Hypermedia als Zwischenrepräsentation bei der Expertensystementwicklung[1]

Frank Maurer & Gerd Pews
Universität Kaiserslautern
AG Expertensysteme Prof. Richter
Postfach 3049
D-6750 Kaiserslautern
e-Mail: {maurer, pews}@informatik.uni-kl.de

In diesem Papier vergleichen wir Hypermedia- und Expertensystemansätze zur Wissensverarbeitung. Wir zeigen, wie ein integrierter Ansatz die Erstellung von Expertensystemen erleichtert. Das von uns entwickelte und implementierte System ermöglicht einen "sanften" Entwicklungsprozeß ausgehend von initialen Protokollen zu einer semi-formale Strukturierung in Form eines getypten Hypertextes. Dem Hypertext ist eine aufgabenorientierte Struktur aufgeprägt, so daß eine anschließende Operationalisierung in Form eines Expertensystems vereinfacht wird. Die in diesem Prozeß erzeugte Zwischenrepräsentation (der Hypertext) wird von einem Interpreter direkt zur interaktiven Lösung von Problemen benutzt, wobei die einzelnen Aufgaben auf die verschiedenen Sachbearbeiter verteilt werden. Abschließend erläutern wir, daß Hypertext und Expertensysteme nur die Ränder eines Kontinuums einer allgemeinen Wissensverarbeitung sind.

1.0 Einleitung und Überblick

Die Integration von Hypermedia- und Expertensystemtechniken erscheint vielen Autoren erfolgversprechend ([2], [3], [11], [14]). Beide Techniken erlauben die Verwaltung von und den effektiven Zugriff auf Wissen mit Hilfe des Rechners.

Atomare Wissenseinheiten (Knoten) in Hypermediasystemen werden typischerweise in einer vom Rechner nicht interpretierbaren Form (z. B. Videosequenzen, Bilder, natürlichsprachlicher Text) gespeichert. Das Wissen kann vom Benutzer kontextabhängig, d.h. mit Hilfe seines "gesunden Menschenverstandes", interpretiert und so zur Lösung eines Problems herangezogen werden. Zusätzlich sind an die einzelnen Knoten Verweise auf weitere Knoten gekettet, denen der Benutzer folgen kann um weitere Informationen zu erhalten. Die Suche nach dem für die Lösung des aktuellen Problems benötigten Wissen wird in erster Linie vom Benutzer gesteuert[2].

---

1. Diese Arbeit wurde teilweise unterstützt vom Ministerium für Wirtschaft und Verkehr des Landes Rheinland-Pfalz im Rahmen des Projektes "Integration von Hypermedia und Expertensystemen"

Expertensysteme hingegen basieren auf einer Formalisierung des Wissens, die von einem Interpreter dann verarbeitet wird. D.h. der Interpreter benutzt das gespeicherte Wissen, um den Benutzer bei der Lösung des Problems anzuleiten. In diesem Sinne liegt die Initiative bei Expertensystemen eher bei der Maschine.

Beide Ansätze haben ihre Stärken und Schwächen:

- Expertensysteme stellen *weniger Anforderungen an den Benutzer*, da dieser sehr stark durch das System bei der Problemlösung angeleitet wird. Das “lost-in-hyperspace”-Problem kommt aus diesem Grund auch weniger zum Tragen, da der Benutzer seinen Weg im Informationsraum nicht selbst sucht und von daher auch nicht verlieren kann.
- Expertensysteme ermöglichen die *weitgehende Automatisierung* der Bearbeitung wissensintensiver Probleme, da Inferenzen durch ein Programm, den Interpreter, gezogen werden.
- Expertensysteme verlangen eine *Dekontextualisierung von Wissen*, da Inferenzen nur aufgrund des formalisierten Wissens gezogen werden können und dabei die Einbettung in den natürlichen Kontext verloren geht.
- Hypermediasysteme erfordern *weniger Aufwand bei der Entwicklung*, da Wissen nicht formalisiert werden muß: Die Interpretation der Knoteninhalte wird vom Benutzer vorgenommen, der dazu sein gesamtes Wissen über die Domäne benutzen kann. D.h. über den Benutzer kann auf den natürlichen Kontext, die Welt, zurückgegriffen werden.
- Die *Kommunikation* mit dem Benutzer wird durch die multimediale Schnittstelle von Hypertextsystemen stark verbessert.

Eine Frage, die sich stellt, ist: Wie kann man die Ansätze miteinander integrieren, so daß die jeweiligen Vorteile übernommen und die entsprechenden Nachteile zurückgedrängt werden? In [13] haben wir uns mit der Integration der beiden Ansätze auseinandergesetzt. Die für dieses Papier relevanten Aspekte fassen wir zusammen und beleuchten sie aus der Sicht der Hypermediasysteme. Im zweiten Kapitel beschreiben wir, wie wir das Wissen über eine neue Anwendung in Form eines Hypertextes strukturieren. Die resultierende Zwischenrepräsentation bildet die Basis des im dritten Kapitel vorgestellten Interpreters, der die verteilte Problemlösung ermöglicht. Im vierten Kapitel erläutern wir dann, warum Hypertexte und Expertensysteme nur die Ränder eines Spektrums von wissensverarbeitenden Systemen sind. Im fünften Kapitel stellen wir kurz die KADS-Methodik vor, die ein Ausgangspunkt unserer Arbeit war, und vergleichen unseren Ansatz damit. Das letzte Kapitel beschreibt den Stand der Realisierung und gibt einen Überblick über weitere Arbeiten.

---

2. “guided tours” verlagern die Initiative stärker auf das System und sind von daher die Ausnahme zu obiger Beschreibung.

2.0 Strukturierung von Wissen

Die Strukturierung von Wissen, mit dem Ziel einer Operationalisierung, ist seit langen Jahren Gegenstand der Forschungen im Knowledge Engineering. Die dabei gewonnenen Erkenntnisse übertragen wir auf die Strukturierung von Hypertexten[3]:

Ausgehend von initialen Daten (wie z.B. Mitschriften von Expertenbefragungen, Schaubildern über die Domäne, etc.) erstellen wir ein Hypermedianetzwerk als Zwischenrepräsentation, dessen Knoten und Kanten typisiert sind. Die vom System bereitgestellten Typen werden in Anlehnung an Ergebnisse der Wissensakquisitions-Forschung definiert, um dadurch einen guten Anknüpfungspunkt für eine spätere Formalisierung des Wissens zu erhalten. Die Hypertext-Zwischenrepräsentation ist die Eingabe eines interaktiven Interpreters, der eine Gruppe von Anwendern bei der Problemlösung unterstützt. Einzelne Unteraufgaben werden über ein lokales Netz (LAN) den jeweiligen "Sachbearbeitern"[4] zugewiesen, d.h. wir unterstützen eine verteilte Problemlösung.

Zur Unterstützung der Wissensingenieure bei der Strukturierung und Formalisierung einer Domäne entwickeln wir das System CoMo-Kit[5], das in Abschnitt 2.1 vorgestellt wird. Abschnitt 2.2 erläutert, wie wir mit CoMo-Kit ausgehend von initialen Daten die Zwischenrepräsentation entwickeln.

2.1 CoMo-Kit: Conceptual Model Construction Kit

Die Erstellung von Expertensystemen der 1. Generation erfolgte im wesentlichen nach dem Prototyp-Ansatz. Die dabei zutage tretenden Schwächen führten zur Entwicklung von modellbasierten Methodiken. Eine in Europa vorrangige ist KADS (vgl. Abschnitt 5.0). In KADS wird das Ergebnis der Analyse einer Domäne *konzeptuelles Modell* genannt.

CoMo-Kit basiert auf dem HyperCAKE-System [12] und Ideen von [14]. HyperCAKE nutzt eine erweiterte Hypertext Abstract Machine [7] für die Verwaltung multimedialer Informationen. HyperCAKE[6] ermöglicht die Definition von anwendungsspezifischen Sichten auf einen globalen Hypertext. Hypermedianetze sind in einer globalen Datenbank gespeichert und von allen Workstations in einem lokalen Netz zugänglich.

---

3. Die hier vorgestellte Arbeit beruht auf [13]. Wir fassen hier die relevanten Ergebnisse zusammen und diskutieren sie aus dem Blickwinkel der Hypertext/Hypermedia-Systeme.
4. "Sachbearbeiter" sind dabei entweder Menschen oder Programme.
5. CoMo-Kit wurde in Zusammenarbeit mit Susanne Neubert, Uni Karlsruhe, entwickelt, die eine Beispielapplikation (Zusammenstellen von Versicherungspaketen) zur Verfügung stellte.
6. HyperCAKE: Hypermedia-Based Computer Aided Knowledge Engineering

CoMo-Kit nutzt das HyperCAKE-System zur Verwaltung *aller* im Verlauf des Knowledge Engineering Prozesses anfallenden Daten. Dazu wurden folgende Knotenklassen definiert (wir beschränken unsere Darstellung auf im folgenden relevante Klassen):

- **Protokoll:** Ein Protokoll enthält initiale, unstrukturierte Daten, die vom Experten oder aus sonstigen Wissenquellen erhoben wurden.
- **Konzept:** Konzepte beschreiben für die Lösung des Problems notwendige Informationseinheiten in textueller, natürlichsprachlicher Form. Wir unterscheiden, wie beim objektorientierten Design üblich, zwischen Klassenbeschreibungen und Instanzen. Konzeptklassen werden in eine IS-A-Hierarchie eingeordnet[7].
- **Aufgabe (Task):** Eine Task beschreibt (in textueller, natürlichsprachlicher Form) eine Aufgabe, die durchgeführt werden muß, um ein gegebenes Problem zu lösen. Jede Task kann aus mehreren Unteraufgaben bestehen, die dann in Form eines Datenfluß-Diagramms beschrieben werden. Aufgaben sind also hierarchisch organisiert. Für jede Aufgabe werden ihre Ein- und Ausgaben definiert.
- **Bearbeiter (Agent[8]):** Agenten werden eindeutig über ihren Namen angesprochen. Jeder Agent kann zu verschiedenen Gruppen gehören. In der Spezifikation wird für jede Aufgabe festgelegt, welcher Agent oder welche Gruppe von Agenten sie potentiell bearbeiten kann.

Da der im Entwicklungsprozeß entstehende Hypertext sehr umfangreich ist, wodurch die Gefahr des “lost-in-hyperspace” wächst, werden in CoMo-Kit verschiedene Sichten auf das Netzwerk definiert[9]:

- **Protokoll-Kontext:** Diese Sicht umfaßt alle Protokolle für ein Projekt.
- **Konzept-Kontext:** Diese Sicht zeigt alle Konzepte und Relationen zwischen Konzepten[10].
- **Aufgaben-Kontext:** Der Aufgaben-Kontext zeigt alle definierten Tasks in einer Verfeinerungshierarchie.
- **Aufgabenstruktur-Kontext:** Diese Sicht zeigt die innere Struktur, d.h. den Datenfluß, einer Aufgabe.

---

7. Die durch die IS-A-Relation gebildete Hierarchie wird durch den später beschriebenen Interpreter nicht direkt ausgenutzt. Dieser arbeitet mit Instanzen der einzelnen Klassen. Die Hierarchie dient nur der Strukturierung der Begriffswelt der Domäne.
8. Der Begriff des Agenten wird bei uns nicht in dem umfassenden Sinn wie in der verteilten KI benutzt, sondern bezeichnet nur den Bearbeiter einer Aufgabe.
9. Technisch unterscheiden wir zwischen statischen Sichten, den *Contexts* der HAM, und dynamischen Sichten, die durch eine Bedingung an die in ihnen enthaltenen Objekte spezifiziert werden. Das Benutzerinterface für beide Arten ist identisch. Deshalb sprechen wir im folgenden nur noch von Kontexten.
10. CoMo-Kit unterstützt weitere Relationen zwischen Konzepten, wie PART-Of, CAUSES etc. Wir gehen auf diese nicht ein, da sie außerhalb des Fokus dieses Papiers liegen.

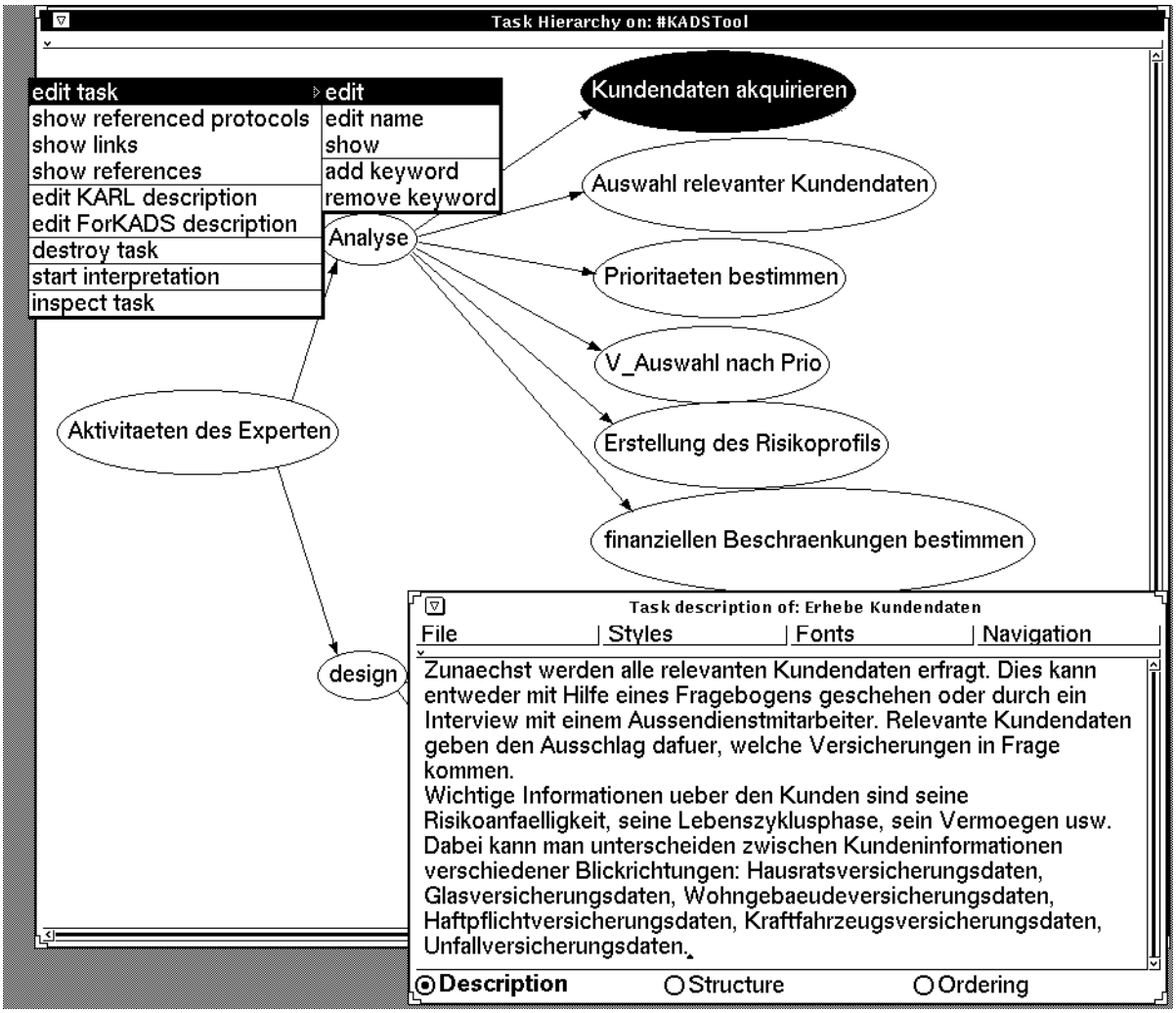

Abbildung 1: Die Aufgaben-Hierarchie und eine Aufgabenbeschreibung

Für jeden Kontext ist ein graphisches Interface implementiert, daß dem Benutzer ermöglicht, den Kontext zu manipulieren. Desweiteren kann der Benutzer jederzeit durch Angabe von Knoten- und Kantenbedingungen eine spezielle Sicht auf das Netz spezifizieren, die ihm dann über ein graphisches Interface präsentiert wird.

2.2 Benutzung von CoMo-Kit

Ausgangspunkt der konzeptuellen Modellierung ist das Protokoll eines Gesprächs mit einem Experten. Dieser soll die zu lösende Aufgabe umgangssprachlich beschreiben. In einem Protokoll kann der Wissensingenieur[11] einen Teil des Textes selektieren, der eine zu bearbeitende Teilaufgabe beschreibt. Anschließend wählt er in einem Menü den Eintrag “create task”, um einen entsprechenden Aufgabenknoten zu erzeugen. Analog kön-

11. In einer Beispielanwendung (Baunutzungsverordnung) von CoMo-Kit wird die Strukturierung der Domäne direkt von Raumplanern, d.h. den Experten, vorgenommen. Wir denken, daß sich das auch auf andere Domänen übertragen läßt, da keine Programmierung im eigentlichen Sinne durchgeführt werden muß.

nen Konzepte und Agenten erzeugt werden. Die Zuordnung eines Bearbeiters zu einer Gruppe und zu Aufgaben erfolgt über eine graphische Schnittstelle, die hier nicht gezeigt wird. Die Terminologie einer Anwendung, d. h. die relevanten Konzeptklassen, kann mit Hilfe eines speziellen Editors strukturiert werden. Dieser ermöglicht über eine graphische Schnittstelle den Aufbau von IS-A- und PART-OF-Hierarchien. Attribute einer Klasse werden typisiert. Aus der so erstellten Spezifikation der Konzepte kann eine maskenorientierte Benutzerschnittstelle generiert werden, die nur die Eingabe von Werten aus dem Wertebereich erlaubt.

Abbildung 1 zeigt eine Aufgabenstruktur. Desweiteren ist die Beschreibung einer Aufgabe zu sehen. Der in einem Protokoll angewählte Text wurde in den neu erzeugten Knoten kopiert. Der Wissensingenieur kann die Beschreibung dann editieren, um die Aufgabe genauer zu spezifizieren.

Ein Datenfluß-Diagramm stellt die innere Struktur einer Aufgabe dar (vgl. Abbildung 2). Konzepte werden als Rechteck gezeigt, wohingegen Ellipsen Aufgaben repräsentieren. Die Hierarchie der Datenfluß-Diagramme eines zu lösenden Problems ist die Basis des im folgenden Kapitel beschriebenen Interpreters.

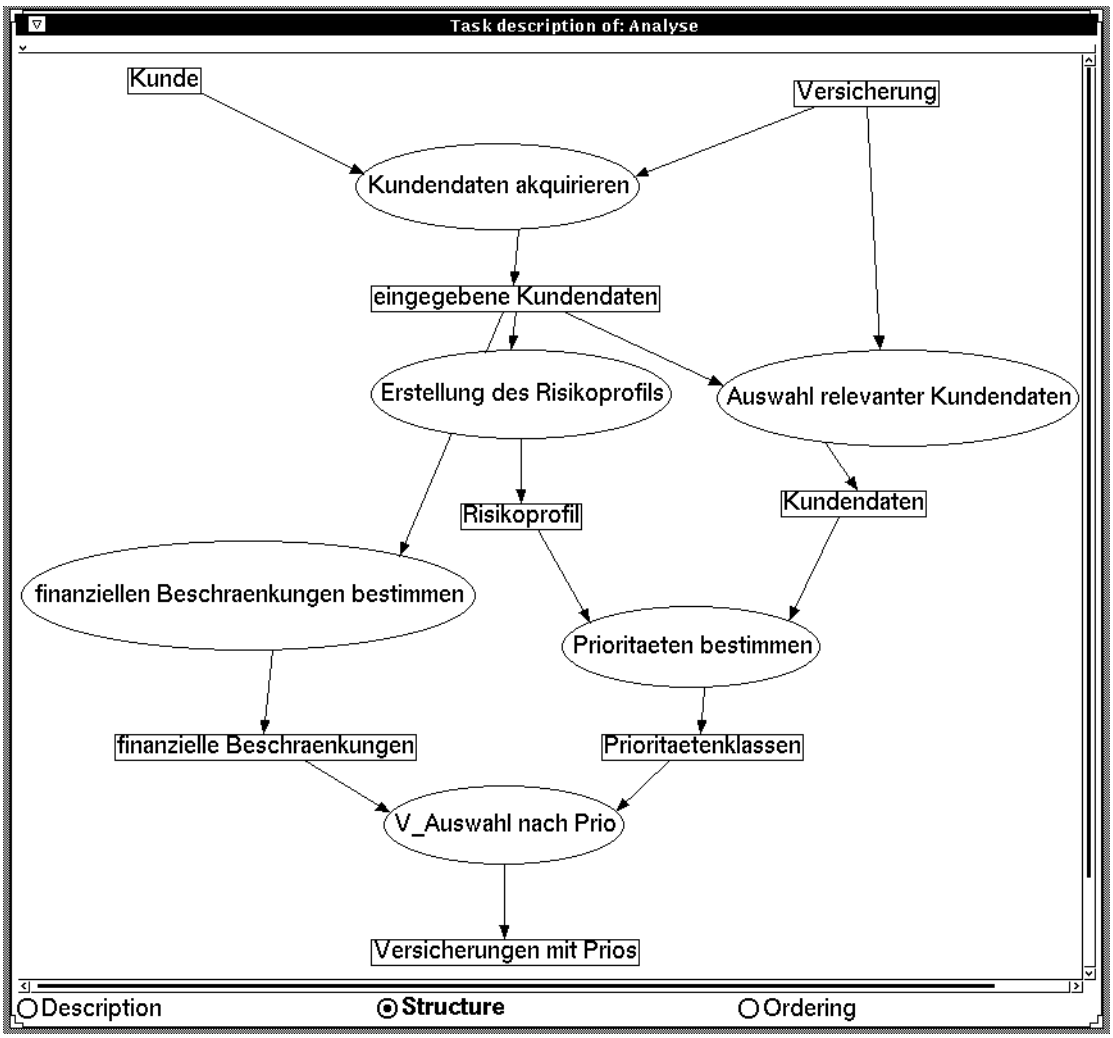

Abbildung 2: Eine Aufgabenstruktur

Jeder Aufgabe kann eine formale, ausführbare Annotation beigefügt werden. D. h. man ersetzt die natürlichsprachliche Beschreibung durch Programmcode. Daran interessierte Leser möchten wir auf [13] verweisen.

3.0 Der Interpreter für Aufgabenstrukturen

CoMo-Kit umfaßt einen Interpreter, der eine semi-formale[12] Aufgabenstruktur in Interaktion mit den Benutzern verarbeiten kann. D.h. schon die Strukturierung der Domäne in Form eines Hypertextes führt zu einem verwertbaren Ergebnis: Komplexe Aufgaben werden in kleinere Teile zerlegt, die dann von (weniger qualifizierten) Sachbearbeitern[13] bearbeitet werden können. Diese haben nur Zugriff auf für die Aufgabe relevante Information; die anderen werden vom System weggefiltert.

Ein Manager initiiert die verteilte Bearbeitung einer Aufgabe (Task), indem er sie an einen oder mehrere andere Benutzer delegiert und zur Ausführung freigibt. Er startet dadurch einen Scheduler-Prozeß. Dieser sorgt dafür, daß nur Aufgaben bearbeitet werden, für die zum jeweiligen Zeitpunkt auch alle benötigten Eingaben vorhanden sind. Ein Benutzer kann dann aus allen von ihn bearbeitbaren Tasks diejenige auswählen, die er als nächstes bearbeiten möchte. Die Aufgaben lassen sich in zwei Gruppen einteilen: komplexe und atomare.

Komplexe Aufgaben bestehen aus mehreren Unteraufgaben. Sie sind die inneren Knoten des in Abbildung 1 gezeigten Baumes. Wenn ein Benutzer eine komplexe Aufgabe bearbeiten will, übernimmt er die Rolle eines Managers. Ein Manager muß Aufgaben auf Sachbearbeiter verteilen und die Durchführung der Aufgaben überwachen.

Dabei wird er vom Rechner unterstützt. Dieser leitet Aufgaben, die nur einen möglichen Bearbeiter haben, direkt an diesen weiter. Desweiteren kann der Manager interaktiv Aufgaben delegieren und den Stand der Bearbeitung überprüfen[14]. Abbildung 3 zeigt ein Verwaltungsfenster. Das Fenster enthält vier Listen mit:

- Aufgaben, die noch unbearbeitet sind (links oben),
- Aufgaben, die schon an einen oder mehrere Benutzer delegiert sind, von diesen aber noch nicht bearbeitet wurden (links unten),
- Aufgaben, die gerade bearbeitet werden (rechts unten),
- Aufgaben, die schon bearbeitet sind (rechts oben)

---

12. Vgl. Abschnitt 4.0, "Informale, semi-formale und formale Wissensrepräsentation"
13. Durch die Strukturierung des Wissens findet indirekt eine *Qualifikation* der Sachbearbeiter statt, die nun neue Aufgabengebiete bearbeiten können. Desweiteren werden “echte Experten” von Routinetätigkeiten entlastet und können sich somit intensiv schwierigen Problemen widmen.
14. Im Moment erweitern wir, auf Anregung eines ungenannten Gutachters, den Interpreter im Sinne der Vorgangsbearbeitung um Möglichkeiten der Terminverfolgung.

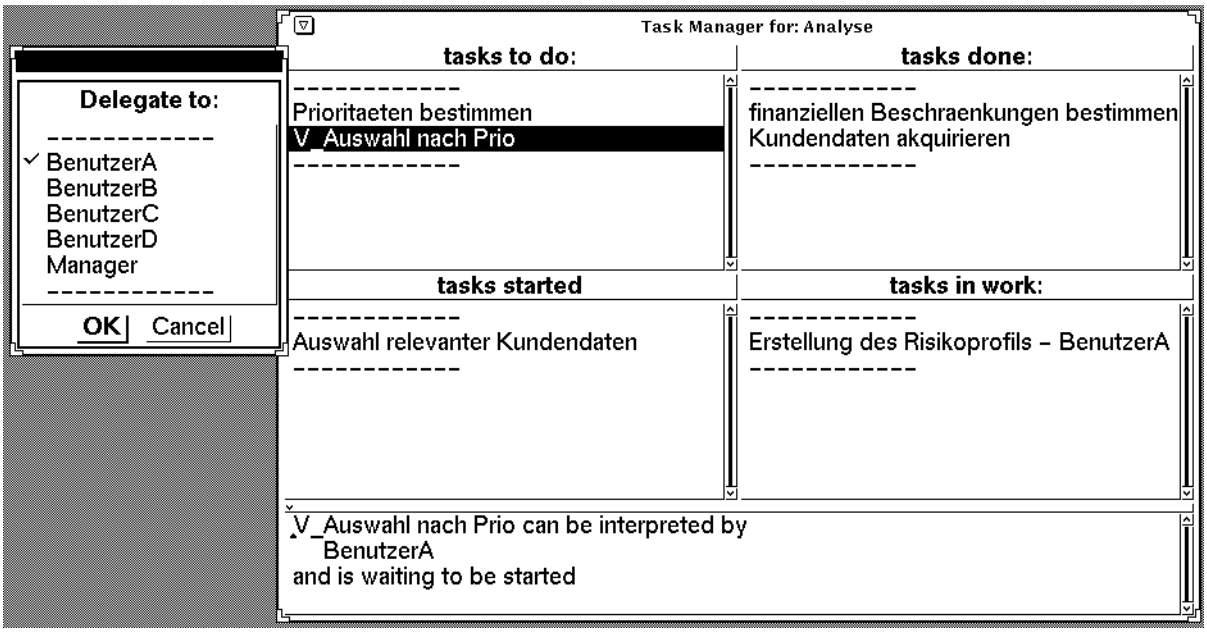

Abbildung 3: Verteilung der Aufgaben auf einzelne Agenten

Unter den Listen befindet sich ein Textfeld, in dem Informationen über den Stand der Bearbeitung jeder Aufgabe dargestellt werden. In der Überschrift ist die auszuführende Aufgabe zu sehen (hier: Analyse). Links oben werden alle noch zu verteilenden Aufgaben gezeigt. Diese können mit Hilfe eines Menüs an einen der möglichen Sachbearbeiter delegiert werden (hier: Benutzer A - Benutzer D, Manager). Sobald dies geschehen ist, erscheint die Aufgabe in der Liste links unten. Das Fenster rechts unten wiederum enthält eine Liste der Tasks, die gerade von dem Sachbearbeiter ausgeführt werden, dessen Namen hinter der Aufgabe gezeigt wird (hier: Benutzer A). Die Liste rechts oben umfaßt alle Aufgaben, die bereits abgeschlossen sind. D.h. im Verlauf der Zeit wandern die einzelnen Aufgaben (entgegen dem Uhrzeigersinn) von links oben nach rechts oben. Der Manager kann den Bearbeitungsvorgang verfolgen, bei Stockungen nach der Ursache forschen und evtl. eingreifen.

Atomare Aufgaben werden mit Hilfe des in Abbildung 4 gezeigten Fensters bearbeitet Der Editor enthält auf der linken Seite dynamisch erzeugte Buttons, die den Zugriff auf für die Aufgabe relevante[15] Informationen ermöglichen. In der Mitte findet der Benutzer eine Beschreibung der Aufgabe. Rechts befindet sich der Editor für die Eingabe des Resultats[16]. Sobald der Propagate-Button gedrückt wird, werden die Ergebnisse an den Scheduler propagiert. Anschließend können in dem Datenfluß-Diagramm nachfolgende Aufgaben von den dafür zuständigen Benutzern durchgeführt werden.

15. Was für eine Aufgabe relevant ist, wird bei der Strukturierung der Domäne festgelegt: Die Eingaben einer Aufgabe sind relevant für deren Lösung.

16. Falls die Aufgabe mehrere Resultate erzeugen soll, sind mehrere Editoren zu sehen. Sind die Ausgaben strukturiert, dann wird kein Texteditor (wie in der Abbildung oben) sondern eine entsprechende Maske gezeigt werden. Die Struktur der Daten muß nicht vorab definiert werden, sondern kann inkrementell entwickelt werden. Jedes Zwischenstadium kann in der Problemlösung eingesetzt werden, wodurch ein sanfter Übergang erreicht wird.

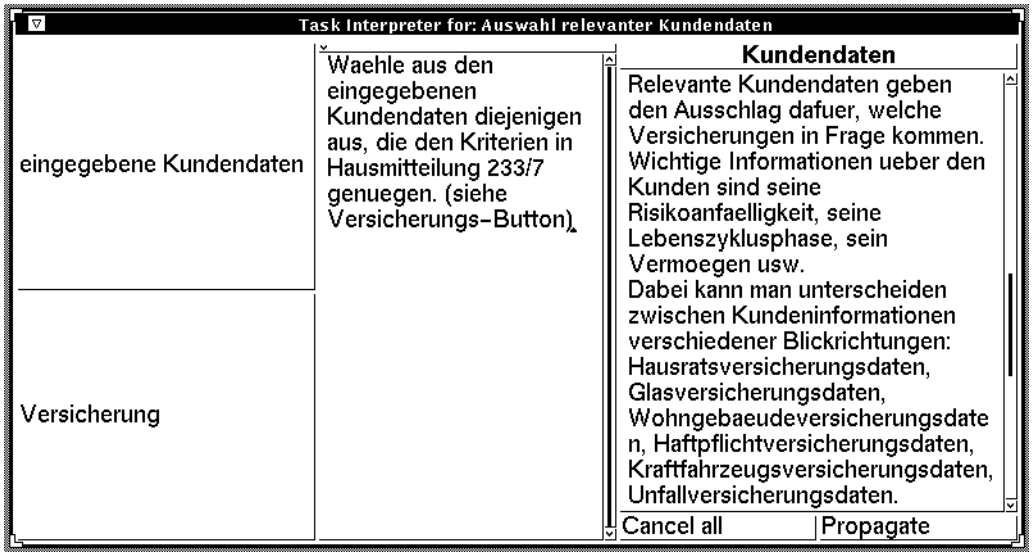

Abbildung 4: Editor für die Bearbeitung einer atomaren Aufgabe durch den Benutzer

Der beschriebene Interpreter hat im wesentlichen zwei interessante Eigenschaften:

1. Das Wissen muß nicht vollständig formalisiert werden, sondern liegt in Form eines Hypermedia-Netzwerks vor. Der Interpreter argumentiert nur über die Struktur des Netzes (d.h. Knoten- und Kantentypen haben eine operationale Semantik), nicht aber über die Knoteninhalte, deren Interpretation dem Benutzer überlassen bleibt.
2. Die Lösung des Problems wird auf mehrere Agenten, d.h. Sachbearbeiter oder Rechner, verteilt. Von daher bestehen Querbezüge zum Computer Supported Cooperative Work (CSCW) und zur verteilten KI, die wir aber hier nicht ausarbeiten können.

4.0 Informale, semi-formale und formale Wissensrepräsentation

Ausgangspunkt der Entwicklung mit Hilfe von CoMo-Kit sind textuelle Beschreibungen von Aufgaben. Diese sind *informal* im Sinne von “nicht durch einen maschinellen Interpreter verarbeitbar”. Unter *formalisiertem Wissen* verstehen wir hingegen, eine Menge von Informationen, die von einem Rechner zur selbständigen Lösung von Problemen herangezogen werden können, ohne daß ein Mensch involviert ist. Das damit beschriebene Modell ist formal: Die Syntax *und* die Semantik der Sprache kann im Sinne formal-logischer Kalküle und Funktionen festgelegt werden. D. h. ein Programm ist im Prinzip in der Lage, Eingabedaten durch eine Menge von syntaktischen Umformungen in die gewünschten Ausgabedaten zu verwandeln, ohne das dabei ein Mensch involviert ist.

Ein *semi-formales Modell* ist eine Mischung der beiden anderen Arten: Die Transformation der Eingabe in die Ausgabe geschieht nicht vollautomatisch, sondern der Rechner und der Mensch teilen sich die Arbeit. *Semi-formal* bedeutet für uns, daß die Topologie des Netzes formal ist (jeder Knotentyp und jeder Kantentyp hat eine durch

den Interpreter festgelegte Bedeutung für die Lösung der Aufgabe), die Inhalte der Knoten können allerdings nur durch den menschlichen Benutzer interpretiert werden.

Nach unserer Definition sind interaktive Programme, deren Verhalten durch Eingaben des Benutzers beeinflußt werden kann, keine (vollständig) formalen Modelle sondern nur semi-formal. Jede Frage muß von einem Benutzer interpretiert werden. Wenn sein Verständnis der Frage nicht mit dem im Rechner festgelegtem übereinstimmt, dann liefert der Rechner in der Regel ein vom Benutzer nicht erwartetes Ergebnis, das dann üblicherweise als Programmfehler bezeichnet wird.

Daß dem Rechner ein Verständnis der von ihm interaktiv manipulierten Symbole fehlt, läßt sich an einem Beispiel verdeutlichen: Bei der Diagnose eines Ottomotor stellt der Rechner dem Benutzer die Frage “Ist der Vergaser nicht vereist?”. Der Benutzer kann nun über ein Menü mit “ja” oder “nein” antworten. Aus dieser Antwort werden dann weitere Fakten abgeleitet. Überliest der Benutzer nun das Wort “nicht” in der Frage, so verdreht sich die Bedeutung der Antwort ins Gegenteil. Und die vom System hergeleiteten Antworten werden falsch. D.h. das System kann nicht *über* das Frage-Antwort-Paar argumentieren, sondern muß davon ausgehen, daß der Benutzer unter den Symbolen dasgleiche versteht, was in ihm festgelegt wurde.

Hypermedianetze strukturieren Wissen auf der informellen Ebene. Die Semantik der Knoten und Kanten ist nicht formal definiert. Typisiert man nun die Knoten und Kanten, so kann für diese Typen eine operationale Semantik festgelegt werden. Genau dies ist die Basis unseres Interpreters, der somit ein Hypermedianetz zum semantischen Netz erweitert. Aus diesen Überlegungen folgt, daß zwischen (getypten) Hypermedianetzen und den im Expertensystembereich üblichen Wissensrepräsentationsmechanismen nur ein gradueller Unterschied besteht und somit ein sanfter Übergang erreicht werden kann.

5.0 Die KADS-Methodik

Eine in Europa vorrangige Wissensakquisitionsmethodik ist KADS ([19], [20], [5]). Diese ist eine Basis unseres Ansatzes und soll im folgenden kurz geschildert werden, wobei wir unsere speziellen Sichtweisen hervorheben.

5.1 Das 4-Ebenen-Modell

Die Spezifikation einer neuen Applikation wird in KADS *konzeptuelles Modell* genannt. Das konzeptuelle Modell besteht aus der Beschreibung der Benutzerschnittstelle (Model of Cooperation) und dem Wissensmodell (Model of Expertise). Das Model of Expertise unterscheidet verschiedene Wissensarten, die vier verschiedenen Ebenen zugerechnet werden:

- Die Domänen-Ebene umfaßt das anwendungsabhängige Wissen über Konzepte, deren Attribute und Beziehungen. Wir unterscheiden dabei zwischen der Struktur der Domäne (die Definition der Klassen) und den konkreten Objekten (den Instanzen).
- Die Inferenzebene enthält das Wissen über die verwendete Problemlösemethode. KADS unterscheidet dabei zwischen den Rollen (roles), die Konzepte in einem Problemlöseprozeß spielen, und den Aktivitäten (knowledge sources), die zu gegebenen Eingaben eine entsprechenden Ausgabe erzeugen.
- Die Task-Ebene beschreibt den Kontrollfluß einer Problemlösemethode, d.h. diese Ebene beschreibt *wann* eine Aktivität ausgeführt werden soll.
- Die Strategie-Ebene soll Meta-Wissen über die Auswahl und Kombination von verschiedenen Tasks enthalten. Sie ist bis jetzt noch nicht vernünftig beschrieben und wird deshalb in unserem Tool nicht unterstützt.

5.2 Vergleich mit unserem Ansatz

Unser Ansatz baut auf KADS auf, unterscheidet sich aber durch die folgenden Punkte davon:

- **hierarchische Aufgabenstrukturen:** Im Gegensatz zu KADS beschreiben wir eine Aufgabe auf verschiedenen Ebenen der Abstraktion.
- **Toolentwicklung:** Wir versuchen, unsere Methodik direkt durch ein Computer-Aided Knowledge Engineering Tool zu unterstützen; dies ist nicht der Schwerpunkt von KADS.
- **Hypermedia:** Unser Ansatz versucht, Hypermedia-Netze als Zwischenrepräsentation einzusetzen.
- **Model of Cooperation:** Wir planen, Multimedia-Schnittstellen im Model of Cooperation zu definieren. Dies wird durch unsere Basis, ein Hypermediasystem, erleichtert.

6.0 Stand der Realisierung und Ausblick

Das HyperCAKE-System ist vollständig implementiert und an die objektorientierte Datenbank GemStone von Servio Cooperation angekoppelt. CoMo-Kit ist ebenfalls implementiert. Der Interpreter für Aktivitätsstrukturen ist als Single-User-System implementiert, die Erweiterung auf den Multi-User-Betrieb ist in Arbeit.

HyperCAKE/CoMo-Kit bilden die Basis der Entwicklung mehrerer Expertensystem-Shells: SAFRaN koppelt ein geographisches Informationssystem mit einem Expertensystem, um eine wissensbasierte Auswertung von Karten zu ermöglichen (vgl. [6], [8] und [10]). HyDi unterstützt die Entwicklung von hypermediabasierten Diagnosesystemen und wird Ende 1992 fertiggestellt (vgl. [17] und [18]).

7. Literatur

[1] Angele, J.; Fensel, D.; Landes, D.; and Studer, R: KARL: An Executable Language for the Conceptual Model. In: Proceedings of the Knowledge Acquisition for Knowledge-Based Systems Workshop KAW'91, October 6-11, Banff, 1991

[2] Bielawski, L., Lewand, R.: Intelligent Systems Design, Wiley 1991

[3] Biethahn, J.; Bogaschewsky, R.; Hoppe, U. (Hrsg.): Expertensysteme in der Wirtschaft 1992 - Anwendungen und Integration mit Hypermedia, Gabler Verlag, 1992

[4] Boehm, B. W.: A Spiral Model of Software Development and Enhancemant, Computer, 21, 5 (May 1988), p. 61-72

[5] Breuker, J.; Wielinga, B.; Someren, M.v.; de Hoog, R.; Schreiber, G.; de Greef, P.; Bredeweg, B.; Wielemaker, J.; and Billault, J.-P.: Model-Driven Knowledge Acquisition: Interpretation Models. Esprit Project P1098, University of Amsterdam (The Netherlands), 1987

[6] Burde, M.: Die langfristige Sicherung von Grundwasservorkommen - durch die Ausweisung von Grundwasservorranggebieten - als gemeinsame Aufgabe von Raumplanung und Fachplanung, Unveröffentliches Manuskript; Dissertation im Fachbereich ARUBI, Universität Kaiserslautern, 1992

[7] Campbell, B., Goodman, J. M.: HAM: A General Purpose Hypertext Abstract Machine, Communications of the ACM, July 1988, Vol. 31, No. 7

[8] Hemker, H..: Entwurf und Implementierung eines Expertensystems mit GIS-Kopplung, Diplomarbeit Uni Kaiserslautern, 1992

[9] Hoppe, U.: Einsatz von Hypertext/Hypermedia zur Verbesserung der Erklärungsfähigkeit Wissensbasierter Systeme. In [3]

[10] Jäckel, Th.: Entwurf und Implementierung einer Benutzeroberfläche für ein Expertensystem unter Berücksichtigung planerischer Vorgehensweisen, Diplomarbeit Uni Kaiserslautern, 1992

[11] Maurer, F. (Hrsg.): Proc. Workshop "Expertensysteme und Hypermedia", Seki-Working-Paper , 7.11.1991, Kaiserslautern

[12] Maurer, F.: HyperCAKE: Ein Wissensakquisitionssystem für hypermediabasierte Expertensysteme. In [3]

[13] Maurer, F., Pews, G.: Validierung von konzeptuellen Modellen, Proc. XPS-93, Springer, 1993

[14] Neubert, S.: Einsatz von Hypermedia im Bereich der modellbasierten Wissensakquisition. In [3]

[15] Neubert, S., Maurer, F.: The Conceptual Model Construction Kit, to appear

[16] Nielsen, J.: Hypertext & Hypermedia. Academic Press, San Diego, London, 1990

[17] Traphöner, R., Maurer, F.: Integrating Hypermedia and Expert System Technology for Technical Diagnosis, Proc. Expersys 92, Paris, 21.-22. Okt. 1992

[18] Traphöner, R., Maurer, F.: HyDi: Integration of Hypermedia and Expert System Technology for Technical Diagnosis, Proc. Gesellschaft für Klassifikation 92

[19] Wielinga, B.J.; Schreiber, A.Th.; Breuker, J.A.: KADS: A Modelling Approach to Knowledge Engineering. ESPRIT Project P5248 KADS-II, An Advanced and Comprehensive Methodolgy for Integrated KBS Development, Amsterdam, 1991

[20] Wielinga, B.J.; Schreiber, A.Th.; Breuker, J.A.: KADS a modelling approach to knowledge engineering. In: Knowledge Acquisition (1992) 4, 5-53, Academic Press Limited, 1992