chern. Nicht realisiert ist dagegen die Dokumentation des gesamten Wortlauts der Entscheidungen, da der Aufwand des Abtippens den Ertrag nicht rechtfertigen würde. Die gespeicherten Kurztexte sowie weiterführenden Hinweise reichen erfahrungsgemäß völlig aus, die Entscheidung zu treffen, ob das Nachlesen der gesamten Entscheidung erforderlich ist oder nicht.

4. Dateistruktur ändern

Eine gewählte Dateistruktur kann jederzeit mit dem Befehl „LIST STRUCTURE" bzw. nach Betätigung der Funktionstaste 5 auf dem Bildschirm angezeigt oder auch ausgedruckt werden. Eine Änderung erfolgt mit dem Befehl „MODIFY STRUCTURE". Es können sowohl die Feldnamen als auch die Typen und Längen geändert werden. Der Dateiinhalt bleibt erhalten, wenn die Änderung nur jeweils an einer Stelle erfolgt und sodann abgespeichert wird. Ebenso können weitere Felder hinzugefügt, dazwischengeschoben oder gelöscht werden. Im oberen Drittel des Bildschirms werden entsprechende Hilfen für die Benutzerführung eingeblendet.

Feldverlängerungen führen zu keinem Datenverlust; bei Verkürzungen werden die Sätze hinten entsprechend abgeschnitten. *(wird fortgesetzt)*

TWAICE - eine Expertensystem-Shell (Teil 2)

Armin Leicht/Volker Siebelink

E. Knowledge Engineering

I. Begriffliches

Als Knowledge Engineering wird der Prozeß bezeichnet, der das Expertenwissen in eine für den Computer verarbeitungsfähige Form überträgt. Die Effizienz eines Expertensystems steigt mit der Anzahl der Regeln und dem Basiswissen. Dabei sind jedoch drei Probleme zu überwinden: das Formulieren des Expertenwissens, das Verarbeiten divergierender Ansichten zu einem Problem und das Übersetzen in eine computerverständliche Sprache. Als Wissensquellen können dabei Experten, Fachbücher, Unterlagen etc. herangezogen werden.

Dem Knowledge Engineer fällt nun die Aufgabe zu, in Interviews mit dem Experten die einzelnen Fakten und die Beziehungen zwischen den einzelnen Informationen herauszufiltern und zu interpretieren, um sie später in die Form von „If-Then-" (Wenn-Dann-) Regeln zu transformieren. Hieraus können Expertensysteme mit mehreren hundert bis zu mehreren tausend Regeln entstehen. Zu guter Letzt wird zur Herstellung von Prototypen mit dem Experten und den späteren Benutzern Rücksprache gehalten und diese Prototypen in bestehende Software- und Hardware-Umgebungen des Kunden integriert. Hinsichtlich der Terminologie sei noch erwähnt, daß mit dem Experten der Fachmann mit Spezialwissen angesprochen ist und mit Benutzer der Laie, bzw. der nicht so fachspezifisch ausgebildete Benutzer, der das System bei seiner Arbeit einsetzt.

Bei der Entstehung der Wissensbank arbeiten der Experte und der Knowledge Engineer sehr eng zusammen. Nur über die intensive Auseinandersetzung mit der Domäne des Experten kann das Fachwissen in geeigneter Weise aufbereitet werden. Dabei wächst die Wissensbank durch einen Prozeß iterativer Entwicklung, der das Expertensystem in Breite und Tiefe ausbaut[7].

II. Knowledge Engineering bei Nixdorf

1. Das 5-Stufen-Konzept

Die Firma Nixdorf beschreibt das Knowledge Engineering in einem 5-Stufen-Konzept[8]. Dies sei an dieser Stelle kurz zusammengefaßt:

In **Stufe 1** ist es die Aufgabe des Knowledge Engineers bei der Erstellung der externen Wissensbank, das Wissen des Experten zu analysieren und in geeignete Regeln umzusetzen. Hierzu müssen sich sowohl der Knowledge Engineer als auch der Experte mit dem Fachgebiet des jeweils anderen auseinandersetzen und vertraut machen, da sonst mit erheblichen Verständigungsschwierigkeiten zu rechnen ist.

Der Knowledge Engineer wird sich also in die Wissensdomäne des Experten einarbeiten, um einen Überblick in dessen Fachgebiet zu erhalten. Andererseits muß er dem Experten die Grundlagen eines Expertensystems soweit erklären, daß dieser eine Vorstellung davon erhält, in welcher Form sein Wissen gefragt ist. Der Knowledge Engineer wird am besten dem Experten ein lauffähiges Expertensystem mit mindestens einer Wissensbank aus einer allgemein verständlichen Domäne vorführen und ihm so die grundlegenden Prinzipien eines Expertensystems verdeutlichen. Gemeinsam werden beide nun analysieren, welche Probleme von dem neuen Expertensystem gelöst werden sollen, wo das Expertensystem eingesetzt werden soll und wie die Umgebung aussieht, da hiervon das spätere Erscheinungsbild der Dialogkomponente abhängt.

Die Erstellung der externen Wissensbank in **Stufe 2** ist ein Prozeß, der sich nur schrittweise durchführen läßt. In ausgiebigen Gesprächen muß der Knowledge Engineer das Wissen des Experten analysieren und strukturieren. Problematisch hierbei ist, daß es sich bei

[7] hierzu: *Nölke* in Savory, Künstliche Intelligenz und Expertensysteme, 2. Aufl., Oldenburg 1985, S. 110 f.

[8] s. Fn(7), dort S. 112 ff.

dem Wissen des Experten nicht nur um Faktenwissen handelt, sondern auch um strategisches Wissen. Faktenwissen ist die Form des Wissens, wie sie häufig in der Fachliteratur zu finden ist, nämlich das Wissen, das zeigt, welche Aufgaben in der jeweiligen Wissensdomäne gelöst werden müssen und wo diese Aufgaben auftreten. Das strategische Wissen ist das Wissen, das den eigentlichen Experten ausmacht. Er hat Erfahrung in seinem Fachgebiet und kann daher von bereits bekannten Problemen auf die Lösung neuer Probleme schließen. Außerdem ist es ihm möglich, gestellte Aufgaben mit einer gewissen Intuition zu lösen. Auch diese Form des Wissens muß der Knowledge Engineer erkennen und in Regeln umsetzen können. Während dieser Gespräche sollten schon drei einfache Testfälle aus der Wissensdomäne erarbeitet werden, mit deren Hilfe später das fertige Expertensystem getestet wird.

Mit Hilfe eines geeigneten Editors werden die Regeln der externen Wissenbank erstellt bzw. geändert. Ist dann das gesamte Fachwissen des Experten in die entsprechenden Regeln umgesetzt, läßt der Experte einige ausgewählte Fälle aus der Testlaufbibliothek laufen **(Stufe 3)**. Es ist darauf zu achten, daß diese Fälle alle Teilaufgaben des Expertensystems ansprechen, um sicher zu gehen, daß nicht noch versteckte Fehler vorhanden sind. Unabhängig voneinander müssen der Experte und das Expertensystem dieselben Testfälle lösen, um so festzustellen, wo die Fehler des Expertensystems noch liegen. Auf diese Art und Weise läßt sich die *Konsistenz*[9] der Wissensbank überprüfen. Sollten in diesem Schritt Probleme auftreten, so müssen Knowledge Engineer und Experte die Fälle schrittweise durchgehen und somit die einzelnen benutzten Regeln auf ihre Gültigkeit testen. Wenn das Expertensystem bis zu diesem Punkt fehlerfrei läuft, muß der Experte solange neue Fälle durchspielen, bis entweder neue Probleme auftreten oder er den Eindruck hat, daß das gesamte System fehlerfrei ist. Treten in diesem Stadium Fehler auf, so muß wiederum das Regelwerk entsprechend genau auf seine Konsistenz hin untersucht und die Fehler korrigiert werden. Auch sollte zu diesem Zeitpunkt der Experte in der Lage sein, selbständig neue Regeln zu erstellen und bereits vorhandene Regeln abzuändern. Bei Fehlern in diesem Entwicklungsstadium handelt es sich meistens um Fehler in den Ein- und Ausgaberoutinen[10], um falsche Regeln oder um falsch gewählte Strategien bei der Lösungsfindung.

Im nächsten Schritt wird das Expertensystem in seine Umwelt eingepaßt **(Stufe 4)**. Es wird in die bestehenden Organisationsabläufe und Computersysteme integriert, und die zukünftigen Benutzer werden mit dem System und seiner Arbeitsweise vertraut gemacht.

Einige Monate nach Einführung des Expertensystems beginnt die **Stufe 5**. In ihr erfolgt der Erfahrungsaustausch zwischen dem Knowledge Engineer auf der einen Seite sowie den gegebenenfalls weiter hinzugezogenen Experten und Benutzern auf der anderen Seite. Die voneinander unabhängigen Experten sollen eine Bewertung über das System abgeben; aus dieser Bewertung lassen sich Rückschlüsse über Vollständigkeit und Konsistenz des implementierten Wissens herleiten. Die Anwender sollen Verbesserungsvorschläge und Anregungen machen, denn sie müssen in der Praxis mit dem System arbeiten. Aufgrund der so gemachten Erfahrungen sollen der Experte und der Knowledge Engineer das System abschließend bewerten und eventuell noch anstehende Verbesserungen und Verfeinerungen vornehmen.

2. Typen des Knowledge Engineering

Zur Erstellung eines kundenspezifischen Expertensystems kann je nach Bedarf und Wunsch des Kunden zwischen mehreren Typen des Knowledge Engineering gewählt werden. So kann entweder ein Mitarbeiter des Kunden in Lehrgängen zum Knowledge Engineer ausgebildet werden, der dann mit Hilfe einer leeren TWAICE-Shell und einigen Beispielprogrammen im eigenen Haus das gewünschte Expertensystem entwikkelt. Oder, der Knowledge Engineer der Firma Nixdorf entwickelt zusammen mit dem Experten des Kunden die Wissensbank (wie oben beschrieben). Außerdem besteht die Möglichkeit, daß der Knowledge Engineer den Experten in der Prototypenphase (bis einschließlich Stufe 2) zum selbständigen Arbeiten mit dem System ausbildet. Zur Zeit versucht Nixdorf auch die räumliche Differenz zu den Kunden zu verringern, indem sie in den einzelnen Zweigstellen in der Bundesrepublik Knowledge Engineers abstellt, die dann jederzeit und schneller vor Ort die Ausbildung und Beratung der Kunden vornehmen können.

F. Erstellen eines Expertensystems mit TWAICE

I. Geschichte von TWAICE

Die geschichtliche Entwicklung des Shell-Konzeptes reicht bis in die 70er Jahre zurück, als problemlösende Systeme mit großen Wissensdatenbanken über fachspezifisches Wissen entwickelt wurden. Jedoch war deren Aufbau unwirtschaftlich und erforderte umfangreiche Programmierkenntnisse bei der Implementierung der Wissensbank. Mithin wurde zur Ausschaltung dieser Mängel die Erfahrung verwertet, daß mit einer Trennung von Wissensbank und dem steuernden Programm eine größere Flexibilität erreicht werden kann. Dies erleichtert auch die Modifizierung der Wissensbanken. Über die Programmierung einer fachgebietsunabhängigen Expertensystem-Shell wurde ein Werkzeug geschaffen, das die Erzeugung von Expertensystemen beliebiger Fachrichtungen erlaubt. Damit wurden diese Systeme gleichzeitig servicefreundlicher und vor allem marktfähig[11].

[9] Die Wissensbank ist dann konsistent, wenn die Ergebnisse der Testfälle mit den erwarteten Ergebnissen identisch sind. Weiterhin dürfen keine Schleifen im Regelwerk auftreten und es darf keine sich widersprechenden Regeln geben.

[10] d.h. ungenaue, schwer verständliche oder zu wortreiche Angaben des Systems.

[11] vgl.: *Mescheder* in Savory, s. Fn(7), S. 57, 58.

Nach ersten Impulsen aus USA seit 1982/83 und über Kontakte mit amerikanischen Firmen wurde bei der Nixdorf AG die Entwicklung einer eigenen Shell in Angriff genommen. Ursprünglich hatte man als potentiellen Anwenderkreis nur den technischen Kundendienst ins Auge gefaßt. Konzipiert war sie für die Fehlerdiagnose und Konfigurationen von Computersystemen, aber nicht für den konventionellen Bereich. Nachdem sich auf dem Markt erhebliche Veränderungen in der Nachfrage und bezüglich des Anforderungsprofils einstellten, wurde das Konzept geändert. Zur Zeit stellen Banken und Universitäten die Hauptanwender der Shell dar.

II. Erstellen der Wissensbank

Damit man sich einmal eine Vorstellung von Art und Umfang der Arbeit beim Erstellen einer Wissensbank machen kann, soll an dieser Stelle anhand der wichtigsten Werkzeuge eine kurze Einführung in das Erstellen einer Wissensbank gegeben werden. Diese Miniatur-Wissensbank soll nur fünf Regeln statt mehrerer Tausend beinhalten und inhaltlich Vertriebsbeauftragte beim Kauf von Aktienzertifikaten beraten unter Berücksichtigung der jeweiligen Ersparnisse und des gewünschten Anlagezeitraumes.

1. Laden und Namensgebung

Der Ladevorgang der TWAICE-Shell entspricht dem bereits oben dargestellten (vgl. Beginn der Konsultation). Damit nicht unautorisierte Personen, hierzu zählt gegebenenfalls auch der Endbenutzer, neue Wissensbanken aufbauen oder sogar mit verheerenden Folgen ändern können, muß zusätzlich zu dem Benutzer-Passwort ein weiteres Kennwort eingegeben werden, das nur der Knowledge Engineer und/oder der Experte kennt.

Bevor mit dem Erstellen der Wissensbank begonnen werden kann, muß diese noch benannt werden. TWAICE benötigt die Bezeichnung zur Erzeugung von Namen für die Dateien, in denen das Wissen später abgelegt werden soll. Im folgenden Beispiel trägt die Wissensbank den Namen *„invest"*.

2. Erfassen des Wissens

Danach erscheint das Standard-Menue *(vgl. Bild 3 im 1. Teil des Beitrags)* auf dem Bildschirm. Hieraus wählt der Bearbeiter (Knowledge Engineer oder Experte) Option 5 *(bearbeitete Systemumgebung)* und aus einem weiteren Menue ruft er den Editor auf[12].

a) Die Regeln

Die wichtigste Form der Wissensdarstellung ist die Produktionsregel. Im allgemeinen sind uns Regeln in vielfältiger Ausprägung im Alltag präsent. Im juristischen Bereich sind dies z.B. die Gesetzestexte. In Expertensystemen beinhalten die Regeln das Expertenwissen. Aufgrund ihrer hohen Modularität können die einzelnen Regeln ohne Beeinflussung anderer Regeln gelöscht oder hinzugefügt werden. Das erlaubt einen schrittweisen Aufbau und eine einfache Modifizierbarkeit der Wissensbanken[13].

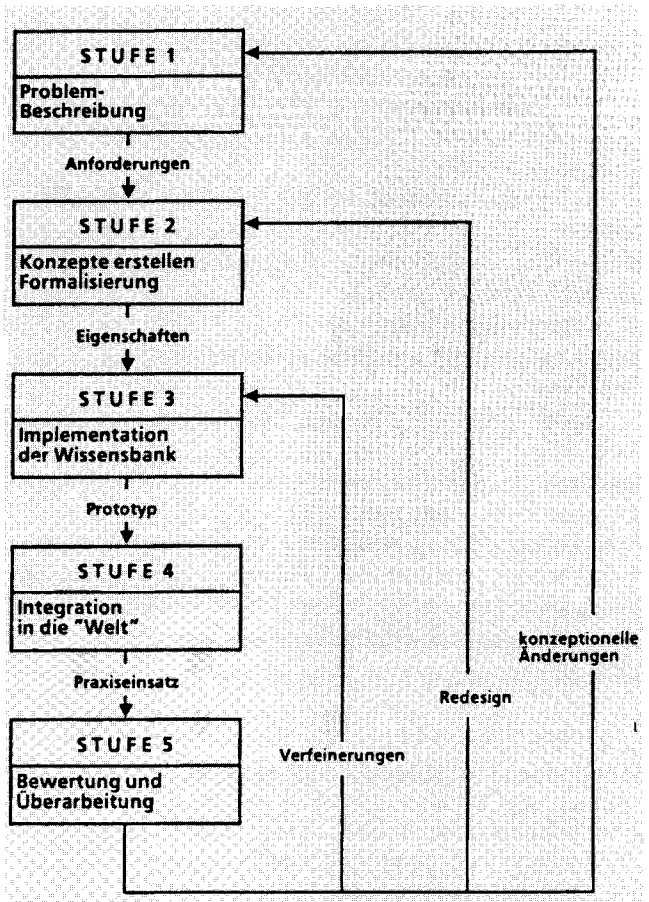

Abb. 15

Jede Regel erhält eine Nummer unter der sie jederzeit wieder angezeigt werden kann. Der Aufbau einer Regel wird durch die Form der IF...Then-Beziehung (Wenn-Dann-Regel) bestimmt. Die Prämissen der Regel haben folgende Form:
‹OBJECT›.‹ATTRIBUT› = ‹WERT›

In **Bild 16** wird der Regelaufbau an einem Beispiel deutlich[14]. Da mit dem Expertensystem Kunden beraten werden sollen, ist es sinnvoll, als Objekt den Begriff *„Kunde"* zu wählen, damit sich der spätere Benutzer nach Auflistung der Regel schneller zurechtfindet. Dem Objekt können eine unbegrenzte Anzahl von Attributen (Eigenschaften) zugeordnet werden. Mehrere

[12] Ein Editor ist ein Hilfsprogramm zur Erstellung von Texten jeglicher Art.

[13] vgl.: *Mescheder* in Savory, s. Fn(7), S. 61.

[14] alle Abbildungen und Beispiele in diesem Kapitel sind der Bedienungsanleitung „TWAICE — 20 Fragen zur Erstellung von Expertensystemen mit TWAICE/Nixdorf AG" entnommen.

Wörter eines Attributs müssen dabei wegen der Prolog-Syntax in Hochkommas gesetzt werden *(vgl. in Bild 16: „Risiko Bereitschaft")*. Im Beispiel ist der Kunde Verkäufer, privater Anleger, hat die höhere Schule besucht und ist bereit, ein hohes Risiko einzugehen.

Außerdem brauchen die Regeln nicht in einer bestimmten Reihenfolge eingegeben werden. Somit sind sie leichter austauschbar und einfacher zu pflegen. In **Bild 17** sind die restlichen vier Regeln des Miniatur-Expertensystems aufgeführt. Nach Eingabe aller Regeln werden diese auf Platte gesichert und der Editor verlassen.

Damit TWAICE mit den Regeln arbeiten kann, müssen sie zunächst geladen werden. TWAICE baut während des Ladevorganges automatisch die entsprechenden Regelbäume auf. Dies wird aus **Bild 18** deutlich. Da mit fortschreitendem Laden immer mehr Bezeichnungen in früheren Regeln aufgeführt sind, stellt TWAICE immer weniger Fragen (siehe **Bild 19**). Nach dem Laden baut TWAICE eine provisorische Taxonomie auf.

b) Steuerung des Konsultationsablaufs

Die Taxonomie ist jedoch noch nicht vollständig, da das Wissen zur Steuerung des Konsultationsablaufs noch ergänzt werden muß.

Schließlich kann TWAICE nicht ahnen, wie der Bearbeiter das Wissen verbinden will. Hierzu ist der Objekt-Frame „Kunde" so auszugestalten, daß das Ziel die „vorgeschlagene Anlage" abgeleitet wird. Die Hauptziele müssen selbstverständlich vorher bekannt sein. Das Beispiel in Bild 20 zeigt den Objekt-Frame vor und nach der Bearbeitung. Mit der Angabe *trace EXISTMORE* wird dem System erklärt, daß noch mehr Kunden beraten werden können. Daß mindestens ein Kunde pro Sitzung beraten wird, ist unter *NUMBER* einzutragen. Zudem bewirkt die Angabe unter *INITIALS,* daß zuerst immer der Beruf des Kunden erfragt wird.

c) Bedienerfreundlichkeit

Sollte der Bearbeiter bei der Installation der Wissensbank etwas Wichtiges vergessen haben, fängt TWAICE diesen Fehler selbständig ab, sobald die Konsultation begonnen wird. Zur Abrundung der erstellten Wissensbank können auch die Fragen des Systems, die Benutzertexte und Rattexte ganz individuell vom Bearbeiter abgeändert werden. Bei diesen Textarten kann sogar eine andere Sprache benutzt werden, so daß dieselbe Wissensbank auch in anderen Ländern Verwendung finden kann.

Sicherlich bietet TWAICE noch mehr Möglichkeiten und der Bearbeiter hat noch einige andere Maßnahmen bei der Erstellung einer Wissensbank zu beachten, prinzipiell ist damit aber das Wichtigste dargestellt.

```
RULE 10
IF Kunde . Beruf = Verkäufer
THEN Kunde . Anlegertyp = privat
AND Kunde . Ausbildung = höhere Schule
AND Kunde . 'Risiko Bereitschaft' = hoch
END
```

Abb. 16

```
RULE 20
IF Kunde . 'Urteil des Beraters' = 'geeignet für Aktien'
AND Kunde . Anlegertyp = privat
AND Kunde . 'Anlagezeitraum (Jahre)' >= 3
AND Kunde . Besteuerungsland = Deutschland
THEN Kunde . 'vorgeschlagene Anlage' = 'BRD Schatzbriefe' (900)
END

RULE 30
IF Kunde . 'finanzielle Reserven' = liquide
THEN Kunde . 'Urteil des Beraters' = 'geeignet für Aktien'
END
```

Abb. 17

```
Regeln werden aus der Datei „invest.erl" geladen...
  Regel: 10

RULE    10
IF      Kunde . Beruf = Verkäufer
THEN    Kunde . Anlegertyp = privat
AND     Kunde . Ausbildung = höhere Schule
AND     Kunde . Risikobereitschaft = hoch
END

Das Objekt „Kunde" existiert nicht in der Taxonomie.
Soll es aufgenommen werden? (ja)
>
Zu dem Objekt „Kunde" gibt es kein Attribut „Beruf".
Soll es in die Taxonomie aufgenommen werden? (ja)
>
„Verkäufer" ist kein zulässiger Wert für „Kunde" . „Beruf".
Soll es in die Taxonomie aufgenommen werden? (ja)
>
```

```
Zu dem Objekt „Kunde" gibt es kein Attribut „Anlegertyp".
Soll es in die Taxonomie aufgenommen werden? (ja)
>
„privat" ist kein zulässiger Wert für „Kunde" . „Anlegertyp".
Soll es in die Taxonomie aufgenommen werden? (ja)
>
Zu dem Objekt „Kunde" gibt es kein Attribut „Ausbildung".
Soll es in die Taxonomie aufgenommen werden? (ja)
>
„höhere Schule" ist kein zulässiger Wert für „Kunde" . „Ausbildung".
Soll es in die Taxonomie aufgenommen werden? (ja)
>
Zu dem Objekt „Kunde" gibt es kein Attribut „Risikobereitschaft".
Soll es in die Taxonomie aufgenommen werden? (ja)
>
„hoch" ist kein zulässiger Wert für „Kunde" . „Risikobereitschaft".
Soll es in die Taxonomie aufgenommen werden? (ja)
>
  Regel: 20

RULE 20
IF Kunde . Urteil des Beraters = geeignet für Aktien
AND Kunde . Anlegertyp = privat
AND Kunde . Anlagezeitraum (Jahre) >= 3
AND Kunde . Besteuerungsland = Deutschland
THEN Kunde . vorgeschlagene Anlage = BRD Schatzbriefe (900)
END

Zu dem Objekt „Kunde" gibt es kein Attribut „Urteil des Beraters"
Soll es in die Taxonomie aufgenommen werden? (ja)
>
„geeignet für Aktien" ist kein zulässiger Wert für „Kunde" . „Urteil des Beraters".
Soll es in die Taxonomie aufgenommen werden? (ja)
>
```

Abb. 18

```
Regel: 40
RULE 40
IF Kunde . Ersparnisse (DM) > 15000
THEN Kunde . finanzielle Reserven = liquide
END
Zu dem Objekt „Kunde" gibt es kein Attribut „Ersparnisse (DM)".
Soll es in die Taxonomie aufgenommen werden? (ja)
>
„15000" ist kein zulässiger Wert für „Kunde" . „Ersparnisse (DM)".
Soll es in die Taxonomie aufgenommen werden? (ja)
>
  Regel: 50
Regeln aus der Datei „invest.erl" geladen.
weiter mit CR
```

Abb. 19

```
Welches Objekt möchten Sie ändern?
>Kunde
Welches Objekt möchten Sie ändern? (kein weiteres)
>

Dateiname für die Arbeitsdatei?
(Default ist invest.wrk)
>
Datei „invest.wrk" existiert bereits!
Soll sie überschrieben werden? (nein)
>ja
Bitte editieren Sie die Arbeitsdatei.
Vorher:
        OBJECT          Kunde
        FATHER          root
        INSTANTIATION   trace_EXIST
        NUMBER          >= 0
        INITIALS
        GOALS
        END
Nachher:
        OBJECT          Kunde
        FATHER          root
        INSTANTIATION   trace_EXISTMORE
        NUMBER          >= 1
        INITIALS        Beruf
        GOALS           'vorgeschlagene Anlage'
        END
```

Abb. 20

G. Technische Beschreibung

I. PROLOG und Betriebssystem

Um die technischen Zusammenhänge bei der Expertensystem-Shell TWAICE vollständig verstehen zu können, ist es notwendig, zuvor in groben Zügen die Programmiersprache PROLOG zu beschreiben[15].

Die bisher bekannten Programmiersprachen (Basic, Fortran, C, usw.) gehören zu der Gruppe der prozeduralen Sprachen. Bei ihnen werden die Befehle eines Programms genau in der vom Programmierer festgelegten Reihenfolge abgearbeitet. In der Praxis des Programmierens werden jedoch häufig Befehle auftauchen, bei denen es egal ist, in welcher Reihenfolge sie vom Rechner bearbeitet werden; vielfach ist es von der Aufgabenstellung her sogar möglich, diese Befehle parallel abzuarbeiten. Anweisungen, die diese Möglichkeit bieten, heißen nicht-prozedural oder auch nicht-algorithmisch.

Bereits im Jahre 1972 gab es in Marseille die erste PROLOG-Implementierung. Bei PROLOG wird nicht so sehr Wert auf die numerischen Fähigkeiten gelegt, sondern die Betonung liegt auf der Lösbarkeit logischer Probleme. Auch wird dem Programmierer die Arbeit dahingehend erleichtert, daß er nicht mehr die Reihenfolge der Programmschritte festlegen muß, sondern sich voll und ganz auf die Beschreibung des Problems beschränken kann.

In PROLOG gibt es drei wesentliche Kontrollstrukturen, die hier kurz erläutert werden sollen. Es sind dies die Mechanismen Unifizierung, Backtracking und Rekursion. Bei der Unifizierung wird versucht, Gemeinsamkeiten in zwei verschiedenen PROLOG-Strukturen (gemeinsame Variablen oder Konstanten) zu erkennen und hieraus neue Schlüsse zu ziehen. Führt ein Weg in einer Argumentationskette zu einem unerwünschten Ziel, weil zum Beispiel von falschen Voraussetzungen ausgegangen worden ist, so wird das PROLOG-Programm um einen Schritt in seiner Entscheidungskette zurückgehen und einen anderen Weg der Lösungsfindung ausprobieren. Dieses Verfahren wird in der Fachsprache Backtracking genannt und findet vor allen Dingen bei der Auswertung strategischen Wissens Anwendung. Die Rekursion ist eine Möglichkeit, in PROLOG Schleifen zu programmieren, so daß ein Programmteil mehrfach abgearbeitet werden kann. PROLOG ist heute in Europa und Japan die für Probleme der künstlichen Intelligenz und Expertensysteme bevorzugte Sprache.

[15] *Ludewig:* Sprachen für die Programmierung, BI Bd. 622, S. 157 ff; s. auch Fn(7) S. 97 f.

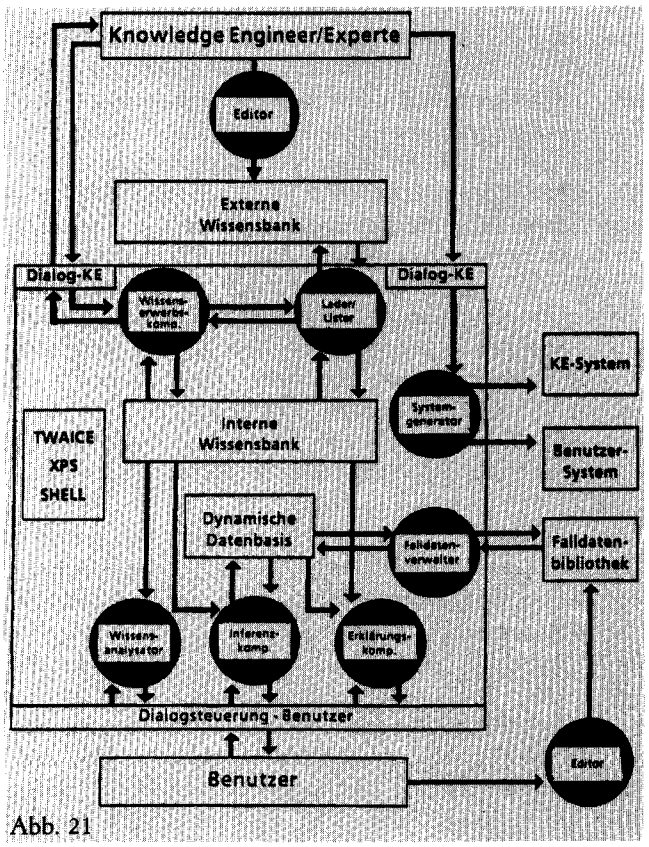

Abb. 21

Die Implementierung von TWAICE und PROLOG erfordert wegen der komplexen Zusammenhänge einen sehr hohen Aufwand an Rechnerleistung. Die zu Beginn der Rechnerentwicklung üblichen „Batch"-Betriebssysteme sind hierfür nicht geeignet, da sie Programme und Programmanweisungen nur in sequentieller Reihenfolge abarbeiten können und damit dem nicht-prozeduralen Charakter von PROLOG entgegenstehen. Daher wurde TWAICE auf das Betriebssystem UNIX zugeschnitten. UNIX ist ein sogenanntes Multitasking-Betriebssystem für den wissenschaftlich-technischen und zunehmend auch für den kommerziellen Bereich; das heißt, es führt mehrere Aufgaben quasi zur gleichen Zeit aus. Ein gegebenes Programm wird in mehrere Teilaufgaben („Tasks") aufgeteilt; jede dieser Tasks wird anschließend abwechselnd vom Betriebssystem für eine bestimmte Zeitdauer (z. B. 10 msec) bearbeitet. Danach schaltet das Betriebssystem auf die nächste Task um und bearbeitet diese. Sind auf diese Art und Weise alle Tasks einmal bearbeitet worden, setzt der Prozessor seine Arbeit bei der ersten Task an genau der Stelle wieder fort, an der er sie zuvor verlassen hat. Dieses Verfahren der Bearbeitung mehrerer Aufgaben wird in der Computertechnik als Job Scheduling oder Dispatching bezeichnet. Da diese gesamte Arbeit nur von einem einzigen Mikroprozessor geleistet wird, werden solche Systeme auch als „Single-Prozessor-Systeme" bezeichnet.

Seit einiger Zeit ist man auf der Suche nach Möglichkeiten, komplexe PROLOG-Programme noch schneller als bisher abarbeiten zu können. Das soll durch den parallelen Einsatz mehrerer Prozessoren innerhalb eines einzigen Rechners erreicht werden. Aus diesem Grunde wurde unter der Bezeichnung „Entwurf, Entwicklung und Erprobung eines kostenoptimierten PROLOG-Parallelrechners (PIPE)" ein Projekt ins Leben gerufen, an dem neben NIXDORF AG noch die Firma Epsilon GmbH sowie die Universitäten Berlin und Kaiserslautern beteiligt sind. Ziel dieses Projektes ist es, auf der Basis der Rechnertypen NIXDORF 8831 und NIXDORF 8832 neuartige Rechnerstrukturen zu entwickeln, deren Hardware einerseits den gestiegenen Anforderungen der Programme entspricht, die andererseits aber noch mit vertretbaren Kosten herzustellen sind. Man verspricht sich von den Forschungsarbeiten einen Rechner, der im Vergleich zu bisher verfügbaren mindestens 30mal schneller arbeiten soll. Die experimentelle Phase dieser Entwicklungen soll noch in diesem Jahr (1987) beginnen, da ein Umschreiben der vorhandenen Software notwendig wird.

In die gleiche Richtung zielt ein Forschungsvorhaben am japanischen Forschungsinstitut für Rechner der neuen Generation (ICOT). Dort ist man mit der Entwicklung eines PROLOG-Parallelrechners mit der Bezeichnung PIE (Parallel Inference Engine) beschäftigt.

Als Betriebssystem für TWAICE wurde — wie bereits früher erwähnt — UNIX ausgesucht. Die passende Rechnerfamilie hierfür bietet NIXDORF unter der Bezeichnung „Targon" an. Es handelt sich hierbei um einige 32-Bit-Maschinen, die in der Größenordnung der Mini-Computer bis zum Micro-Computer (nicht zu verwechseln mit Home- oder Personal Computern!) anzusiedeln sind. Die Targon-Rechner arbeiten im Bänder- oder Kassettenbetrieb; der Benutzer hat somit die Möglichkeit, wichtige Daten außerhalb des Rechners zu sichern. Eine TWAICE-Version für IBM Mainframe-Computer unter dem Betriebssystem VM/SP wurde bereits in Aussicht gestellt. NIXDORF weist darauf hin, daß TWAICE nicht auf den Rechnern der Serien 8860 und 8870 lauffähig ist.

Einer der Gründe, weshalb TWAICE nicht auf den weitverbreiteten Personal Computern lauffähig ist, ist darin zu sehen, daß eine typische Anwendung dieses Programms einen Speicherbedarf von 2...2,5 MB hat, wohingegen die Personal Computer selten mit einem Speicherangebot von mehr als 1 MB aufwarten können. Der NIBEX-Melder hat sogar einen Speicherbedarf von 4...4,5 MB! Ein weiterer Grund liegt darin, daß die Personal Computer nicht den geforderten Datendurchsatz erreichen können, da sie meist nur mit 16-Bit-Prozessoren ausgestattet sind. Es ist jedoch abzusehen, daß bei der derzeitigen Entwicklung der Halbleitertechnologie und -preise und nach dem Erscheinen der ersten 32-Bit Personal Computer TWAICE wohl auch bald für Rechner dieser Größenordnung erhältlich sein wird.

II. Aufbau von TWAICE

Anhand des Bildes 21 soll der innere Aufbau von TWAICE beschrieben werden. Die Shell läßt sich in zwei Bereiche unterteilen; der erste Teil ist nur dem Experten und dem Knowledge Engineer zugänglich[16] und beinhaltet die externe Wissensbank, die Wissenserwerbskomponente und den Lader/Lister. Der dem Benutzer zugängliche Teil enthält den Wissensanalysator, die Inferenzkomponente sowie die Erklärungskomponente. Beide Teile besitzen je einen Editor, mit dessen Hilfe die Eingabe der benötigten Daten erfolgt.

1. Dialogsteuerung

Die Dialogsteuerung bildet die Schnittstelle zwischen dem Expertensystem und dem Benutzer bzw. Knowledge Engineer. Sie bereitet die von der Erklärungskomponente, dem Wissensanalysator und der Inferenzkomponente übergebene Daten für die Ausgabe zum Benutzer vor. Die Dialogkomponente übernimmt weiterhin die Aufgabe der Benutzerführung; d. h., daß dem Benutzer über geeignete Menüs angezeigt wird, welche Systemfunktionen er (Konsultation, Wissensanalyse ...) gerade anwählen kann. Dabei ist sie in der

[16] Die Zugriffsrechte auf die einzelnen Teile werden durch die Vergabe von Paßwörtern geregelt.

Lage, mit Hilfe vorgegebener Texte natürlichsprachliche Sätze zu bilden. Andererseits ist es dem Benutzer auch möglich, über die Dialogsteuerung dem System neue Daten zur Verfügung zu stellen, wenn diese benötigt werden. Hierbei ist es dem Benutzer erlaubt, seine Sätze in einer natürlichsprachlichen Form einzugeben. Er muß nicht mehr — wie dies bei anderen Programmen oft der Fall ist — aus einer vorgegebenen Liste eine mehr oder weniger passende Antwort aussuchen, sondern kann Daten genau so eingeben, wie es ihm gerade in den Sinn kommt. Falls dem Benutzer dabei Tippfehler[17] unterlaufen sollten, versucht die Dialogsteuerung diese Fehler zu korrigieren; ist die Korrektur nicht eindeutig möglich, bitte die Dialogsteuerung den Benutzer um weitere Informationen. Der Benutzer ist weiterhin zu jedem Zeitpunkt der Konsultation in der Lage, eine Liste der erlaubten Kommandos mit dem Befehl „?" anzufordern. Auf Wunsch schreibt die Dialogsteuerung den gesamten Dialog in eine sogenannte Logdatei, damit der Benutzer sich nach Beendigung einer Sitzung noch einmal mit dem gefundenen Ergebnis befassen kann.

2. Interne Wissensbank/Wissensanalysator

Als zentraler Kern der Shell ist die interne Wissensbank erkennbar; sie wird aus der externen Wissensbank derart erstellt, daß die im Klartext eingegebenen Regeln über das Expertenwissen hier in Form eines ablauffähigen PROLOG-Programms abgelegt werden. Das Übersetzen von der externen in die interne Darstellungsform übernimmt der Lader/Lister. Während der Übersetzung erfolgt außerdem eine Überprüfung der Regeln auf syntaktische und semantische Korrektheit. Syntaktische Fehler, d.h. Fehler in der Rechtschreibung, versucht der Lader/Lister soweit wie möglich selbständig zu korrigieren. Ist eine Korrektur nicht möglich, so wird in die Wissenserwerbskomponente verzweigt. Im Dialog mit ihr kann der Experte oder Knowledge Engineer neue Begriffe in das Taxonomiewissen aufnehmen, oder aber auch den fehlerhaften Begriff zurückweisen. Im Falle der Zurückweisung wird die zugehörige Regel als fehlerhaft gekennzeichnet und kann nach Beendigung des Ladevorganges aufgelistet und korrigiert werden. Semantische Fehler, also Fehler in der Logik der Regeln, werden von TWAICE zur Zeit noch nicht erkannt. Es ist allerdings denkbar, daß in künftigen Versionen Schleifen oder Widersprüche innerhalb der Regeln erkannt werden.

Falls Modifikationen innerhalb der internen Wissensbank vorgenommen wurden, wird dieses dem Knowledge Engineer nach Beendigung einer Sitzung mitgeteilt, so daß dieser die Möglichkeit hat, die neuen Daten auf externen Datenträgern zu sichern.

3. Inferenzkomponente

Die Inferenzkomponente steuert bei geladener Wissensbank die Durchführung der Konsultation und den Inferenzprozeß. Bei TWAICE wird als wichtigste Inferenzmethode die Rückwärtsverkettung angewandt. Bei dieser Methode geht die Shell von einem gegebenen Ziel aus und versucht anhand der vorgegebenen Regeln einen Weg zu diesem Ziel zu finden. Häufig werden zu diesem Zweck von der Shell Zwischenziele aus den vom Knowledge Engineer vorgegebenen Zielattributen gebildet, die dann einzeln verifiziert werden. Fehlen dem System Informationen in seiner Argumentationskette, so werden diese über die Dialogsteuerung vom Benutzer angefordert. Die Shell ist hierbei in der Lage, klare Fragen zu formulieren und eine Begründung für die Datenanforderung abzugeben. Wird die Inferenzkomponente vom Experten oder Knowledge Engineer benutzt (dies ist am eingegebenen Paßwort erkennbar), so kann sie auf Wunsch einen Trace über den Ablauf des Inferenzprozesses ausgeben. Ein solcher Trace beinhaltet alle Informationen, die notwendig sind, um den Inferenzprozeß nachvollziehen zu können. Dazu gehören: Angabe der Zwischenziele, eine Liste der benutzten Regeln, Kennzeichnung der Benutzerantworten sowie eine Erklärung über den Herleitungskontext.

4. Erklärungskomponente

Ihre Aufgabe ist es, Fragen, die von der Inferenzkomponente gestellt werden, zu erklären (warum-Erklärung) und die gefolgerten Fakten zu rechtfertigen (wie-Erklärung).

5. Falldatenverwaltung

Die Falldatenverwaltung übernimmt die Organisation der Falldatenbibliothek. Sie fügt auf Anweisung des Benutzers neue Testfälle in die Bibliothek ein oder löscht bereits vorhandene, die nicht länger benötigt werden.

6. Systemgenerator

Mit Hilfe des Systemgenerators ist es möglich, das gesamte bisher erstellte System auf einen externen Datenträger zu retten. Hierzu werden die Expertensystem-Shell TWAICE und die interne Wissensbank zu einer einzigen PROLOG-Objektdatei zusammengefaßt. Der Systemgenerator erstellt also ein Expertensystem für eine ganz spezielle Wissensdomäne. Darüberhinaus ist es möglich, verschiedene Teile des Systems durch Paßwörter zu schützen, um so einer unerlaubten Änderung innerhalb der Wissensbank vorzubeugen.

[17] Hierzu zählen: Buchstabenvertauschungen, fehlende Buchstaben oder fehlende Endungen; a. Fn(7) S. 67.