Ein Knowledge-Engineering-Ansatz für kooperatives Design am Beispiel der Bebauungsplanung[1]

Frank Maurer & Gerd Pews
Universität Kaiserslautern
AG Expertensysteme Prof. Richter
Postfach 3049
67653Kaiserslautern
{maurer, pews}@informatik.uni-kl.de
http://wwwagr.informatik.uni-kl.de/{~maurer, ~pews}

1. Motivation

Die Koordination mehrerer an einen Projekt arbeitender Personen stellte schon immer ein Problem der betrieblichen Praxis dar und hat in letzter Zeit an Bedeutung gewonnen. Bedingt durch den sich immer weiter verschärfenden globalen Wettbewerbs haben sich die Produktlebenszyklen stark verkürzt. Daraus folgt insbesondere, daß die Produktentwicklung drastisch beschleunigt werden muß. Erreicht wird dies u.a., indem die Aufgabe auf eine größere Anzahl von Personen verteilt wird, die sie gemeinsam bearbeiten sollen.

Desweiteren gibt es Entwurfsprozesse, die von ihrer Natur her Kooperation erfordern. Bei unserer Anwendung, der Bebauungsplanung, sind Einflußnahmen verschiedener Fachbehörden und auch von Bürgern auf den zu erstellenden Plan vom Gesetzgeber vorgesehen.

Die Koordination der Entwicklungsteams erfordert einen immer größeren Aufwand, der im Extremfall einen Zugewinn an Arbeitsleistung wieder zunichte machen kann. Das klassische Zitat [3] von Brooks „Adding manpower to a late project makes it later“ aus dem Software Engineering weist in die gleiche Richtung.

Prinzipiell gibt es zwei Möglichkeiten dieses Problem anzugehen. Zum einen kann man die Arbeitsabläufe in den Entwurfsprozessen dahingehend optimieren, daß weniger Koordination erforderlich ist. Zum anderen kann die Koordination an sich verbessert werden. Dazu kann man neben organisatorischen und sozialen Maßnahmen auch technische Hilfsmittel einsetzen, worauf wir uns im folgenden konzentrieren wollen.

In diesem Beitrag zeigen wir, wie mit Hilfe von Knowledge-Engineering-Ansätzen Systeme entwickelt werden können, die das kooperative Design unterstützen. Mit „kooperativen Design“ bezeichnen wir Entwurfs- und Konstruktionsprozesse, die die folgenden Eigenschaften haben:

- mehrere Bearbeiter arbeiten zeitlich und/oder räumlich verteilt an einem gemeinsamen Entwurf,
- der Entwurf ist nicht vollständig automatisierbar,
- es bestehen komplexe Abhängigkeiten zwischen den Teilentwürfen der einzelnen Bearbeiter und

1 Die hier beschriebene Arbeit wird teilweise gefördert von der Volkswagen-Stiftung als fachübergreifendes Gemeinschaftsprojekt „Intelligenter Bebauungsplan“ unter Aktenzeichen I/68 707 und I/69 374 - 375. Das Projekt wird an der Universität Kaiserslautern durchgeführt von den Arbeitsgruppen Prof. Richter, Prof. Stich und Prof. Streich.

- im Verlauf des Entwurfsprozesses müssen die Bearbeiter auf komplexe und umfangreiche Informationen („Wissen“) wie z.B. Normen, Richtlinien, Lehrbücher zurückgreifen.

Durch ein praxisnahes Beispiel des kooperativen Designs, die Erstellung von Bebauungsplänen, wollen wir den Nutzen von Knowledge-Engineering-Methoden verdeutlichen. Die Anwendung wird im Anschluß vorgestellt. Im Abschnitt 3 beschreiben wir, wie eine Applikation mit unseren Techniken modelliert wird. Abschnitt 4 erläutert die Umsetzung des Modells, deren Ziel im Sinne des Prototyping ein lauffähiges System ist. Im fünften Abschnitt berichten wir aus der Sicht des Knowledge Engineering von unseren bisherigen Erfahrungen bei der Entwicklung des IBP-Systems. Zum Abschluß fassen wir unsere Ergebnisse zusammen und zeigen noch offene Probleme auf.

2. Die Anwendung „Bebauungsplanung“

Gemeinden haben im Rahmen der Bauleitplanung die Aufgabe, die bauliche und sonstige Nutzung von Grundstücken vorzubereiten und zu leiten (vgl. [16]). Ein Teil davon besteht in der Erstellung eines Bebauungsplans. Dieser legt durch zeichnerische (siehe Abbildung 1) und textliche Festsetzungen Art und Maß der baulichen Nutzung von Gebieten innerhalb einer Gemeinde fest. Beispielsweise kann so festgelegt werden, ob ein Baugebiet ein reines Wohngebiet ist (Art der baulichen Nutzung), wie groß die maximale Anzahl der Vollgeschosse ist und wie hoch die baulichen Anlagen sein dürfen (Maß der baulichen Nutzung). In der Regel ist als Vorarbeit für den Bebauungsplan bereits ein städtebaulicher Vorentwurf erstellt worden, im dem schon Ideen entwickelt worden sind, wie das betreffende Gebiet genutzt werden soll. Der gestalterisch-kreative Akt ist also nicht der Inhalt eines Bebauungsplans, sondern die rechtlich verbindliche Umsetzung des Vorentwurfs als ein Teil der kommunalen Gesetzgebung.

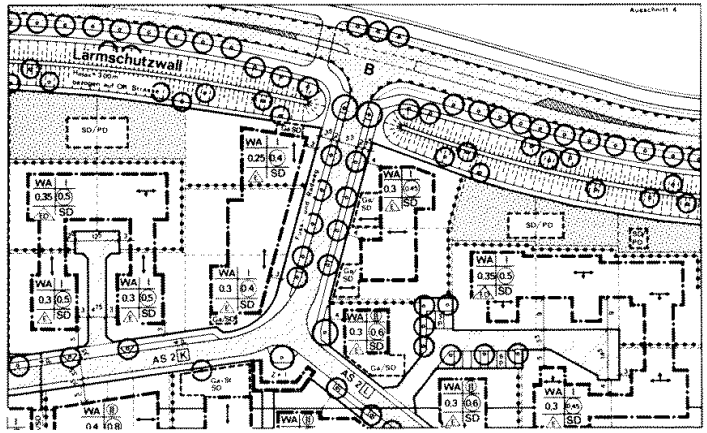

Abbildung 1: Zeichnerischer Teil eines Bebaungsplans (nach Hangarter [10])

Die Erstellung von Bebauungsplänen weist Aspekte auf, die die Problemstellungen des kooperativen Designs berühren:

- Der fertige Plan muß inhaltlich den rechtlichen Anforderungen genügen. Dazu müssen besonders die *gesetzlichen Grundlagen* für die einzelnen Arbeitsschritte bekannt sein. Zu jedem Schritt benötigt der Planer umfangreiches Wissen über die gerade benötigten Gesetze, Verordnungen und Kommentare. Oft - etwa wenn bei der Ansiedlung von Industrie Umweltschutzbelange berücksichtigt werden müssen - sind noch zusätzliche Informationen notwendig.
  Der Planer benutzt große Mengen von Wissen. Er benötigt unter anderem Wissen über das Baugesetzbuch, die Baunutzungsverordnung, die dazu gehörigen Kommentare und die aktuelle Rechtsprechung.
- Die Erstellung von Bebauungsplänen ist nicht vollständig automatisierbar, da sie im Sinne von Gesetzgebung und Rechtsprechung stattfinden soll und große Mengen von Wissen über die „reale Welt“ voraussetzt. So könnte man festsetzen, daß Hausdächer ortsüblich eingedeckt werden müssen, obwohl sich auf der anderen Straßenseite ein Industriegebiet befindet. Ein Algorithmus ist nicht in der Lage abzuwägen, ob diese Festsetzung vertretbar ist.
- Es bestehen komplexe Abhängigkeiten zwischen den einzelnen Festsetzungen. Die Entscheidung etwa, ob ein Gebiet als reines Wohngebiet oder als Dorfgebiet ausgewiesen wird, kann weitreichende Folgen haben. Treten Änderungen im Plan auf - und das ist während der Erstellung eines Bebauungsplans die Regel - müssen solche Abhängigkeiten berücksichtigt werden.
- Es werden Koordinationsstrukturen für mehrere beteiligte Personen benötigt. Es kommt zwar in der Praxis oft vor, daß ein Bebauungsplan nur von einem Planer allein erstellt wird, allerdings besteht trotzdem der Zwang zur Koordination. Einerseits gibt es immer andere Personen, die auf den Planungsvorgang Einfluß nehmen, wie etwa übergeordnete Planungsbehörden oder der interessierte Bürger im Rahmen der Bürgerbeteiligung. Auf der anderen Seite kann es vorkommen, daß größere zeitliche Abstände zwischen verschiedenen Bearbeitungsschritten oder Überarbeitungen liegen, die auch von anderen Planern ausgeführt werden können.

  Auf jeden Fall besteht die Notwendigkeit, die Gedankengänge nachzuvollziehen, die bei erledigten Bearbeitungsschritten stattgefunden haben, sei es vom Planer selbst, der sich in seine eigene Arbeit wieder einarbeiten muß, oder sei es von Außenstehenden, die den Plan verstehen müssen. Die Abhängigkeiten zwischen den Bearbeitungsschritten werden genau so verwaltet wie bei der Koordination von gleichzeitig arbeitenden Planern.
- Das Verfahren der Erstellung muß ordnungsgemäß ablaufen, damit der Plan später rechtsgültig werden kann. Dabei hat der Gesetzgeber z.B. vorgegeben, wie die Bürger bei der Planerstellung beteiligt werden müssen oder welche Auslagefristen eingehalten werden müssen.

Aus der beschriebenen Anwendung ergeben sich Anforderungen an die Modellierungsmethodik, die wir im nächsten Abschnitt beschreiben.

3. Modellierung der Arbeitsabläufe

Um ein System zur Unterstützung des kooperativen Designs zu realisieren, ist eine umfassende Analyse der Arbeitsabläufe, in unserem Fall der Bebauungsplanerstellung, nötig. Im einzelnen muß herausgearbeitet werden:

- Die einzelnen Bearbeitungsschritte, in die der gesamte Vorgang zerfällt.

- Abhängigkeiten zwischen diesen Bearbeitungsschritten.
- Alternative Methoden, eine Aufgabe zu bearbeiten.
- Die nötigen Informationen in Form von Teilen von Lehrbüchern, Normen, Gesetzen, Verordnungen und Kommentaren für jeden Schritt (d.h. das zur Problemlösung benötigte Wissen muß erfaßt werden).
- Eine Aufarbeitung der inhaltlichen Querverweise zwischen den einzelnen Informationen.
- Die Zuordnung von Arbeitsschritten zu Bearbeitern.
- Es muß festgelegt werden, welche Arbeitsschritte von Menschen und welche automatisch von Rechnern durchgeführt werden sollen.

Das dokumentierte Ergebnis dieser Analyse ist ein konzeptuelles Modell (eine Spezifikation) eines kooperativen wissensbasierten Systems. Zur Spezifikation von wissensbasierten Systemen bietet es sich an, auf Methoden des modellbasierten Knowledge Engineering[2] zurückzugreifen. Kooperative Arbeitsabläufe, an denen mehrere Agenten bzw. Systeme bei der Problemlösung zusammenarbeiten müssen, sind mit diesen Ansätzen aber nicht „natürlich“ beschreib- und operationalisierbar. Die Operationalisierung eines konzeptuellen Modells, d.h. die Implementierung der Spezifikation, ist letztlich aber das Ziel der Systementwicklung.

Unser Bestreben war nun, einen methodischen Rahmen und die dazugehörigen Werkzeuge zu entwickeln, die die Beschreibung und prototypische Implementierung von kooperativen wissensbasierten Anwendungen ermöglicht. Dabei ergeben sich aufgrund unseres Anwendungsgebiets folgende Anforderungen:

- **Auswahl der Aufgabenzerlegung während der Problemlösung:** Unser Ansatz zielt auf die Unterstützung von Design-Aufgaben, wobei die Bebauungsplanung als Beispiel dient. Design-Probleme können in der Regel auf verschiedene Weisen gelöst werden. Die geeignete kann nur aufgrund des aktuell vorliegenden Problems bestimmt werden. Deshalb läßt unser konzeptuelles Modell alternative Aufgabenzerlegungen zu, von denen zur Laufzeit des Systems eine gewählt wird.[3]
- **Zielgerichtetes Backtracking:** Zur Lösung von Problemen im Entwurf müssen Entscheidungen getroffen werden. Oft basieren diese auf zusätzlichen Annahmen, die sich im Nachhinein als falsch herausstellen können und dann zu einer Inkonsistenz im Problemlöseprozeß führen. Als Folge davon muß eine der Entscheidungen, die zu der Inkonsistenz geführt haben, zurückgenommen werden. Ein zielgerichtetes, daher effizientes Backtracking wird von den bisher implementierten Interpretern für konzeptuelle Modelle nicht unterstützt.
- **Integration von menschlichen Bearbeitern in den Problemlöseprozeß:** Die vollständige Formalisierung jedes Problemlöseschritts ist in unserer Domäne nicht möglich: Viele (Teil-)Aufgaben werden auch in Zukunft von menschlichen Bearbeitern gelöst werden, wobei der Zugriff auf benötigtes Wissen (z.B. Gesetze, Verordnungen, Richtlinien,

2 Unter modellbasiertem Knowledge Engineering verstehen wir in diesem Beitrag KADS und die daran angelehnten Ansätze [1], [2], [18], [19], [23].

3 Die Selektion von Aufgabenzerlegungen zur Laufzeit kann als Beispiel für die Verwendung der (nicht ausgearbeiteten) Strategieebene der KADS-Methodik angesehen werden.

Normen etc.) durch Hypertext-Techniken vereinfacht werden soll. Die Ergebnisse dieser Problemlöseschritte werden anschließend dem wissensbasierten System übergeben und dann im weiteren Verlauf des Problemlöseprozesses benutzt.

- **Modellierung von kooperativen wissensbasierten Systemen:** Das konzeptuelle Modell muß die Kooperation der verschiedenen Agenten beschreiben. Deshalb muß der methodische Rahmen den Begriff des Agenten enthalten.

Um kooperative wissensbasierte Systeme zu beschreiben benutzt unsere Methodik vier Grundbegriffe (vgl. [11]): Aufgabe (task), Methode (method), Konzept (concept) und Agent (agent).

Aufgaben und Methoden: Eine Aufgabe wird durch ihr Ziel, ihre Eingaben und ihre Ausgaben beschrieben. Um eine Aufgabe zu bearbeiten, wird eine Methode angewandt. Für jede Aufgabe kann es mehrere alternative Methoden geben. Methoden werden von Agenten ausgeführt.

Wir unterscheiden zwischen atomaren und komplexen (zusammengesetzten) Methoden. Atomare Methode weisen Variablen aktuelle Werte zu. Eine komplexe Methode wird als Datenflußgraph (siehe Abbildung 2) beschrieben. Ein Datenflußgraph[4] besteht aus Knoten, die (Unter-)Aufgaben, Variable und zur Problemlösung benötigtes Wissen definieren, und aus Kanten, die den Aufgaben ihre Ein- bzw. Ausgaben zuordnen. Jeder Variable ist ihr Typ (Konzeptklasse) zugeordnet, von dem zur Laufzeit eine Instanz erzeugt wird, die dann an die Variable gebunden werden muß. Jede Unteraufgabe kann weiter mit Hilfe von Methoden zerlegt werden. Aufgaben in parallelen Zweigen können gleichzeitig von verschiedenen Agenten bearbeitet werden, d. h. das System interpretiert die Datenflußgraphen als gefärbte Petri-Netze (ein ähnlicher Ansatz liegt [9] zugrunde).

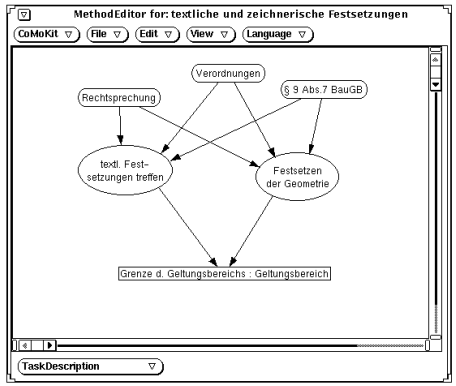

Abbildung 2: Ein einfacher Datenflußgraph

4 Aufgaben werden in der Graphik durch Ovale repräsentiert. Rechtecke zeigen Variable und deren Typ. Abgerundete Rechtecke enthalten Wissen. Dieses ist während der Problemlösung konstant. In unserer Anwendung wird dadurch der Zugriff auf einzelne Teile von Gesetzen (z.B. Paragraphen oder Absätze) ermöglicht.

Insgesamt wird der Problemlöseprozeß als UND-ODER-Baum beschrieben, wobei Aufgaben die UND-Knoten sind und Methoden den ODER-Knoten entsprechen. Durch die Datenflußbeziehungen entstehen allerdings Reihenfolgeabhängigkeiten, die über einen reinen UND-ODER-Baum hinausgehen und einen Kontrollfluß definieren. Abbildung 3 zeigt einen Ausschnitt der Aufgabenzerlegung unserer Beispielanwendung. Eine Aufgabe wird mit Hilfe eines Ovals dargestellt. Rechtecke repräsentieren in dieser Sicht Methoden.

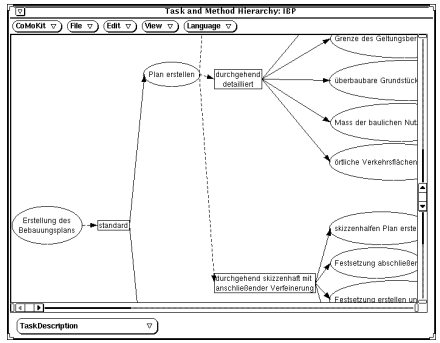

Abbildung 3: Aufgabenzerlegung der Beispielanwendung

Agenten: Aufgaben werden von Agenten bearbeitet. Im konzeptuellen Modell wird für jede Aufgabe definiert, welche Agenten fähig sind, sie zu bearbeiten (siehe Abbildung 4). Erst zur Laufzeit des Systems wird festgelegt, welcher Agent sich um eine konkrete Aufgabe kümmern soll.

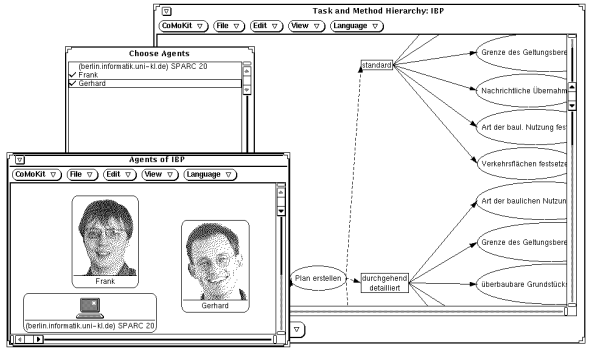

Abbildung 4: Die Zuordnung von Aufgaben zu Agenten

Agenten können nach ihren Fähigkeiten zu Gruppen zusammengefaßt werden, z.B. der Gruppe aller Personen mit PC-Kenntnissen. Dadurch ist es möglich zu spezifizieren, daß eine Aufgabe von einer Person mit PC-Kenntnissen gelöst werden soll.

Wir unterscheiden zwischen zwei verschiedenen Arten von Agenten: Menschen und Rechner. In Abhängigkeit von den zugeordneten Agenten werden Methoden in einer natürlichen oder formalen Sprache beschrieben.

Konzepte: Datenstrukturen werden mit einem objekt-zentrierten Ansatz modelliert, wobei wir zwischen Klassen und Instanzen unterscheiden. Klassen definieren die Struktur der Instanzen durch eine Menge von Attributen (Slots). Jedem Attribut wird ein Typ und eine Kardinalität zugeordnet. Typen sind andere Konzeptklassen oder Basistypen wie SYMBOL, STRING, REAL, etc. Ein ausgezeichneter Basistyp HT-ANCHOR definiert eine Einsprungstelle in einen Hypertext.[5]

Instanzen der Ankerpunkte HT-ANCHOR repräsentieren Einsprungstellen in Gesetze, Verordnungen, Kommentare und sonstige Informationen. Im Modell des Arbeitsablaufs sind die Ankerpunkte Eingaben für einzelne Arbeitsschritte, d.h. der Bearbeiter kann auf die zum Lösen einer Aufgabe notwendigen Informationen gezielt zugreifen und muß nicht lange im Hypertext suchen.

Die Paragraphen der Gesetze enthalten dann beispielsweise wieder Querverweise zu Abschnitten anderer Gesetze oder zu zugehörigen Kommentaren.

Klassen werden in einer Vererbungshierarchie angeordnet. Abbildung 5 zeigt einen Ausschnitt der Klassenhierarchie unserer Beispielanwendung.

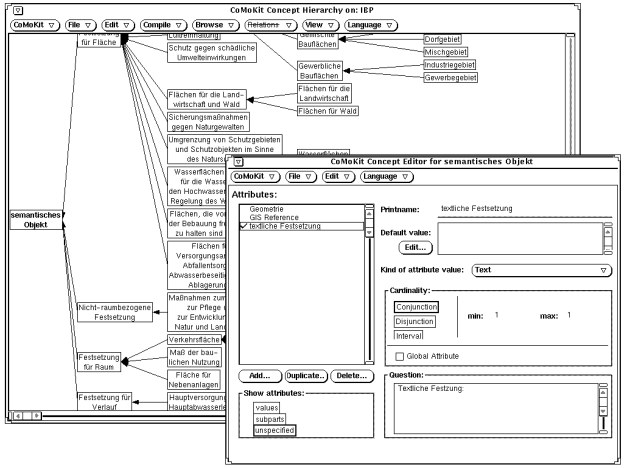

Abbildung 5: Hierarchie und Attribute von Konzepten

Unser Modellierungswerkzeug CoMo-Kit umfaßt einen Generator, der zu der spezifizierten Hierarchie automatisch Klassen in Programmiersprachen erzeugt.[6] Desweiteren wird für jede Klasse eine Bildschirmmaske[7] generiert, mit deren Hilfe Instanzen der

5 Die einzelnen Hypertextdateien werden im HTML-Format des WWW abgelegt.

6 Im Moment wird Programmcode in Smalltalk und in Magik, der Programmiersprache des für unsere Anwendung eingesetzten geographischen Informationssystems, erzeugt (vgl. auch Abschnitt 4.3).

7 Zum Generieren von Benutzungsschnittstellen vgl. auch [8].

Klasse editiert werden können. Diese Maske kann mit Hilfe eines graphischen Interface-Builders überarbeitet werden. Durch den Code-Generator und den Einsatz eines graphischen Interface-Builders wird die Entwicklung von Benutzungsschnittstellen stark beschleunigt.

Mit Hilfe der beschriebenen Terminologie lassen sich Modelle von kooperativen Arbeitsabläufen spezifizieren. Diese Modelle sind für reale Anwendungen zu komplex, um in allen Einzelheiten von Anwendern verstanden zu werden. Von daher sind diese kaum in der Lage, zu entscheiden, ob das spezifizierte System ihnen die gewünschte Unterstützung bieten kann. Dieses Problem kann mit Hilfe eines Prototypen überwunden werden. Um den Entwicklungsaufwand für diesen möglichst gering zu halten, haben wir Werkzeuge entwickelt, die ein konzeptuelles Modell in dem oben definiertem Sinn als Eingabe erhalten und dieses in Interaktion mit den Benutzern abarbeiten.[8]

4. Nutzung des konzeptuellen Modells

Zunächst beschreiben wir die Architektur unseres Interpreters, der konzeptuelle Modelle operationalisiert. Der Interpreter unterstützt die Entwicklung von Prototypen und ermöglicht die Durchführung von Projekten. Anschließend gehen wir kurz auf die Mechanismen zur Abhängigkeitsverwaltung ein, die das zielgerichtete Backtracking unterstützen. Wir schließen das Kapitel mit einer Übersicht über die Architektur des IBP-Systems

4.1 Architektur des Interpreters

Der Interpreter hat eine Client-Server-Architektur. Den Server bezeichnen wir als Scheduler. Dieser speichert anstehende Aufgaben und die aktuell gültigen Variablenbindungen. Zur Verwaltung von Abhängigkeiten zwischen Aufgaben benutzen wir ein erweitertes TMS (vgl. [4], [5], [13]). Aufgaben werden von Clients bearbeitet: Ein Client greift auf eine Aufgabe zu, bestimmt die anzuwendende Methode, führt diese aus und transferiert die dabei entstandenen Ergebnisse zurück zum Scheduler. Die Schnittstelle zwischen Server und Client ist genau in [17] beschrieben.

Um Aufgaben zu bearbeiten, werden Methoden angewandt. In komplexen Methoden werden Aufgaben in Unteraufgaben zerlegt und lokale Variablen definiert. Atomare (primitive) Methoden belegen Variable mit Werten. Die Auswahl einer Methode ist eine Entscheidung im Problemlöseprozeß, die später zurückgezogen werden kann.

Abbildung 6 erläutert die Arbeitsweise des Interpreters: Aufgabe A eines konzeptuellen Modells wird gestartet. *User-1* wählt *method-1*, um Aufgabe A zu zerlegen. Deshalb werden die Aufgaben *A-1-1* und *A-1-2* in die Agenda der anstehenden Aufgaben aufgenommen. Anschließend akzeptiert *User-2* die Aufgabe *A-1-1* und *Computer-1* bearbeitet Aufgabe *A-1-2*. Der resultierende Zustand ist der Abbildung dargestellt.

Der Scheduler verwaltet den aktuellen Stand der Problemlösung mit Hilfe einer Menge von Listen, die die verschiedenen Abarbeitungszustände (z.B. „in Bearbeitung“, „Eingaben fehlen“, „blockiert“ etc.) von Aufgaben repräsentieren. Die Zustände und Zustandsübergänge sind genau in [4] und [5] beschrieben.

8 [12] beschreibt die Grundidee der interaktiven Validierung von konzeptuellen Modellen genauer.

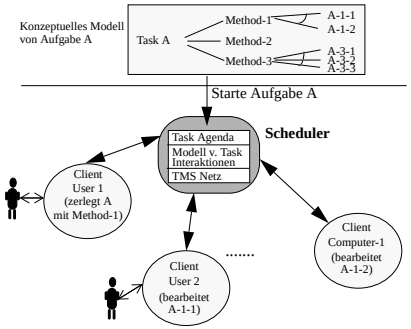

Abbildung 6: Architektur des Interpreters

4.2 Abhängigkeitverwaltung durch den Scheduler

Die im Laufe des Lösungsprozesses anfallenden Aufgaben werden durch die Anwendung von Methoden gelöst. Ein Agent muß sich während des Lösungsprozesses für eine der alternativen Methoden entscheiden. Die durch die ausgewählten Methoden hergeleiteten Teilaufgaben und Konzeptinstanzen stellen die gültige (Teil-)Lösung, die Entscheidungen den Lösungsweg dar. Der Scheduler verwaltet den aktuellen Stand des Problemlöseprozesses und die entstehenden Abhängigkeiten. Zu seinen Aufgaben gehört die Verwaltung

- der Zustände, die eine Aufgabe annehmen kann,
- der Beziehungen, die sich aus der Zerlegung von Aufgaben in Teilaufgaben ergeben,
- der Abhängigkeiten, die zwischen einer atomaren Methode und der damit verbundenen Variablenbelegung bestehen (insbesondere müssen der Datenfluß und die damit verbundenen zeitlichen Abhängigkeiten zwischen den Aufgaben gesteuert werden),
- der durch die Anwendung von komplexen oder atomaren Methoden entstehenden Entscheidungen und
- der Delegierung von Aufgaben an ihre Bearbeiter.

Der Scheduler unterstützt den Rückzug von Entscheidungen, die zu einer inkonsistenten Lösung führen. Der Rückzug von Entscheidungen hat in der Regel globale Auswirkungen auf den Lösungsprozeß:

- Die Anwendung einer komplexen Methode auf eine Aufgabe zerlegt diese in neue Teilaufgaben. Der Rückzug einer Zerlegung führt dazu, daß die resultierenden Aufgaben und Lösungen ihre Gültigkeit verlieren. Dies pflanzt sich in dem gesamten Lösungsbaum fort.
- Atomare Methoden belegen Variablen mit Werten, die die Grundlage für weitere Entscheidungen sind. Nach dem Rückzug muß jede Lösung zurückgenommen und überdacht werden, die auf der Gültigkeit der bisherigen Belegung beruht.
- Sind alle einer Aufgabe zugeordneten Methoden nicht (mehr) anwendbar, kann die Aufgabe nicht mehr bearbeitet werden. Die resultierende Blockade kann durch chronologisches Backtracking behoben werden. Effizienter und sinnvoller ist es jedoch, ursachengesteuertes Backtracking durchzuführen.

Für Entscheidungen, die von der Gültigkeit anderer Entscheidungen abhängen, werden daher Rückzugsbedingungen formuliert. Damit kann das System die Ursachen einer Inkonsistenz ermitteln und zur Auflösung von Blockaden durch abhängigkeitsgerichtetes Backtracking beitragen.

Als Basismodell der Abhängigkeitsverwaltung dient uns das allgemeine Planungs- und Designmodell REDUX [14][15], welches die Dekomposition von Aufgaben, den Entscheidungsrückzug und abhängigkeitsgerichtetes Backtracking ermöglicht.

Abhängigkeiten entsprechen logischen Implikationen. Um die Menge aller Abhängigkeiten zu verwalten, setzen wir ein TMS [6] ein. Die Änderungen der logischen Werte von Formeln werden durch den Markierungsalgorithmus des TMS effizient propagiert. Unser System bildet alle Aufgaben, Entscheidungen und deren Abhängigkeiten auf TMS-Strukturen ab. Die Techniken zur Abhängigkeitsverwaltung sind genau in [4] und [5] beschrieben.

4.3 Systemarchitektur IBP und Arbeitsumgebung des Planers

Nachdem wir in den vorhergehenden Abschnitten beschrieben haben, welche Methodik unserem Modell zugrunde liegt, wollen wir im folgenden auf die Architektur des IBP-Systems für die Bebauungsplanung eingehen und die von uns entwickelte Arbeitsumgebung des Planers erläutern.

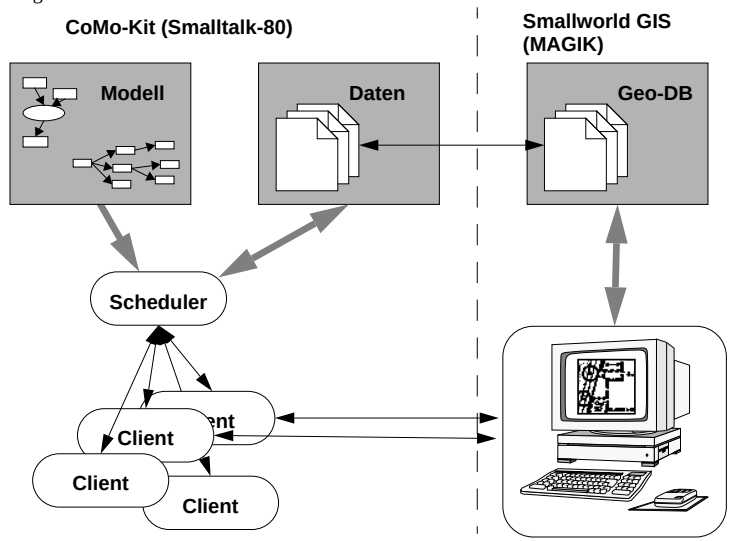

Abbildung 7: Systemarchitektur von IBP

Das Rückgrad von IBP bildet das CoMo-Kit-System (siehe Abbildung 7). Dieses stellt Werkzeuge zur Verfügung, um Arbeitsabläufe und Datenstrukturen zu modellieren, die bereits in Abschnitt 3 erläutert wurden. Mit diesen wird der Vorgang der Bebauungsplaner-

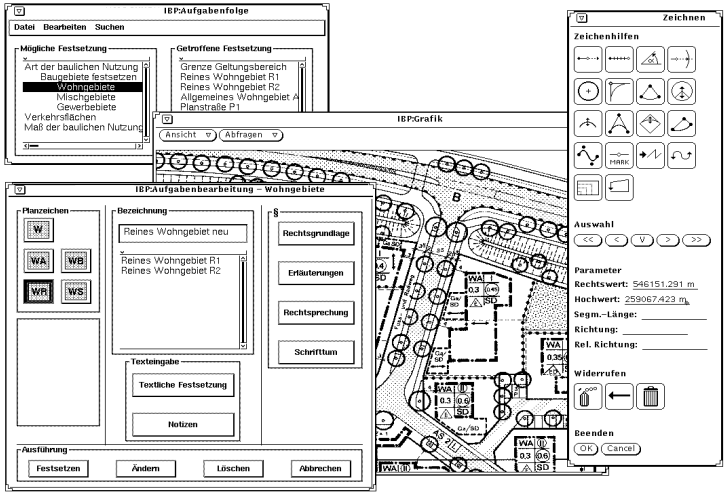

Abbildung 8: Die Arbeitsumgebung des Planers

stellung und die Begriffswelt der Domäne modelliert. Der Interpreter, der das Modell operationalisiert, wurde in den vorhergehenden Abschnitten beschrieben.

Zum Zeichnen der Pläne verwenden wir ein kommerziell verfügbares geographisches Informationssystem (GIS) mit CAD-Komponente (Smallworld GIS). Das GIS ist über eine UNIX-Socket-Schnittstelle mit Clients gekoppelt und verwaltet geometrisch-topologische Daten. Die Verwaltung aller anderen Daten obliegt dem Scheduler.

Die Arbeitsumgebung des Planers ist in Abbildung 8 dargestellt. Im Zentrum sieht man den zeichnerischen Teil eines Bebauungsplans, der mit Hilfe des GIS/CAD dargestellt wird. Rechts daneben befindet sich die Palette mit den Zeichenwerkzeugen. Links unten befindet sich ein Editor, mit dem textliche und andere Festsetzungen für das gerade ausgewählte Planelement getroffen werden können. Der Planer ersieht den Stand der Aufgabenbearbeitung in dem darüber liegenden Fenster, in dem Listen mit den bereits abgearbeiteten und den noch anstehenden Aufgaben zu sehen sind. Nicht dargestellt ist der Hypertexteditor mit den gesetzlichen Grundlagen und Kommentaren.

5. Erfahrungen

Das konzeptuelle Modell der Anwendung wurde gemeinsam von einem Informatiker, einem Raumplaner und zeitweise einem Experten für Baurecht direkt am Bildschirm mit dem CoMo-Kit-System erstellt. In einem ersten Schritt wurden die Datenstrukturen definiert, die zur Beschreibung des Bebauungsplan notwendig sind. Die zu modellierenden Objekte sind im wesentlichen durch das Baugesetzbuch und die Planzeichenverordnung vorgegeben. Der Aufwand der Modellierung bestand im Erstellen einer objektorientierten Klassenhierarchie.

Im nächsten Schritt wurde die Aufgabenzerlegung mit den benötigten Datenflüssen festgelegt. Dabei mußten weitere Datenstrukturen für Zwischenergebnisse spezifiziert werden. Durch die graphische Darstellung waren die Datenflüsse und die Aufgabenzerlegung für unseren Experten intuitiv verständlich und ein guter Ausgangspunkt für Diskussionen über die Arbeitsabläufe. Alternative Methoden zur Bearbeitung einer Aufgabe wurden von unseren Experten als selbstverständlich vorausgesetzt, da z. B. fast alle Festsetzungen im Bebauungsplan entweder zeichnerisch oder textlich getroffen werden können. Alternative Methoden waren auch ein Mittel, dem Verhalten von Experten Rechnung zu tragen, sich zunächst auf schwierige Ausnahmefälle zu konzentrieren. Ausnahmen und Regelfälle konnten beide schnell als alternative Methoden modelliert werden. Dadurch wurde der Experte nicht gezwungen, sich auf *genau* eine Vorgehensweise festzulegen.

Trotz der guten Erfahrungen, die wir beim Einsatz von CoMo-Kit gemacht haben, sehen wir die Gefahr, daß die flexiblen Werkzeuge einen ungeübten Benutzer zum schnellen Erstellen unstrukturierter Modelle verführen können.

6. Zusammenfassung & Ausblick

In diesem Beitrag haben wir gezeigt, wie mit Hilfe von Knowledge Engineering Techniken eine komplexe Domäne modelliert werden kann. Dabei sind wir insbesondere auf notwendige Erweiterungen bekannter Ansätze eingegangen, die sich aus einer praxisbezogenen Entwurfsdomäne ergeben. Der beschriebene Ansatz unterstützt die Koordination mehrerer Agenten durch die Verwaltung von Abhängigkeiten.

Das beschriebene Projekt „Intelligenter Bebauungsplan“ integriert GIS/CAD, Hypertext und Expertensystemtechnologie. Die Strukturierung der benötigten Informationen als Hypertext wurde von den beteiligten Stadtplanern gewünscht und ist eine natürliche Repräsentation für Gesetzestexte (und Kommentare), da sie ohnehin viele Querverweise enthalten.

Die *aufgabenorientierte Strukturierung* des Hypertextes im Sinne des hier vorgeschlagenen Modells kann das „Lost-in-Hyperspace“-Problem entschärfen, da im Modell festgelegt werden kann, welche Teile des Gesetzes zur Bearbeitung einer bestimmten Aufgabe betrachtet werden müssen. Dadurch erhält der Planer gezielten Zugriff auf die benötigten Informationen.

Das CoMo-Kit-System mit Modellierungswerkzeugen und Interpreter ist vollständig implementiert. Im Moment arbeiten wir an der Anbindung des GIS und erweitern die Aufgabenmodellierung.

Aktuelle Arbeiten in dem beschriebenen Projektrahmen konzentrieren sich auf zwei Punkte:

- **Methodendefinition während der Laufzeit:** Komplexe Projektabläufe können in der Praxis nicht vollständig vorab modelliert werden. In der Regel muß man während der Projektabwicklung neue Wege (Methoden) entwickeln, eine Aufgabe zu lösen. Um CoMo-Kit zu einem Projektabwicklungswerkzeug weiterzuentwickeln, ist es von daher erforderlich, die Definition neuer Methoden zuzulassen, während das Projekt vom Interpreter abgearbeitet wird.
- **Offene Menge von Unteraufgaben:** In der vorliegenden Version von CoMo-Kit muß im konzeptuellen Modell exakt festgelegt werden, in wieviele Teile eine Aufgabe zerlegt wird. Für unsere Anwendung „Bebauungsplanung“ wäre eine flexiblere Modellierung wünschenswert: Z.B. möchte man beschreiben, daß der Planer im nächsten Schritt

beliebig viele Verkehrsflächen (Straßen etc.) festsetzen kann. In unserer Modellierung bedeutet das, daß die Aufgabe „Verkehrsflächen festsetzen“ in beliebig viele Teilaufgaben „Eine Verkehrsfläche festsetzen“ zerlegbar sein muß.

7. Danksagung

Wir danken Barbara Dellen und Willi Schmitz für fruchtbare Diskussionen und die Implementierung des Interpreters. Thomas Schmidt betreut als Fachexperte für die Raumplanung die Modellierung der Anwendung „Bebauungsplanung“.

8. Literatur

[1] Angele, J.; Fensel, D.; Landes, D.: Modellbasiertes und inkrementelles Knowledge Engineering: der MIKE Ansatz, KI 1/95, interdata Verlag, 1995.

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

[3] Brooks, F.P.: The mythycal man month, Reading MA, Addison-Wesley, 1975

[4] Dellen, B.: Operationalisierung von konzeptuellen Modellen. Diplomarbeit Universität Kaiserslautern, 1995 (in Vorbereitung).

[5] Dellen, B., Maurer, F., Paulokat, J.: Verwaltung von Abhängigkeiten in kooperativen wissensbasierten Arbeitsabläufen. In: Expertensysteme-95, infix Verlag, St. Augustin, 1995.

[6] Doyle, J.: A Truth Maintenance System, Artificial Intelligence, 12:231-272, 1979.

[7] Fensel, D., van Harmelen, F.: A Comparison of Languages which Operationalize and Formalize KADS Models of Expertise, The Knowledge Engineering Review, vol 8, no 2, 1994.

[8] Gappa, U., Puppe, F., Radestock, G.: Generierung graphischer Wissenserwerbssystem für starke Problemlösemethoden, KI 1/95, interdata Verlag, 1995.

[9] Gebhardt, F., Groß, E., Hemmann, Th., Voss, H.: Knowledge Engineering mit MoMo, KI 1/95, interdata Verlag, 1995.

[10] Hangarter, E.: Grundlagen der Bauleitplanung - Der Bebauungsplan, Werner Verlag, 1988

[11] Maurer, F.: Hypermediabasiertes Knowledge Engineering für verteilte wissensbasierte Systeme, Dissertation Universität Kaiserslautern, 1993, auch: DISKI 48, infix-Verlag, ISBN 3-929037-48-3.

[12] Maurer, F., Pews, G.: Validierung von konzeptuellen Modellen. In: F. Puppe, A. Günter (Hrsg.): Expertensysteme 93, Springer, 1993.

[13] Maurer, F., Paulokat, J.: Operationalizing Conceptual Models Based on a Model of Dependencies, in: A. Cohn (Ed.): ECAI 94. 11th European Conference on Artificial Intelligence, 1994, John Wiley & Sons, Ltd.

[14] Petrie, Ch.: Context Maintenance, in: Proceedings of AAAI-91, Menlo Park, California, 1991, MIT Press.

[15] Petrie, Ch.: Planning and Replanning with Reason Maintenance, Dissertation, University of Texas, Austin, 1991.

[16] Pews, G.: Der intelligente Bebauungsplan, URL: http://wwwagr.informatik.uni-kl.de/~pews/ibp.html

[17] Schmitz, W.: Multiple Aufgabenzerlegung von konzeptuellen Modellen, Diplomarbeit an der Universität Kaiserslautern, 1994

[18] Schreiber, G. (Ed.): Special Issue: The KADS Approach to Knowledge Engineering, Knowledge Acquisition, Vol. 4 No. 1, March 1992, Academic Press.

[19] Steels, L.: Components of Expertise, AI Magazine, 11 (2 (Summer)), 1990.

[20] van Harmelen, F., Balder, J.: $(ML)^2$: A Formal Language for KADS Models of Expertise, 1992, in [18].

[21] Wetter, Th.: First-order Logic Foundation of the KADS Conceptual Model, 1990, in [22].

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

[23] Wielinga, B.J.; Schreiber, A.Th.; Breuker, J.A.: KADS: A Modelling Approach to Knowledge Engineering, 1992, in [18].