Validierung von konzeptuellen Modellen[1]

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

In diesem Papier stellen wir einen Interpreter vor, der die Validierung von konzeptuellen Modellen bereits in frühen Entwicklungsphasen unterstützt. Wir vergleichen Hypermedia- und Expertensystemansätze zur Wissensverarbeitung und erläutern, wie ein integrierter Ansatz die Erstellung von Expertensystemen vereinfacht. Das von uns entwickelte Knowledge Engineering Werkzeug ermöglicht einen "sanften" Übergang von initialen Protokollen über eine semi-formale Spezifikation in Form eines getypten Hypertextes hin zu einem operationalen Expertensystem. Ein Interpreter nutzt die in diesem Prozeß erzeugte Zwischenrepräsentation direkt zur interaktiven Lösung von Problemen, wobei einzelne Aufgaben über ein lokales Rechnernetz auf die Bearbeiter verteilt werden. Das heißt, die Spezifikation des Expertensystems wird direkt für die Lösung realer Probleme eingesetzt. Existieren zu einzelnen Teilaufgaben Operationalisierungen (d.h. Programme), dann werden diese vom Computer bearbeitet.

1.0 Einleitung und Überblick

Bei der Bewertung von Software werden von Boehm [7] zwei Ziele unterschieden:

Verifikation: Erfüllt das Programm die in der Spezifikation gestellten Anforderungen?

Validierung: Löst das Programm das Problem des Anwenders?

In diesem Papier behandeln wir Aspekte des zweiten Punktes. Wir stellen einen Interpreter vor, der eine semi-formale Spezifikation in Kooperation mit dem Benutzer verarbeiten kann und somit bereits in sehr frühen Phasen der Expertensystementwicklung die Validierung unterstützt.

Rapid Prototyping unterstützt die Validierung von Systemen und war das übliche Vorgehen bei der Entwicklung von Expertensystemen der ersten Generation. Die dabei zutage tretenden Schwächen (z.B. entsteht unwartbarer "Spaghetticode" von Regeln) führten zur Entwicklung von modellbasierten Methodiken, z.B. KADS [9], bei der Expertensysteme zuerst spezifiziert und anschließend implementiert werden. In KADS wird das Ergebnis der Analyse einer Domäne *konzeptuelles Modell* genannt. Das konzeptuelle Modell besteht aus der Beschreibung der Schnittstellen zwischen den Agenten (Model of Cooperation) und dem Wissensmodell (Model of Expertise). Als Nachteil der modellbasierten Wissensakquisition ergeben sich Pro-

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"

bleme mit einer frühen Validierung durch die Benutzer, da die (natürlichsprachliche) Spezifikation von diesen oft nicht hinreichend verstanden wird. Verschiedene Autoren ([1], [26], [27], [25]) haben aus diesem Grund formale, operationale Spezifikationssprachen für "*Models of Expertise*" entwickelt. Unser Ansatz geht einen anderen Weg: Wir entwickeln einen Interpreter, der eine semi-formale Spezifikation in Interaktion mit dem Benutzer abarbeiten kann und dabei auch das "Model of Cooperation" validiert.

Unser Ansatz basiert auf der Integration von Hypermedia- und Expertensystemtechniken, die vielen Autoren erfolgversprechend erscheint ([5], [6], [15], [17], [20]). Beide Techniken erlauben die Verwaltung von und den effektiven Zugriff auf Wissen mit Hilfe des Rechners.

Atomare Wissenseinheiten (Knoten) in Hypermediasystemen (vgl. [12] oder [21]) werden typischerweise in einer vom Rechner nicht interpretierbaren Form (z. B. Videosequenzen, Bilder, Tonsignale, natürlichsprachlicher Text) repräsentiert. Das gespeicherte Wissen kann vom Benutzer kontextabhängig 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 (assoziativer Zugriff). Die Suche nach dem für die Lösung des aktuellen Problems benötigten Wissen wird in erster Linie vom Benutzer gesteuert[2]. In Tabelle 1 vergleichen wir an Hand mehrerer Dimensionen die Stärken und Schwächen von Hypermedia- und Expertensystemtechnologie. Die Bewertungen der Technologien bilden nur die Ränder eines Kontinuums.

Dimension	**Hypermedia**	**Expertensystem**
Grad der Formalisierung des Wissens	Wissen nur vom Menschen interpretierbar	Wissen ist formalisiert und dadurch von einem maschinellen Interpreter zu verarbeiten
Initiative bei der Problemlösung	Initiative bei der Problemlösung beim Benutzer, d.h. es werden hohe Anforderungen an ihn gestellt	Initiative bei der Problemlösung beim System, d.h. die Anforderungen an den Benutzer sind niedriger
Entwicklungsaufwand	Mittlerer Entwicklungsaufwand: Formalisierung des Wissens nicht notwendig; Hintergrundwissen und Common Sense des späteren Benutzers muß nicht operationalisiert werden	Hoher Entwicklungsaufwand: Wissensakquisition muß zur Dekontextualisierung von Wissen führen, da Inferenzen nur aufgrund des formalisierten Wissens gezogen werden können
Benutzerschnittstelle	Kommunikation mit Benutzer über multimediale Schnittstelle; Schnittstelle ist ein wichtiger Forschungsgegenstand	Multimediale Benutzerschnittstelle nur als Add-On

Tabelle 1: Vergleich von Hypermedia- und Expertensystemtechnologie

Die Fragen, die sich nun stellen sind: Was können die beiden Ansätze voneinander lernen? Wie kann man die Ansätze miteinander integrieren, so daß die jeweiligen Vorteile übernommen und die entsprechenden Nachteile zurückgedrängt werden? Vorläufige Antworten und ein darauf beruhendes System werden im folgenden dargestellt.

2 "guided tours" verlagern die Initiative stärker auf das System und sind die Ausnahme zu obiger Beschreibung.

Im zweiten Kapitel erläutern wir grob, wie das Wissen einer neuen Anwendung strukturiert wird. Der resultierende Hypertext bildet die Basis des im dritten Kapitel vorgestellten Interpreters, der die verteilte Bearbeitung des Problems ermöglicht. Im vierten Kapitel erläutern wir dann, wie ein "sanfter" Übergang zum Expertensystem geschaffen wird. Im fünften Kapitel vergleichen wir unseren Ansatz mit der KADS-Methodik, die den Ausgangspunkt unserer Entwicklung bildet. Zum Schluß fassen wir unsere Ergebnisse zusammen, beschreiben den Stand der Realisierung und geben einen Überblick über weitere Arbeiten.

2.0 Strukturierung von Wissen

In Abschnitt 2.1 stellen wir das System CoMo-Kit[3] vor. Dieses unterstützt die Entwicklung von Expertensystemen im Sinne eines Computer-Aided Knowledge Engineering Werkzeugs. Abschnitt 2.2 erläutert, wie wir mit CoMo-Kit ausgehend von initialen Daten die Zwischenrepräsentation entwickeln.

2.1 CoMo-Kit: Conceptual Model Construction Kit

CoMo-Kit unterstützt Teams von Experten und Wissensingenieuren bei der Entwicklung von konzeptuellen Modellen. Desweiteren erlaubt CoMo-Kit die Zuweisung einzelner Aufgaben an einen oder mehrere Bearbeiter (Task Distribution, vgl. [13]).

CoMo-Kit basiert auf dem HyperCAKE-System ([18]) und Ideen von [20]. HyperCAKE nutzt eine erweiterte Hypertext Abstract Machine [11] für die Verwaltung multimedialer Informationen. HyperCAKE 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. CoMo-Kit nutzt das HyperCAKE-System zur Verwaltung *aller* im Verlauf des Knowledge Engineering Prozesses anfallenden Daten. Dazu wurden folgende Knotenklassen definiert[4]:

- **Protokoll (Protocol):** Protokolle sind die Ausgangsdaten des Wissensakquisitionsprozesses. Ein Protokoll enthält unstrukturierte Informationen, die vom Experten oder anderen Wissenquellen erhoben wurden. CoMo-Kit unterstützt im Moment Texte, Bitmaps, Audio und Video.
- **Konzept (Begriff, Concept)**: 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[5]. Jede Klassenbeschreibung umfaßt eine Menge von Attributen, denen jeweils ein Typ zugeordnet ist.

---

3 CoMo-Kit wurde in Zusammenarbeit mit Susanne Neubert, Uni Karlsruhe, entwickelt, die eine Beispielapplikation (Zusammenstellen von Versicherungspaketen) zur Verfügung stellte.

4 Wir stellen nur im folgenden relevante Klassen dar.

5 Auf weitere Relationen zwischen Konzepten (PART-Of, CAUSES etc.), die CoMo-Kit unterstützt, gehen wir hier nicht ein.

- **Aufgabe (Task):** Eine Aufgabe beschreibt (in textueller, natürlichsprachlicher Form), was durchgeführt werden muß, um ein gegebenes Problem zu lösen. Für jede Aufgabe werden die Eingabe[6]- und Ausgabedaten[7] spezifiziert, indem Links zwischen Konzepten und ihr definiert werden. Jede Aufgabe kann aus mehreren Unteraufgaben bestehen, die dann in Form eines Datenfluß-Diagramms beschrieben werden. D.h. Tasks sind hierarchisch organisiert. Jeder Aufgabe werden in der Spezifikation ein oder mehrere Bearbeiter zugeordnet, die sie erfüllen können.
- **Bearbeiter (Agent[8]):** Agenten werden über ihren Namen identifiziert und können zu verschiedenen Gruppen gehören. In der Spezifikation wird für jede Aufgabe festgelegt, welcher Agent oder welche Gruppe von Agenten sie potentiell bearbeiten kann. Agenten können sowohl Menschen als auch Rechner sein.

Für Konzepte und Aufgaben können operationale Annotationen in Form von Programmen definiert werden. Auf diesen Punkt kommen wir zurück (vgl. “Operationalisierung der semiformalen Strukturen”).

2.2 Benutzung von CoMo-Kit

Ausgangspunkt der konzeptuellen Modellierung ist das Protokoll eines Gesprächs mit einem Experten, das die zu lösende Aufgabe umgangssprachlich beschreibt. Abbildung 1 zeigt auf

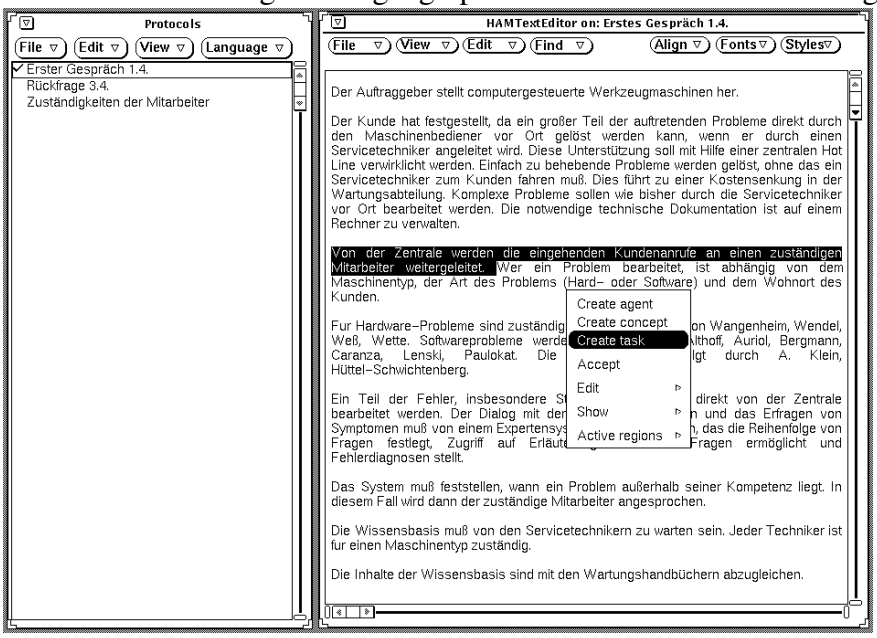

Abbildung 1: Die Liste der Protokolle und ein Protokoll-Editor

der linken Seite eine Liste von Protokollen für eine kleine Beispieldomäne. Rechts ist ein Protokoll-Editor zu sehen. Der Wissensingenieur[9] selektiert einen Teil des Textes, der eine zu

6 Der Benutzer kann als Eingabe für eine Aufgabe sowohl Konzeptklassen als auch -instanzen angeben. Instanzen werden zur Laufzeit nicht verändert und repräsentieren z.B. Gesetzestexte oder Richtlinien. Eingabe einer Aufgabe ist also alles, was zur Bearbeitung benötigt wird.

7 Ausgaben sind immer Instanzen von Konzepten, da ihr Inhalt immer erst als Ergebnis der Aufgabe erzeugt wird.

8 Der Begriff des Agenten bezeichnet den Bearbeiter einer Aufgabe.

bearbeitende Teilaufgabe beschreibt, und wählt anschließend den Menüeintrag “create task”, um einen entsprechenden Aufgabenknoten zu erzeugen. Analog können Konzepte und Agenten erzeugt werden. Sowohl Zuordnung eines Bearbeiters zu einer Gruppe und zu Aufgaben als auch die Modellierung der Konzeptklassenstrukturen erfolgen über graphische Schnittstellen, die hier nicht gezeigt werden.

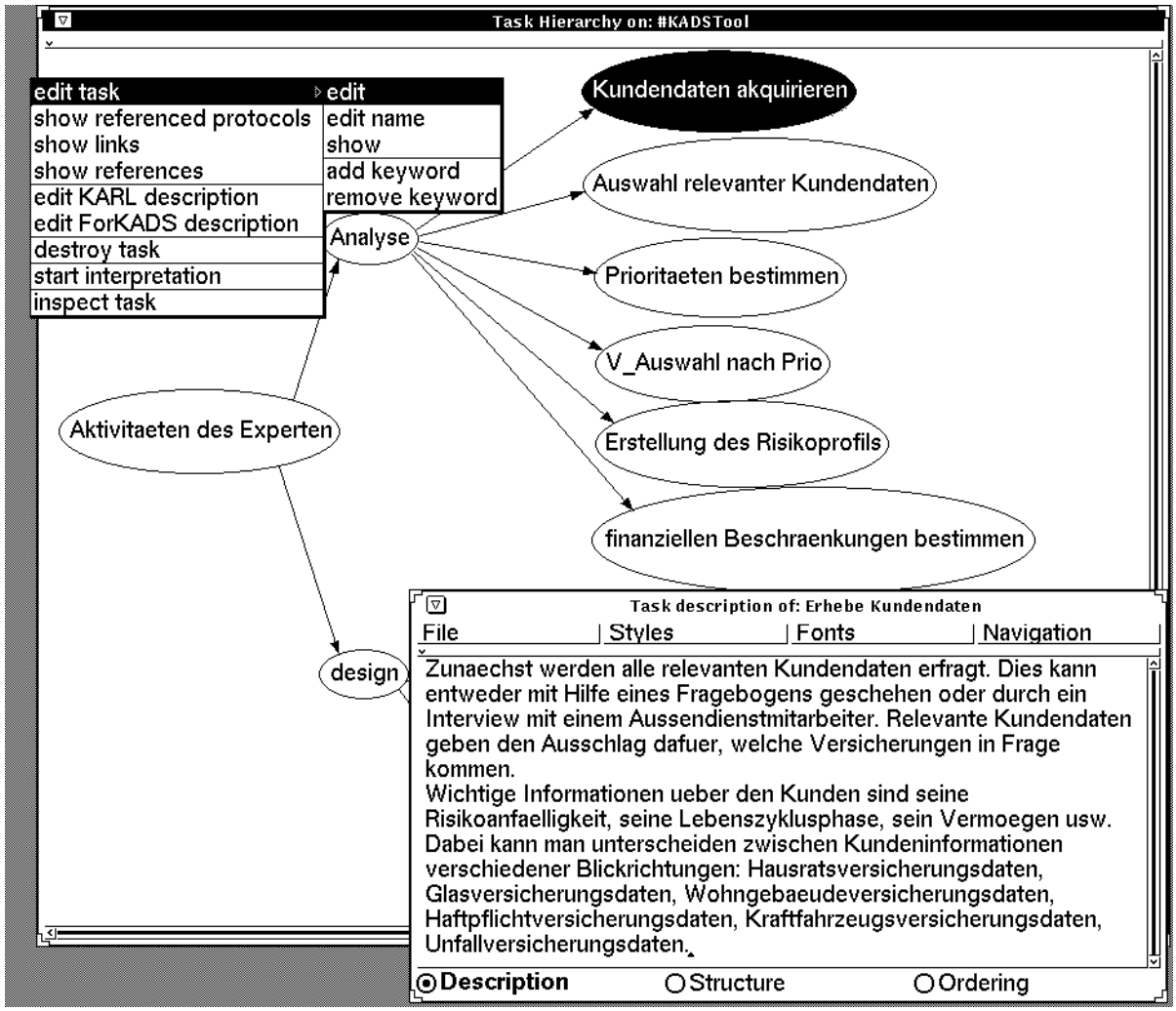

Abbildung 2: Die Aufgaben-Hierarchie und eine Aufgabenbeschreibung

Abbildung 2 zeigt eine Aufgabenstruktur. Desweiteren ist die Beschreibung der in Abbildung 1 neu erzeugten Aufgabe zu sehen. Man erkennt, daß der in dem Protokoll angewählte Text in den neu erzeugten Knoten kopiert wurde. 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 3). 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.

9 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.

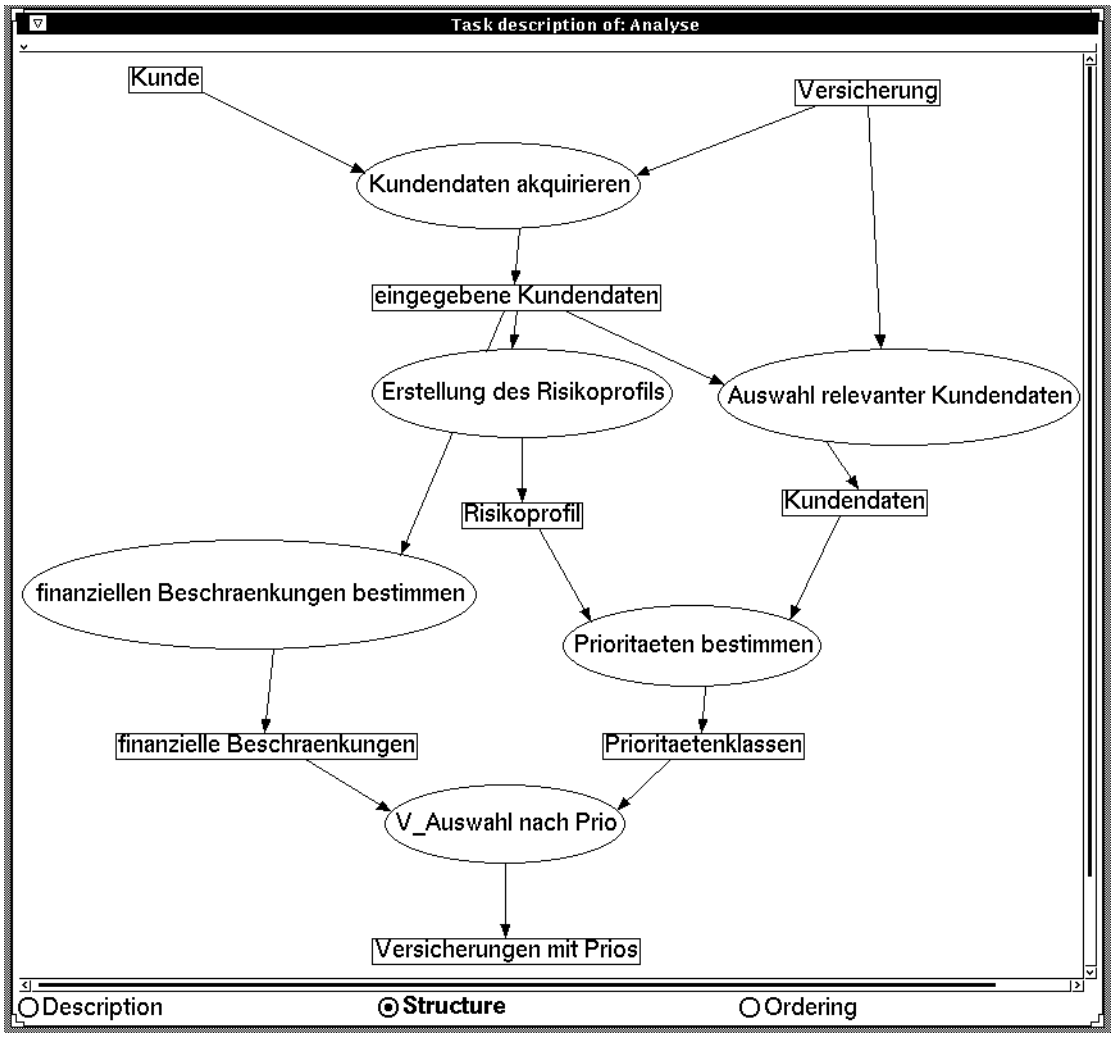

Abbildung 3: Eine Aufgabenstruktur

3.0 Der Interpreter für hierarchische Aufgabenstrukturen

In [13] werden die Vorteile eines “Wizard of Oz”-Experiments bei der Bewertung der Kooperation Mensch-Maschine aufgezeigt. Mit Hilfe des im folgenden beschriebenen Interpreters kann eine neu erzeugte Aufgabenzerlegung direkt von den späteren Benutzern validiert und damit durch ein (leicht abgewandeltes) “Wizard of Oz”-Experiment evaluiert werden. Um zu testen, ob die einzelnen Teilaufgaben für die späteren Benutzer genau genug beschrieben sind, schließen wir Rückfragen an einen Experten aus, indem wir den Benutzern nur den Zugriff auf die Beschreibung der Aufgabe und die Eingabedaten ermöglichen.

Schon die semi-formale[10] Strukturierung der Domäne in Form eines Hypertextes führt zu einem verwertbaren Ergebnis: Komplexe Aufgaben werden in kleinere Teile zerlegt, die dann mit Hilfe des Interpreters von (weniger qualifizierten) Sachbearbeitern bearbeitet werden kön-

10 Semi-formal heißt hier, 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 (vgl. auch [19])

nen. Diese haben nur Zugriff auf für die Aufgabe relevante Information; andere werden vom System weggefiltert.

Abbildung 4 gibt einen Überblick über die verteilte Bearbeitung einer Aufgabe. Ein Benutzer initiiert eine Aufgabe (Task), indem er sie an einen oder mehrere andere Benutzer delegiert und zur Ausführung freigibt. Er startet dadurch einen Scheduler-Prozeß. Der Scheduler erzeugt

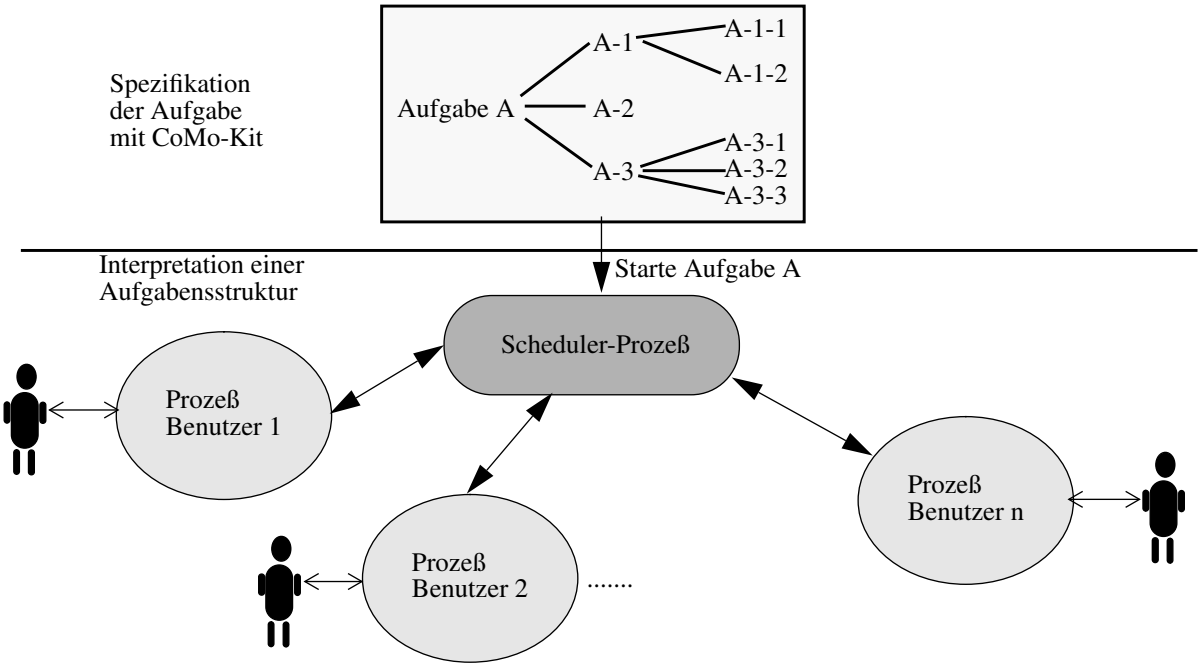

Abbildung 4: Die Prozeß-Struktur des Interpreters

Instanzen aller Ausgabedaten[11]. Er 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 ihm bearbeitbaren Tasks diejenige auswählen, die er als nächstes bearbeiten möchte. Die Tasks lassen sich in zwei Gruppen einteilen: komplexe und atomare.

Komplexe Aufgaben bestehen aus mehreren Unteraufgaben. Sie sind die inneren Knoten des in Abbildung 2 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. Abbildung 5 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)

11 Eine Aufgabe kann mehrmals mit verschiedenen Eingabedaten in Bearbeitung sein (z. B. Zusammenstellen der Versicherungspakete für verschiedene Personen).

- Aufgaben, die gerade bearbeitet werden (rechts unten)
- Aufgaben, die schon bearbeitet sind (rechts oben).

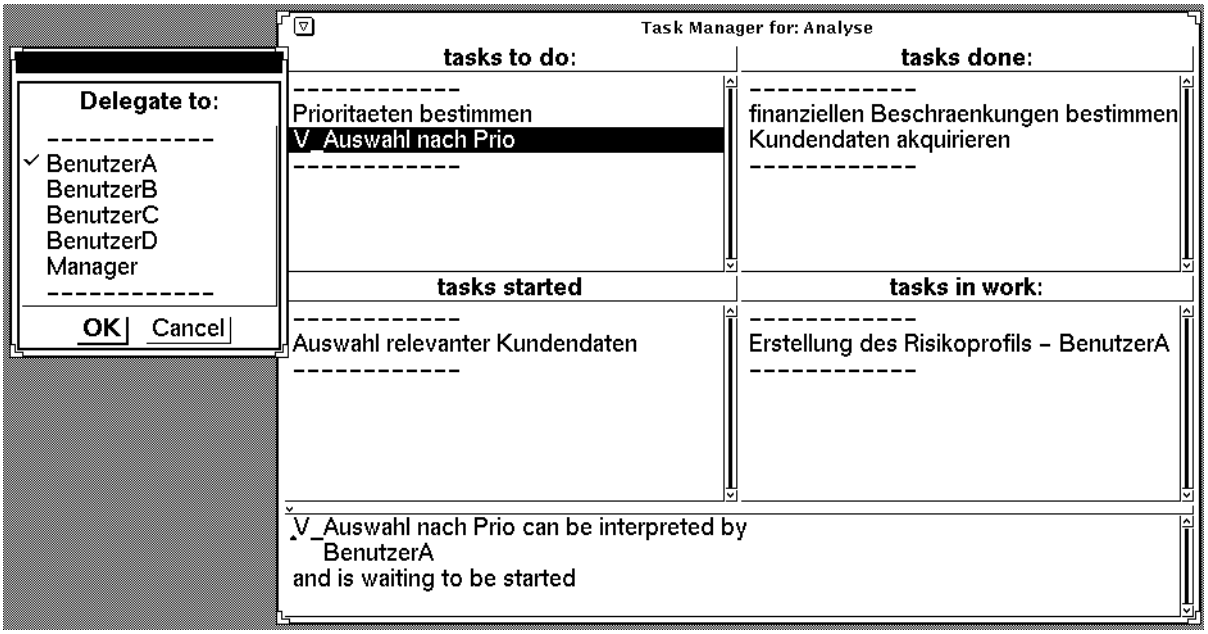

Abbildung 5: Verteilung der Aufgaben auf einzelne Agenten

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). 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 6 gezeigten Fensters bearbeitet. Dieses enthält auf der linken Seite dynamisch erzeugte Buttons, die den Zugriff auf für die Aufgabe relevante[12] Informationen ermöglichen. In der Mitte findet der Benutzer eine Beschreibung der Aufgabe. Rechts befindet sich der Editor für die Eingabe des Resultats[13]. Sobald der Propagate-Button gedrückt wird, werden die Ergebnisse an die Scheduler propagiert. Anschlie-

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

13 Falls die Aufgabe mehrere Resultate erzeugen soll, sind mehrere Editoren zu sehen.

ßend können in dem Datenfluß-Diagramm nachfolgende Aufgaben von den dafür zuständigen Benutzern durchgeführt werden.

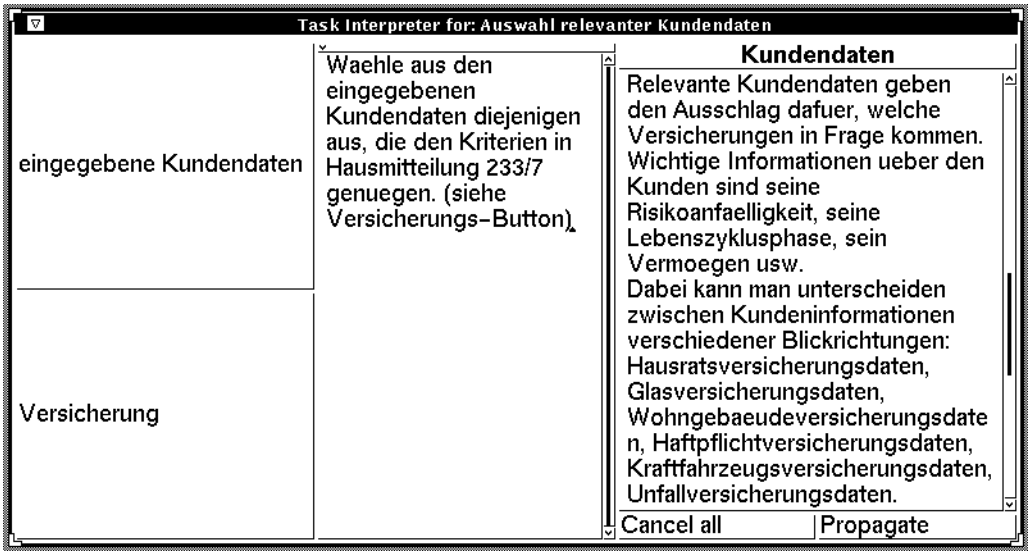

Abbildung 6: Editor für die Bearbeitung einer atomaren Aufgabe durch den Benutzer

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. Dies ist aber nur ein gradueller Unterschied zu interaktiven Expertensystemen, da jede Frage an den Benutzer von diesem auch interpretiert wird. *Aus dieser Überlegung folgt auch, daß die Grenze zwischen Hypertexten und Expertensystemen fließend ist und man einen "sanften" Übergang zwischen beiden Formen der Wissensrepräsentation erreichen kann.*

2. Die Lösung des Problems wird auf mehrere Agenten, d.h. Sachbearbeiter oder Rechner, verteilt. Von daher unterstützt der Interpreter die Validierung des Model of Cooperation.

4.0 Operationalisierung der semi-formalen Strukturen

Der Interpreter für Taskstrukturen leitet die Benutzer bei der gemeinschaftlichen Problemlösung an. Er steuert den Gesamtprozeß. Atomare Aufgaben hingegen werden von den einzelnen Benutzern unter Rückgriff auf die im Hypertext gespeicherte Information gelöst.

Wenn man nun Searl's Chinese Room Experiment in unseren Kontext überträgt, erkennt man, daß der beschriebene Interpreter "Chinesisch" nur partiell übersetzen kann und für die Sätze, die er in seinem Lexikon nicht findet, auf einen externen Agenten (einen Mensch) zurückgreift[14]. Einen Verarbeitungsmechanismus, der einen Teil seiner Aufgaben an externe Agenten weiterleitet, da er sie nicht "versteht", bezeichnen wir als semi-formal. Verschiedene semi-for-

14 Für den Hinweis auf diese Analogie möchte ich mich bei Prof. Richter bedanken.

male Mechanismen lassen sich, über die Anzahl der weitergeleiteten Aufgaben, leicht in eine Halbordnung einordnen. Ziel einer Expertensystementwicklung ist, möglichst tief in diese Halbordnung abzusteigen. D.h. der Verarbeitungsmechanismus bearbeitet möglichst viele Aufgaben selbst und leitet nur wenige an einen Menschen weiter.

Ein Schritt in der Entwicklung, d.h. der Übergang von einem Knoten der Halbordnung in den nächsten, besteht in der Formalisierung einzelner atomarer Aufgaben. Diese werden dann von dem Benutzer "Computer" bearbeitet, was sich einfach in das oben beschriebene Verarbeitungsschema integriert. Um eine spätere Wartung zu unterstützen, sollte die Formalisierung die Struktur des Wissens erhalten ("structure preserving design"). Aus diesem Grund liegt es nahe, eine speziell auf die konzeptuelle Modellierung nach KADS zugeschnittene Sprache zu benutzen. Die Zusammenarbeit mit der Arbeitsgruppe Prof. Studer, Universität Karlsruhe, führte zu der Anbindung der Sprache KARL [1]. CoMo-Kit erlaubt, jedem Konzept und jeder Aufgabe eine formale Annotation zuzuordnen. Die Gesamtheit aller KARL-Beschreibungen eines Projektes kann dann an den KARL-Interpreter übergeben und von diesem ausgeführt werden. Im Moment arbeiten wir an der Integration alternativer Sprachen.

Insgesamt ergibt sich ein "sanfter" Übergang von initialen Daten zum Expertensystem, da der Hypertext als Zwischenrepräsentation hilft, die Aufgabe zu lösen und durch strukturerhaltende Formalisierungen *partiell* zum Expertensystem ausgebaut werden kann. Partiell heißt, daß im Verlauf des Entwicklungsprozesses entschieden werden kann, ob und welche Aufgaben formalisiert werden. Diese Entscheidung wird bestimmt durch den Aufwand der Formalisierung[15] und den erwarteten Nutzen. Mit CoMo-Kit besteht also die Möglichkeit einen beliebigen Teil der Aufgaben zu formalisieren. Das finanzielle Risiko einer Fehlentscheidung zu Beginn wird deutlich verringert, da die Entwicklung spiralförmig in mehreren Phasen durchgeführt werden kann. Dabei können nach jeder Phase das Risiko und die Kosten der folgenden relativ genau eingeschätzt werden (vgl. [8]).

5.0 Vergleich mit der KADS-Methodik

Die KADS-Methodik ist eine Basis unseres Ansatzes und drängt insbesondere in Europa immer mehr in den Vordergrund. Für ausführliche Beschreibungen sei auf z.B. [23], [29], [30] oder [9] verwiesen.

Die Beschreibung von wissensbasierten Systemen wird in der KADS-Methodik in mehrere Modelle aufgespalten: Organizational Model, Application Model, Task Model, Conceptual Model und Design Model.

Das Ergebnis der Analyse einer Domäne wird *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

15 Der nach der Strukturierung der Domäne natürlich besser abgeschätzt werden kann, als bei Projektbeginn.

Wissensarten, die vier verschiedenen Ebenen zugerechnet werden: Domänen-, Inferenz-, Aufgaben- und Strategieebene.

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

- **hierarchische Inferenzstrukturen**: Im Gegensatz zu KADS besteht eine Inferenzstruktur bei uns nicht nur aus primitiven Inferenzen (Knowledge Sources), sondern kann auf verschiedenen Abstraktionsebenen beschrieben werden. Wir setzen dafür hierarchische Datenfluß-Diagramme ein. Diese Art der Modellierung integriert bewußt Aspekte des Task Models mit denen des Inferenzlayers. Die strikte Trennung von Kontroll- von Inferenzaspekten erschien uns unnatürlich, da a priori nicht klar sein kann, was eine primitive und was eine komplexe Aufgabe ist[16]. Hierarchische Inferenzstrukturen sind außerdem notwendig, um eine Modularisierung der Spezifikation und damit einer späteren Implementierung zu unterstützen.
- **Toolentwicklung**: Wir gehen davon aus, daß die Unterstützung der Entwickler durch ein Wissensakquisitionstool ein inhärenter Bestandteil unseres Ansatzes ist. Werkzeuge bestimmen oft, was überhaupt produziert werden kann, und Methodiken sind ohne entsprechende Werkzeuge in der Praxis nicht anwendbar[17]. Dies läßt sich durch eine Analogie zum Maschinenbau begründen: Es ist ein fundamentaler Unterschied, ob in der Produktion eine CNC-Maschine eingesetzt wird oder ein Steinkeil. Entwickler einer Methodik berücksichtigen unserer Ansicht nach automatisch, welche Werkzeuge benutzt werden sollen. Geht man nun davon aus, daß die Methodik mit Papier und Bleistift durchgeführt werden soll, so wird eine "optimale" Methode anders aussehen, als wenn man auf den Rechner als Werkzeug setzt. Die Werkzeugentwicklung ist für KADS zweitrangig (trotz der Entwicklung von Shelley [2]).
- **Inkrementelle Entwicklung**: Unser Ansatz versucht Hypermedia-Netze als Zwischenrepräsentation einzusetzen und diese direkt für die Lösung von Problemen verwertbar zu machen. Wir unterstützen dadurch die inkrementelle Entwicklung von wissensbasierten Systemen nach dem Spiralmodell von Boehm.
- **User Interface**: Da wir als Basis der Entwicklungsumgebung ein objektorientiertes Hypermediasystem benutzen, können wir relativ problemlos multimediale Benutzerschnittstellen erzeugen. Unser Tool unterstützt die schnelle Entwicklung von Benutzerschnittstellen, die dann vom Anwender evaluiert werden können.
- **Knowledge Repository**: Unser Tool verwaltet alle im Rahmen des Knowledge Engineering Prozesses anfallenden Informationen in einer globalen Datenbasis (Knowledge Repository). Analoges wird auch für konventionelle Software Engineering Werkzeuge gefordert.
- **Strategieebene:** Die Strategie-Ebene soll Meta-Wissen über die Auswahl und Kombination von verschiedenen Tasks enthalten. Sie ist bis jetzt noch nicht klar spezifiziert und wird deshalb in unserem Tool nicht unterstützt.

---

16 Wenn man Case-Based Reasoning als KADS-Modell beschreibt (vgl. [3]) stellt man fest, daß die Knowledge Source "select best case" in eine komplexe Unterstruktur zerfällt, sofern die Spezifikation hinreichend genau für eine Implementierung sein soll. Dieser Effekt läßt sich bei vielen der in [4] vorgestellten Modelle feststellen

17 Umgekehrt gilt natürlich auch, daß die Basis eines Werkzeugs eine Methode sein muß.

- **Interaktive Simulation**: Der beschriebene Interpreter verarbeitet in Kooperation mit den Benutzern die semi-formale Spezifikation des wissensbasierten Systems. D. h. die Validierung des wissensbasierten Systems durch die Anwender kann ohne vorherige Formalisierung der Domäne erfolgen, wodurch Probleme frühzeitig erkannt und Kosten gesenkt werden.
- **Validierung des *konzeptuellen* Modells**: Im Gegensatz zu formalen Spezifikationssprachen (KARL, ML^2, ForKADS, MoMo) unterstützt unser Ansatz die Validierung von konzeptuellen Modellen und nicht nur von dem Model of Expertise. D. h. mit Hilfe des beschriebenen Interpreters kann auch die Zuweisung von Aufgaben zu Agenten überprüft werden.

6.0 Zusammenfassung und Ausblick

In diesem Papier haben wir gezeigt, wie man durch die Integration von Hypermedia- und Expertensystemtechniken die Wissensakquisition und -strukturierung unterstützt. Wir haben einen Interpreter beschrieben, der die Validierung einer semi-formalen Spezifikation eines wissensbasierten Systems in Interaktion mit den Anwendern ermöglicht. Dadurch erkennt man frühzeitig, ob das System das Problem der Anwender löst. Durch das (inkrementelle) Ersetzen von einzelnen Teilaufgaben wird ein sanfter Entwicklungsprozeß ermöglicht, der das finanzielle Risiko für den Auftraggeber reduziert.

HyperCAKE und CoMo-Kit sind vollständig implementiert und an die objektorientierte Datenbank GemStone von Servio Cooperation angekoppelt. Der Interpreter für Aufgabenstrukturen ist derzeit als Single-User-System implementiert, die Erweiterung auf den Multi-User-Betrieb wird 1992 abgeschlossen werden. Die Implementierungssprache ist Smalltalk-80 Rel. 4.1 von ParcPlace Systems. Von daher sind die Single-User-Versionen, mit Ausnahme der plattformabhängigen Audio- und Videoschnittstellen, auf verschiedenen Unix-Workstations (Sun, HP/Apollo, IBM, Dec), 386er PC und Apple Macintosh ablauffähig.

HyperCAKE/CoMo-Kit bilden die Basis der Entwicklung mehrerer Expertensystem-Shells: SAFRaN koppelt ein geographisches Informationssystem (GIS) mit einem Expertensystem, um eine wissensbasierte Auswertung von Karten zu ermöglichen. SAFRaN und eine Anwendung sind beschrieben in [10], [14] und [16]. HyDi ([24]) unterstützt die Entwicklung von hypermediabasierten Diagnosesystemen und wird Ende 1992 fertiggestellt.

Danksagung

Prof. Richter hat durch wichtige Anregungen zum Entstehen dieser Arbeit beigetragen. Für viele intensive Diskussionen über den Einsatz von Hypermedia in der Wissensakquisition möchten wir uns bei Susanne Neubert bedanken. Unsere Kollegen in der Arbeitgruppe Expertensysteme von Prof. Richter haben uns durch ein angenehmes Arbeitsklima und wiederholtes Korrekturlesen bei diesem Papier unterstützt.

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] Anjewierden, A., Wielemaker, J., Toussant, C.: Shelley - computer aided knowledge engineering, 1992, in [23]

[3] Bartsch-Spörl, B.: A Simple Interpretation Model for Case-Based Reasoning, 1992, in [4]

[4] Bauer, Ch., Karbach, W.: Proc. 2nd KADS User Meeting, Siemens AG, 17-18. Feb. 1992

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

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

[7] Boehm, B. W.: Software Engineering Economics, Englewood Cliffs NJ: Prentice-Hall, 1981

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

[9] 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

[10] 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

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

[12] Conklin, J.: Hypertext: An introduction and survey. Survey and tutorial series. IEEE Computer, September 1987, S. 17-41

[13] de Greef, H. P., Breuker, J. A.: Analysing system-user cooperation in KADS, In [23]

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

[15] Hoppe, U.: Einsatz von Hypertext/Hypermedia zur Verbesserung der Erklärungsfähigkeit wissensbasierter Systeme, 1992, in [6]

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

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

[18] Maurer, F.: HyperCAKE: Ein Wissensakquisitionssystem für hypermediabasierte Expertensysteme, 1992, in [6]

[19] Maurer, F., Pews, G.: Hypermedia als Zwischenrepräsentation bei der Expertensystementwicklung, Proc. Hypermedia 93, Springer Verlag, 1993

[20] Neubert, S.: Einsatz von Hypermedia im Bereich der modellbasierten Wissensakquisition, 1992, in [6]

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

[22] Sommerville, I.: Software Engineering, Addison-Wesley, 1992 (Fourth edition)

[23] Schreiber, G. (Ed.): Special Issue: The KADS approach to knowledge engineering, Knowledge Acquisition, Vol. 4 No. 1, March 1992, Academic Press

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

[25] van Harmelen, F., Balder, J.: $(ML)^2$: A formal language for KADS models of expertise, 1992, in [23]

[26] Walther, J., Voß, A., Linster, M., Hemmann, Th., Voß, H., Karbach, W.: MoMo, Arbeitpapiere der GMD 658, Juni 1992

[27] Wetter, Th.: First-order logic foundation of the KADS conceptual model, 1990, in [28]

[28] Wielenga, B., Boose, J., Gaines, B., Schreiber, G., van Someren, M. (ed.): Current Trends in Knowledge Acquisition, IOS Press, Amsterdam, Mai 1990

[29] 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

[30] Wielinga, B.J.; Schreiber, A.Th.; Breuker, J.A.: KADS a modelling approach to knowledge engineering, 1992, in [23]