterion» benötigt werden. Der Vergleich wird mit der Inferenz *compare* durchgeführt.

Die so erhaltene Inferenzstruktur hilft dem Wissensingenieur bei den weiteren Befragungen und der Analyse, da sie die Einordnung von neuem Wissen in das Modell erleichtert. Die ersten beiden Tätigkeiten werden so oft iterativ durchgeführt, bis die ersten beiden Ebenen des konzeptuellen Modells erstellt wurden. Zu bemerken ist, dass auch die anderen Schritte der Entwicklung iterativ durchgeführt werden.

Die Inferenzstruktur ist also eine Beschreibung aller Grundfunktionen, die der Experte für die Problemlösung verwendet. Die Konzepte der Domäne werden zu Klassen zusammengefasst, die in dem Diagnoseprozess unterschiedliche Rollen spielen können. Solche Klassen und ihre Rollen werden als Metaklassen bezeichnet. Als Beispiel kommen alle Komponenten des Strukturbaums, (die im Prinzip auch alle einen Fehler verursachen können), in den Metaklassen «hypothesis», «differential», und «focus» vor. Die Domain-Ebene ist sehr umfangreich, da darin alle Konzepte beschrieben werden. So zum Beispiel sind alle Fehlermeldungen zu der Metaklasse «complaint» zusammengefasst.

Im zweiten Analyseschritt (Analyse/Entwurf) werden aus den verbalen Daten und der Inferenzstruktur die Aufgaben- und Strategieebenen ermittelt. Letztere ist in diesem Projekt nicht vorhanden, da in der Expertise keine Aufgabe vorhanden ist, die einen flexiblen Kontrollfluss benötigen würde. Die Aufgaben können mit Hilfe einer formalen Sprache wiedergegeben werden, die einer Programmiersprache ähnlich ist. Eine vereinfachte Version der Aufgabe «diagnose» ist in *Figur 6* dargestellt. Die Aufgaben und die Inferenzen sind den Prozeduren von Programmiersprachen ähnlich. Die Ein- und Ausgabeparameter sind mit «in» und «out» gekennzeichnet. Die Aufrufe von Unteraufgaben sind kursiv gedruckt.

22 Verwirklichung

Der Entwurfsprozess und das Modell «Design» werden nicht behandelt. Ein funktionales Modell ist aber im ersten Kapitel (Fig. 1) zu finden. Um die Implementation zu verstehen, ist eine kurze Einführung in die Expertensystem-Entwicklungsumgebung Smeci nötig. Eine Entwicklungsumgebung stellt dem Benützer unter anderem Formalismen für die Darstellung von Wissen und eine generische Problemlösungskomponente zur Verfügung. Smeci ist in unserem Fall geeignet, da das Modell des Entwurfs dem Grundgerüst eines wissensbasierten Systems, das mit Smeci realisiert werden kann, ähnlich ist.

Das Wissen wird in Smeci wie folgt dargestellt:

- Durch Objekte des *Objektsystems.* Die Objekte sind Abstraktionen realer und ideeller Objekte. Das Objekt-

```
aufgabe diagnose (in: complaints, universe_of_observables ; out: solution)

        declaration: differentials, hypothesis;

            create (complaints, universe_of_observables, differentials);
            do
                refine (differentials,  universe_of_observables, solution);
            until "solution or no further refinement possible"
end diagnose

aufgabe  create (in:complaints, universe_of_observables ;out:  differentials)

        declaration: differential, complaint;

        differentials := { };
        for each complaint of complaints do
            transform (complaint, universe_of_observables, hypothesis);
            augment (hypothesis, differential);
            differentials:= "union of differentials and differential";
        end for
end create
```

Fig. 6 Vereinfachte Darstellung der Aufgabe «Diagnose»

system besteht aus Klassen und deren Instanzen. Die Klassen definieren die Struktur der Objekte. Den Klassen sind Funktionen zugeordnet, genannt *Methoden.* Die Struktur wird durch *Slots* definiert. Ein Beispiel wäre die Klasse «Komplexe Zahlen» mit den Slots «Realteil» und «Imaginärteil». Die Methoden sind die Operationen auf den komplexen Zahlen. Die Instanzen einer Klasse heissen *Objekte (objects).* In dem Beispiel sind dies komplexe Zahlen, etwa $2i+3$.
- Durch einen Baum von *Regelbasen (rule bases).* Eine Regelbasis enthält die Regeln für die Lösung eines Teilproblems. In jeder Regelbasis kann die Art der Abarbeitung der Regeln spezifiziert werden.

Das Grundgerüst der Problemlösungskomponente von Smeci bilden folgende Mechanismen:

- Die *Agenda* ist ein Stapel von Regelbasen. Sie bestimmt dynamisch die Reihenfolge der Bearbeitung der Regelbasen, indem jeweils die oberste Regelbasis der Agenda abgearbeitet wird.
- Das System befindet sich in einem *Zustand (state),* in dem eine Zahl von Objekten, deren Slots gewisse Werte haben, vorhanden ist und die Problemlösungskomponente eine bestimmte Regelbasis zu bearbeiten hat. Bei der Bearbeitung einer Regelbasis werden Regeln abgearbeitet. Bei der Abarbeitung einer Regel wird ein neuer Zustand erzeugt, da die Regel Werte in Objekten ändert. Bei der Abarbeitung der Regelbasis werden je nach der Spezifikation der Abarbeitungsart ein oder mehrere neue Zustände erzeugt. Aus dieser Menge von Zuständen ist der nächste zu bearbeitende Zustand auszuwählen. Die Art der Auswahl kann der Benützer von Smeci definieren.

Im folgenden werden die Regelbasen und die Regeln an Hand eines Beispiels erklärt. Dabei wird auch der Zusammenhang zwischen dem konzeptuellen Modell und der Implementation, dem Modell-Code, gezeigt.

Der Strukturbaum (Fig. 4) definiert den Suchraum. Da für die Diagnose jedes Komponenten spezifische Regeln vorhanden sein müssen, kann jeder Komponente ein Satz von Regeln zugeordnet werden. In Smeci wird also ein Regelbasis-Baum *(Fig. 7)* erstellt. Dieser ist mit einem elektronischen Diagnosehandbuch vergleichbar. Jeder Komponente oder Hypothese ist eine Regelbasis zugeordnet, die die Tests und die Schlussfolgerungsregeln zur Bestätigung oder Widerlegung der Hypothese umfasst.

Am Beispiel der Regelbasis «Search-in-Decnet» *(Fig. 8)* wird nun der Aufbau und die Funktionsweise der Regelbasen erklärt.

Die Regelbasis «Search-in-Decnet» ist eine Instanz oder ein Objekt der Klasse «S.RuleBase», die alle Regelbasen definiert. Der Slot «rule-list» enthält die Liste der Regeln zur Lösung des Teilproblems «Ist die Fehlerursache Decnet?». Diese Regeln führen Tests durch, um festzustellen, ob Decnet die Fehlerursache ist. Ist dies der Fall, so bestimmen die Regeln, ob der Fehler auf dem Decnet des Sender- oder Empfänger-Rechners zu suchen ist. Der Slot «sub-rule-base» enthält die Liste der Teil-Regelbasen dieser Regelbasis. Dies sind die Diagnose des Decnets auf dem Rechner des Senders (A) und des Empfängers (B).

Nun werden die zwei Regeln der Regelbasis der Figur 8 detailliert erklärt. Die folgende Regel fordert den Benützer auf, einen Test durchzuführen und das Ergebnis dem System einzugeben. Dieser Test könnte auch automatisch ausgeführt werden.

DefRule Perform-SetHost

Die Regel implementiert einen Teil der Inferenz «select»

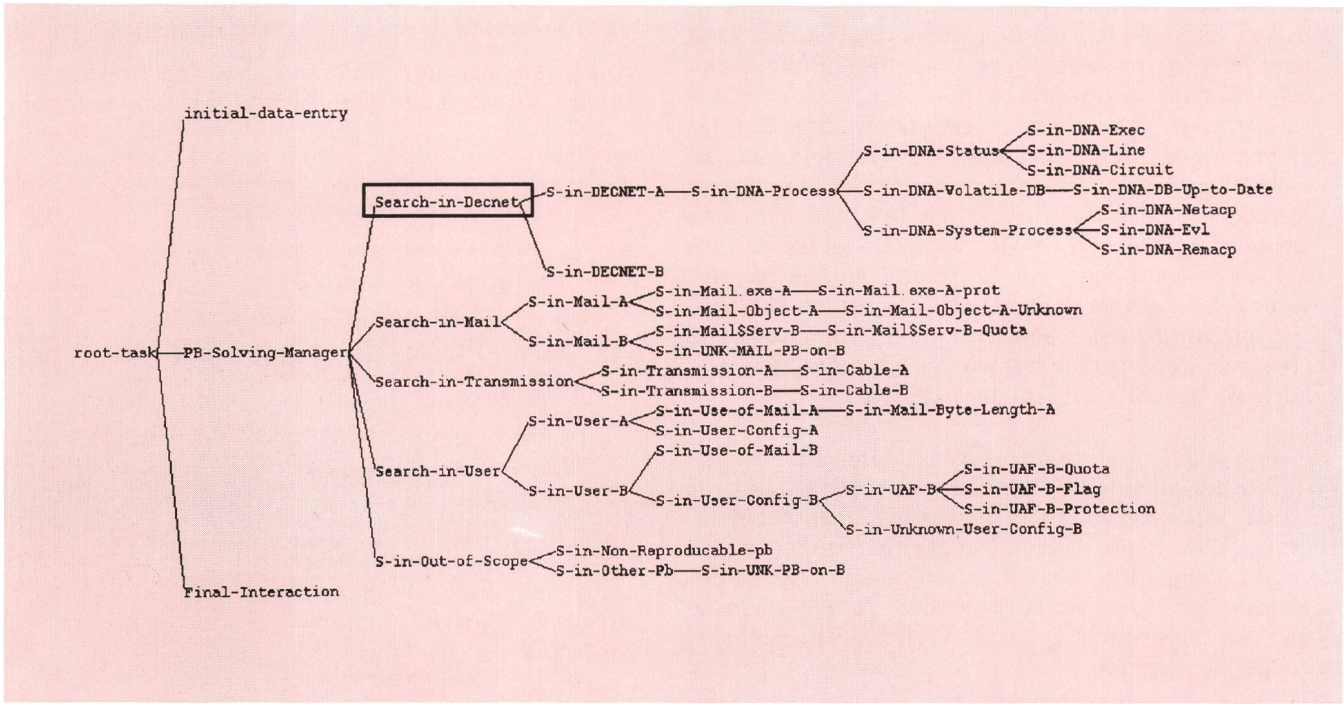

Fig. 7 Ein vereinfachter Regelbasis-Baum von Mail-Diagnostic

```
Object Search-in-Decnet                Object of Class S.RuleBase
       rule-list                       = List of
                                              ...
                                              Perform-SetHost
                                              ...
                                              If-Perform-SetHost-ok-then-focus-mail-problem
                                              ...
       sub-rule-base                   = List of
                                              S-in-DECNET-A
                                              S-in-DECNET-B
```

Fig. 8 Die Regelbasis «Search-in-Decnet»

Wenn das folgende Objekt vorhanden ist
other-data
und
keine Test-Meldung zum Referenzrechner* gesandt werden kann
dann
frage den Benützer, ob eine SET HOST B möglich ist.
Die Antwort («yes» oder «no») ist dem Slot «set-host-functions» des Objekts «other-data» zuzuweisen.

Die Regel implementiert die Inferenz «select» für den «focus» Decnet und das «criterion» «Ist SET HOST B möglich?». Dieses «criterion» ist ein Test, der feststellt, ob Decnet funktioniert, im Falle, dass vom Sender-Rechner aus keine Meldung erfolgreich versandt werden konnte.

DefRule If-Perform-SetHost-ok-then-focus-mail-problem
Die Regel implementiert einen Teil der Inferenzen «augment» und «choose»
Wenn die folgenden Objekte vorhanden sind
other-data, mail-pb
und
der Wert des Slots «set-host-functions» des Objekts «other-data» den Wert «yes» hat und der Slot «problem-here» des Objekts «mail-pb» keinen Wert hat
dann
führe die Regelbasis «mail-pb» an erster Stelle in die Agenda ein. Fahre mit der Abarbeitung der Regelbasis «mail-pb» fort.

Da der Systembetreuer des Sender-Rechners Fernzugriff über Decnet auf den Empfänger-Rechner hat, folgert die Regel, dass der Fehler in der Anwendung Mail zu suchen ist. Auf Grund des «complaints» «SET HOST B funktioniert» ermittelt die Regel als «differential» «Mail Problem». Dieser Teil der Regel gehört zur Implementation der Inferenz «augment». Im weiteren wählt die Regel aus den möglichen «differentials» «Mail Problem» als «focus» aus. Dies ist Teil der Implementation der Inferenz «choose».

Die beiden Beispiele zeigen, dass in der Implementation mit Smeci die Inferenzen nicht als eigenständige Funktionen, die auf dem Wissen der Domäne operieren, implementiert sind. Die Abstraktion der Inferenz wird für jede Inferenz durch eine grössere Zahl von Regeln umgesetzt. Jede der Regeln implementiert dabei eine oder mehrere Inferenzen für ganz bestimmte Konzepte der Domäne. Die Metaklassen sind Abstraktionen der Konzepte der Domäne.

Im Formalismus der Implementationsregeln wird nicht ausdrücklich zwischen der Domän- und Inferenzebene unterschieden. Die Regeln werden über Kommentare den Inferenzen zugeordnet. Der in der Aufgabenebene modellierte Kontrollfluss ist in den Regeln und der Problemlösungskomponente einbezogen. Die explizite Unterscheidung der Ebenen erleichtert die Wartung des konzeptuellen Modells.

3 Schlussfolgerungen

Folgende wichtigen technischen Erfahrungen wurden im Projekt gemacht:

- Das konzeptuelle Modell ist ein sehr umfangreiches Dokument. Da die Elemente des Modells zueinander in Beziehung stehen, müsste für die produktive Anwendung von KADS ein Werkzeug zur Erstellung und Wartung des konzeptuellen Modells verwendet werden. Im Rahmen des Esprit-Projekts KADS II wird ein solches entwickelt [4].
- Es hat sich gezeigt, dass der Wissensingenieur die Domäne gut verstehen muss. Dieses Verständnis kann er sich entweder bei der Analyse der Expertise erarbeiten, oder er bringt es schon mit. Im ersten Fall ist der Aufwand für die Analyse grösser, und der Experte übt auch noch die Funktion eines Lehrers aus. Im zweiten Fall könnte der Aufwand für die Analyse im Prinzip kleiner sein. Es treten meistens mehr Missverständnisse auf, die mit einigem Aufwand korrigiert werden müssen. Daher sind beide Fälle etwa gleichwertig. Diese Abhängigkeiten können bei der Wis-

* Der Referenzrechner ist ein Rechner, von dem angenommen werden kann, dass er korrekt funktioniert. Damit kann die Diagnose vereinfacht werden. Ist kein solcher Rechner vorhanden, so könnte dieser durch eine ungerade Zahl k verschiedener Rechner ersetzt werden (k > 1). In diesem Fall wären k Tests durchzuführen. Wenn alle k Tests negativ verlaufen, so könnte mit grosser Wahrscheinlichkeit geschlossen werden, dass der Sender-Rechner fehlerhaft funktioniert.

senserhebung durch einen Wissensingenieur prinzipiell nicht durchbrochen werden.
- KADS ist eine aufwendige Methode, aber sie ermöglicht die Modellierung der Expertise unabhängig von der späteren Implementation. Dadurch ist eine Dokumentation vorhanden, die auch für die Wartung und Weiterentwicklung verwendet werden kann. Somit ist KADS eine Software-Engineering-Methode für wissensbasierte Systeme. Sie ist aber noch nicht vollständig entwickelt. Für einfache und kleine wissensbasierte Systeme, die nicht weiterentwickelt werden sollen, können auch einfachere Methoden verwendet werden, wie die Erfahrung aus anderen Projekten zeigt.

Die Ziele des Forschungsprojekts wurden weitgehend erreicht. Die Projektgruppe verfügt nun über ein gewisses Know-how im Fachgebiet wissensbasierter Systeme. Damit sind deren Mitglieder nun in der Lage, die Entwicklung in diesem Bereich weiterzuverfolgen sowie eine gewisse Beratung für wissensbasierte Systeme anzubieten. Im besonderen kann nun bei der Abklärung der Anwendbarkeit der Technologie und der Machbarkeit wissensbasierter Systeme geholfen werden. Auch Fragen der Wissenserhebung, Modellierung und Ausführung können beantwortet werden. Die Entwicklung des Fachgebietes wird weiterverfolgt, um den möglichen Nutzen solcher Systeme für die PTT abzuschätzen und die Kenntnisse auf diesem Gebiet zu vertiefen.

Bibliographie

[1] *Liver B.* und *Prim A.* Wissensbasierte Systeme im Netzwerkmanagement. F&E-Projekt Nr. 180, V-Bericht VI2.2008 U, GD PTT, Bern, 1991.

[2] *Decitre P.* The KADS Methodology: A de-facto standard at hand, Cap Gemini Innovation, August 1990.

[3] *Boon A. C., Koopman M. R.* and *Dillema G.* The development of a knowledge-based system for diagnosing a mail-handling machine, PTT Research, Netherlands, 11. International Conference on Expert Systems & their Application, Vol. 3, p. 137—148, Avignon 1991.

[4] *Brunet E., Bouchet C.* and *Anjewierden A.* Shelley, an integrated workbench for KBS development, 9. International Conference on Expert Systems & their Application, Avignon 1989.

Zusammenfassung

Wissensbasierte Systeme im Netzwerkmanagement

Teil 2: Erfahrungsbericht

In diesem zweiten Teil wird das Ergebnis des Forschungsprojektes «Wissensbasierte Systeme im Netzwerkmanagement» vorgestellt. Zuerst wird die Auswahl und grobe Definition einer geeigneten Anwendung, der vorhandene Prototyp und dessen Weiterentwicklung in einem Folgeprojekt beschrieben. Danach wird die Entwicklung der verschiedenen Modelle nach der Methodologie KADS dargestellt. Der Schwerpunkt liegt auf dem konzeptuellen Modell und dessen Implementation.

Résumé

Systèmes-expert pour la gestion de réseaux

2e partie: Compte rendu des résultats

La deuxième partie de cet article résume les résultats du projet (systèmes-expert pour la gestion de réseaux). On décrit d'abord le choix d'une définition sommaire de l'une des applications appropriées, le prototype existant et son perfectionnement au cours d'un projet subséquent. L'article se poursuit par la description du développement de divers modèles selon la méthodologie KADS. L'accent est mis sur le modèle conceptuel et son implémentation.

Riassunto

I sistemi basati sulla conoscenza nella gestione delle reti

Seconda parte: Resoconto delle esperienze

In questa parte gli autori presentano il risultato del progetto di ricerca «Sistemi basati sulla conoscenza nella gestione delle reti». Essi descrivono la procedura adottata per scegliere e definire sommariamente un'applicazione adeguata, il prototipo esistente e l'ulteriore suo sviluppo in un progetto successivo. Illustrano quindi lo sviluppo dei diversi modelli secondo il metodo KADS. Essi attribuiscono grande importanza al modello concettuale e alla sua implementazione.

Summary

Knowledge Based Systems for Network Management

Part 2: Experience Report

In this second part the conclusion of the research project, «Knowledge Based Systems in Network Management» is described. First, the selection and rough definition of a suitable application of the existing prototype and its development in a following project is described. Then the development of the different models according to the KADS method is described. The emphasis is on the conceptual model and its implementation.