Modellbasiertes und Inkrementelles Knowledge Engineering: der MIKE Ansatz

Jürgen Angele[1], Dieter Fensel[2], Dieter Landes[3] und Susanne Neubert[4]

[1] Institut für Angewandte Informatik, Fachhochschule Braunschweig, 38302 Wolfenbüttel
e-mail: angele@informatik.fh-wolfenbüttel.de
[2] Institute SWI, Universität Amsterdam, Roeterstraat 15, NL-1018 WB Amsterdam
e-mail: fensel@swi.psy.uva.nl
[3] Institut AIFB, Universität Karlsruhe, 76128 Karlsruhe
e-mail: landes@aifb.uni-karlsruhe.de
[4] software design and management, Thomas-Dehler-Str. 18, 81737 München
e-mail: susanne.neubert@sdm.de

Im folgenden werden die wesentlichen Prinzipien und Methoden des MIKE-Ansatzes (Modellbasiertes Inkrementelles Knowledge Engineering) beschrieben. MIKE bietet eine Sammlung von Methoden zur Erhebung, Interpretation, Formalisierung und Implementierung von Wissen zur Erstellung eines wissensbasierten Systems. Ziel von MIKE ist es, die Vorteile von Lebenszyklusmodellen, Prototyping und formalen Spezifikationssprachen in einem übergreifenden Ansatz für das Knowledge Engineering zu integrieren.

Einleitung

In der “Software-Krise” der sechziger Jahre zeigte sich, daß zur Erstellung großer und komplexer Systeme anstelle der bis dahin praktizierten Methoden bessere, “ingenieurmäßige” Methoden erforderlich waren. Eine ähnliche Entwicklung zeichnete sich in den letzten Jahren auch bei wissensbasierten Systemen ab: die Vorgehensweise zur Entwicklung kleiner prototypischer Expertensysteme läßt sich nicht auf große Systeme übertragen, die längere Zeit benutzt und gewartet werden müssen. Für solche Wissenssysteme sind wiederum methodisch fundierte Enwicklungsansätze notwendig.

Mit MIKE [AFL+93] beschreiben wir in diesem Artikel einen solchen Ansatz. MIKE stellt eine integrierte Sammlung von Methoden zur Erhebung, Interpretation, Formalisierung und Implementierung von Wissen zum Bau wissensbasierter Systems dar. Für die Beschreibung der Resultate der Entwicklungsschritte bietet MIKE jeweils dafür geeignete Beschreibungssprachen an. Die einzelnen Entwicklungsschritte sind in ein Vorgehensmodell eingebettet, das besonderes Augenmerk auf die Integration von Prototyping richtet.

Im folgenden werden zunächst grundlegende Annahmen und Prinzipien von MIKE erläutert, bevor im zweiten Abschnitt die Methoden und Beschreibungssprachen vorgestellt werden. Die Anwendung der Methoden wird im dritten Kapitel anhand des Vorgehensmodells von MIKE aufgezeigt.

1 Prinzipien des MIKE Ansatzes

1.1 Modellbasiertes Knowledge Engineering

Knowledge Engineering als Modellierungsprozeß

Bei der Entwicklung wissensbasierter Systeme soll ein Computermodell entstehen, das in Bezug auf die gestellte Aufgabe ähnliche Lösungen wie der Experte produziert, diesen aber nicht notwendigerweise simuliert. Ein großer Teil des erforderlichen Wissens ist dem Experten nicht bewußt, sondern in seinen Erfahrungen verborgen und deshalb nicht direkt durch Befragung zugänglich. Somit muß die Entwicklung eines solchen Systems nicht als bloßer Transfer von Wissen vom Experten in den Computer, sondern als die Entwicklung eines neuen Computermodells durch den Knowledge Engineer angesehen werden (vgl. z.B. [JIS93]).

Der Knowledge Engineer als Moderator

Wegen des teilweise unbewußten Wissens wird in MIKE ein Knowledge Engineer als Moderator für den Modellierungsprozeß als notwendig erachtet. Ein Erstellen des Modells durch Eingabe von Fakten durch den Experten selbst ist nur für sehr einfache Anwendungsbereiche oder bei starker Unterstützung durch vordefinierte Shells möglich. Problematisch an diesen Shells ist bislang ihre Beschränkung auf einen Anwendungsbereich oder eine spezielle Problemlösungsmethode. Ziel laufender Forschungsarbeiten [PoG93] ist es, die Flexibilität solcher Shells durch Konfiguration aus einfacheren Bausteinen zu erhöhen[1].

1.2 Unterschiedliche Ebenen der Wissensrepräsentation

Zwischen dem Wissen des menschlichen Experten und der Implementierung eines wissensbaiserten Systems klafft eine große Lücke. Indem diese Lücke in mehreren Schritten überbrückt wird (vgl. Bild 1), kann die Komplexität des Entwicklungsprozesses entscheidend verringert werden, weil in jedem Entwicklungsschritt bestimmte Aspekte unabhängig von anderen Aspekten betrachtet werden können.

Die Erstellung und Wartung eines wissensbasierten Systems ist sehr schwierig, wenn das im System enthaltene Wissen nicht auch auf einer von Realisierungsdetails unabhängigen Ebene beschrieben wird, weil diese Details leicht den Blick auf wesentliche Inhalte verstellen. Auch die Erklärungsfähigkeit des Systems kann von einer abstrakteren Beschreibung profitieren, weil das Systemverhalten in einer dem Experten verständlichen Terminologie erläutert werden kann. Eine solche Beschreibung, die sich auf konzeptuelle Aspekte beschränkt, entspricht der Systemspezifikation im Software Engineering. Durch diese Beschreibung auf konzeptueller Ebene, das sog. *Modell der Expertise*, wird die Repräsentationslücke in zwei Abschnitte unterteilt, weil Realisierungsaspekte während der Wissensakquisition zunächst ausgeklammert und erst in der Entwurfs- und Implementierungsphase betrachtet werden können.

Beschreibung auf konzeptueller Ebene

Das Modell der Expertise muß trotz seiner Unabhängigkeit von Realisierungsaspekten eindeutig und präzise beschrieben sein. Die Lücke zwischen der informalen Beschreibung des Wissens und dem Modell der Expertise muß also weiter unterteilt werden.

Interviewtranskripte, Auszüge aus Fachbüchern, Fallbeschreibungen usw. als Ergebnisse der Wissenserhebung werden zunächst im sog. *Erhebungsmodell* [Neu94] durch natürlichsprachliche Protokolle beschrieben, die die Grundlage für die Interpretation und Strukturierung des Wissens sind. Ziel der Interpretation ist es, relevante Konzepte sowie grundlegende Schritte des Experten bei der Problemlösung und ihre Abfolge zu identifizieren und miteinander in

---

1. Zum Vergleich dieses Ansatzes mit MIKE siehe [FeP94].

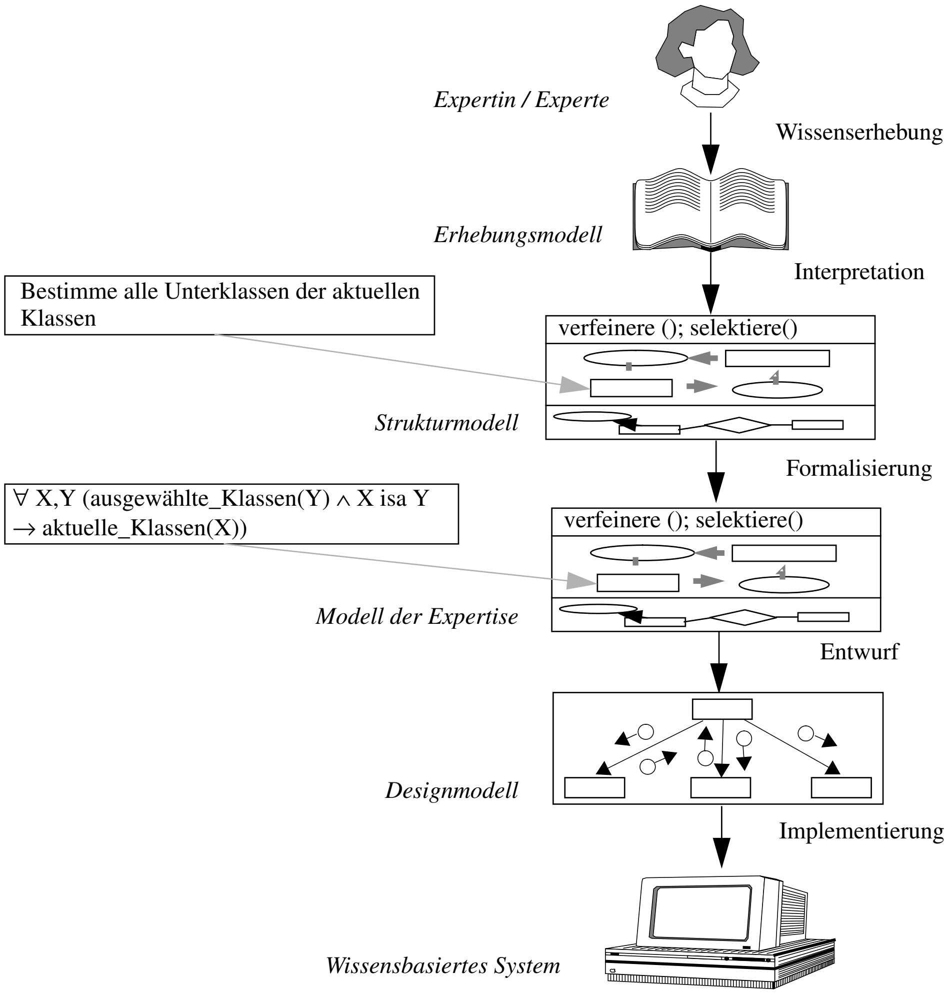

Bild 1 Repräsentationsebenen in MIKE

Bezug zu setzen. Das Resultat dieses Schrittes wird in MIKE im sog. *Strukturmodell* dargestellt [Neu94]. Konzepte und Aktivitäten werden dort mit Hilfe vordefinierter Knotentypen beschrieben und durch Kanten verbunden, die bestimmte Beziehungen zwischen Knoten ausdrücken. Während die Kanten eine formale Bedeutung besitzen, sind die Inhalte der Knoten nach wie vor informal beschrieben. Insgesamt ist das Strukturmodell demnach eine semiformale Repräsentation, die mit dem auf Hypermediatechniken basierenden Werkzeug MeMoKit erstellt und modifiziert wird. Der Strukturierungsprozeß selbst gibt dem Knowledge Engineer und dem Experten eine frühe Rückkopplung. Die semiformale Repräsentation des Wissens stellt eine gute Kommunikationsbasis mit dem Experten dar. Der Inhalt des Strukturmodells kann zur Entwicklung der Erklärungskomponente des Expertensystems benutzt werden. Im Strukturmodell werden zusätzlich Modellierungsentscheidungen dokumentiert, die die spätere Wartung des Systems unterstützen.

Ausgehend von der semiformalen Repräsentation im Strukturmodell wird das Wissen dann formalisiert. Ergebnis des Formalisierungsschrittes ist das Modell der Expertise, zu dessen Beschreibung in MIKE die Sprache KARL ([Ang93], [Fen93]) entwickelt wurde. KARL basiert auf Prädikatenlogik, reichert diese aber um epistemologische Modellierungsprimitive an, die denen des Strukturmodells entsprechen. Somit ergibt sich ein bruchloser Übergang zwischen der semiformalen und der formalen Darstellung.

Die Eindeutigkeit und Präzision einer formalen Beschreibung kann dazu beitragen, einzelne Problemlösungsschritte besser zu verstehen. Zudem kann die formalisierte Problemlösungsmethode den weiteren Akquisitionsprozeß steuern und Grundlage für eine formale Verifikation des Modells sein. Eine formale Beschreibung läßt sich auf eine ausführbare Beschreibung abbilden, so daß die Erstellung und Evaluierung des Modells der Expertise durch Prototyping unterstützt werden kann. In MIKE ist kein expliziter Abbildungsschritt erforderlich, weil für KARL neben einer modelltheoretischen eine operationale Semantik definiert wurde und KARL damit direkt ausführbar ist.

Beschreibung auf realisierungsnaher Ebene

Im Modell der Expertise werden die funktionalen Anforderungen an das zu entwickelnde System beschrieben, zusätzliche nichtfunktionale Anforderungen, z.B. Effizienz der Ausführung, Wartbarkeit etc., die die Realisierung des wissensbasierten Systems betreffen, zunächst jedoch ausgeklammert. Diese Aspekte werden erst in der Entwurfsphase im sog. *Designmodell* beschrieben und umgesetzt. Dazu wurde KARL zur Sprache DesignKARL [Lan94] erweitert, so daß ein bruchloser Übergang von Modell der Expertise zum Designmodell möglich ist. Mit DesignKARL lassen sich Designentscheidungen zu nichtfunktionalen Anforderungen in Bezug setzen und die Auswirkung von Entscheidungen zur Umsetzung dieser Anforderungen, also z.B. der Einsatz geeigneter Algorithmen und Datenstrukturen, beschreiben. Zusätzlich erlaubt DesignKARL, das Modell der Expertise weiter zu strukturieren [LaS94], wobei dessen grundlegende Struktur jedoch erhalten bleiben sollte. Somit kann durch das Designmodell die Lücke zwischen dem Modell der Expertise und der endgültigen Implementierung des Systems weiter unterteilt werden. Die Umsetzung in die Zielumgebung kann für ausgewählte Zielumgebungen durch ein Werkzeug unterstützt werden, das Teile des Designmodells automatisch auf Codestücke für die entsprechende Zielumgebung abbildet.

Die unterschiedlichen Modelle, in denen das Wissen repräsentiert wird, beschreiben entweder das gleiche Wissen auf eine andere Art oder enthalten zusätzliches Wissen, das aber in enger Verbindung zu Wissen in anderen Modellen steht. Um diese Bezüge für Dokumentation, Wartung und Erklärungsfähigkeit nutzen zu können, werden diese Querbezüge explizit repräsentiert.

1.3 Wiederverwendung

Um den Entwicklungsaufwand zu reduzieren und die Qualität des wissensbasierten Systems sicherzustellen, sollten Teile von Modellen wiederverwendbar sein. Im Modell der Expertise wird Problemlösungswissen unabhängig von konkretem Anwendungswissen beschrieben und kann deshalb für eine ähnliche Aufgabe in einem anderen Anwendungsbereich genutzt werden. Umgekehrt können Teile des anwendungsspezifischen Wissens für andere Aufgaben wiederverwendet werden. MIKE bietet eine Bibliothek sowohl semiformal, als auch formal beschriebener Problemlösungsmethoden; die Auswahl einer geeigneten Methode wird durch das Werkzeug MeMoKit unterstützt.

2 Methoden und Beschreibungssprachen

MIKE bietet ein Spektrum von Modellen und Beschreibungssprachen mit unterschiedlichen Schwerpunkten. Das Erhebungs- und Strukturmodell repräsentiert Wissen in der frühen Akquisitionsphase informal oder semiformal. Die Sprache KARL wurde zur formalen und ausführbaren Beschreibung des Modells der Expertise entwickelt. Das Designmodell und DesignKARL integrieren nichtfunktionale Anforderungen und erleichtern den Übergang vom Modell der Expertise zum endgültigen System.

2.1 Das Erhebungs- und Strukturmodell

Das Erhebungs- und Strukturmodell [Neu93] beschreibt Wissen semiformal in Form von Protokollen, Konzepten und Aktivitäten und stellt den Datenfluß und den Kontrollfluß zwischen Aktivitäten dar. Eine solche semiformale Repräsentation erlaubt, den Experten in den Strukturierungsprozeß komplexen Wissens zu integrieren. Somit wird die Kooperation zwischen Experten und Knowledge Engineer verbessert und die Formalisierung vereinfacht.

Das Erhebungs- und Strukturmodell besteht aus speziellen Knoten- und Kantentypen. Ein Knoten stellt ein Hypermediadokument dar, das einen Zustand, einen Prozeß oder ein Konzept ggf. multimedial beschreibt. Eine Kante stellt eine Beziehung zwischen zwei Knoten dar und wird durch Startknoten, Endknoten, Namen und ein Erklärungsfeld definiert.

Das Erhebungsmodell repräsentiert das Ergebnis der Wissenserhebung und enthält informale Wissensprotokolle, die in sog. Protokollknoten gespeichert werden.

Das Strukturmodell wird aufbauend auf den Protokollen erstellt. Um über die unterschiedlichen Wissensrepräsentationen hinweg weitgehend die gleiche Strukturierung zu erhalten, bietet das Strukturmodell Beschreibungselemente, die leicht auf entsprechende Modellierungsprimitive des Modells der Expertise in der formalen Sprache KARL abgebildet werden können (vgl. Bild 2):

- Der Aktivitätenkontext enthält Knoten, die jeweils einen Schritt innerhalb des Problemlösungsprozesses beschreiben. Aktivitäten können mit Verfeinerungskanten hierarchisch verfeinert werden. Der oberste Aktivitätenknoten stellt die gesamte Problemlösungsmethode dar.
- Der Reihenfolgekontext beschreibt die zeitliche Abfolge der Aktivitäten durch Reihenfolgekanten.
- Der Konzeptkontext beschreibt alle Konzepte. Konzeptknoten können durch is-a Kanten, part-of Kanten oder Kanten, die beliebige Beziehungen beschreiben, verbunden sein.
- Der Strukturkontext beschreibt den Datenfluß zwischen Aktivitäten einer Hierarchiestufe, indem Aktivitätenknoten durch sog. Datenflußkanten verbunden werden.

Das Werkzeug MeMoKit unterstützt die Entwicklung des Erhebungs- und Strukturmodells durch graphische Editoren und durch eine graphische Auswahl von Problemlösungsmethoden.

2.2 Das Modell der Expertise in KARL

Das konzeptuelle Modell, auf dem KARL ([Ang93], [Fen93]) basiert, wurde aus dem KADS Modell der Expertise [SWB93] abgeleitet und unterscheidet mehrere Arten von Wissen.

Domänenwissen besteht aus Wissen über den Anwendungsbereich des Systems, definiert eine Konzeptualisierung des Anwendungsbereichs und beschreibt die deklarative Theorie, die das zur Problemlösung nötige Wissen enthält. Zu dessen Beschreibung integriert KARL Frames und Logik in der Teilsprache L-KARL. Terminologisches Wissen wird durch eine Konzepttaxonomie beschrieben. Fakten und regelhaftes Wissen werden in logischen Formeln ausgedrückt, in denen auf Instanzen der Taxonomie, auf deren Attributwerte und auf die Taxonomie selbst Bezug genommen werden kann.

Inferenzwissen definiert sowohl die einzelnen Inferenzen (Inferenzaktionen), die während des Problemlösungsprozesses gezogen werden, als auch sog. Rollen, die Ein- und Ausgaben der Inferenzen enthalten. Es werden drei Arten von Rollen unterschieden: Views liefern Domänenwissen für eine Inferenzaktion, Stores modellieren den Datenfluß zwischen verschiedenen

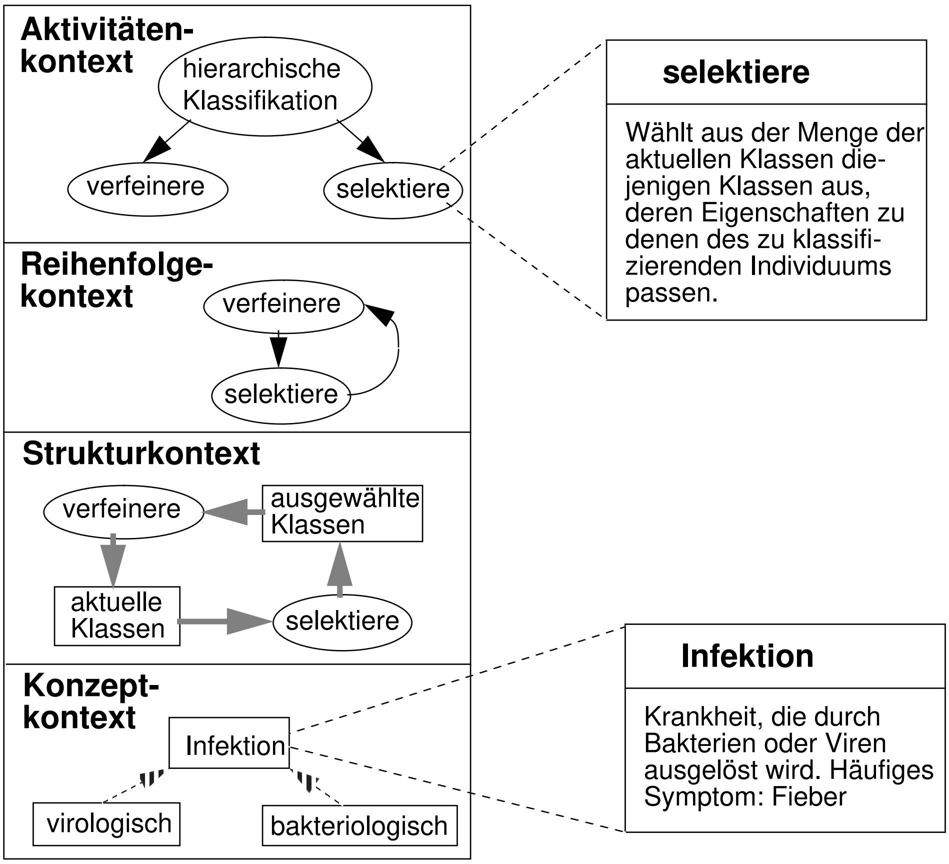

Bild 2 Beispiel eines Strukturmodells für die hierarchische Klassifikation von Krankheiten

Inferenzaktionen, Terminatoren erlauben Ergebnisse des Problemlösungsprozesses zurück auf die Domänenebene zu schreiben. L-KARL wird zur Spezifikation der Inferenzaktionen, zur Definition einer generischen problemlösungsmethodenspezifischen Terminologie innerhalb der Rollen und Inferenzaktionen und zur Abbildung zwischen dieser Terminologie und der anwendungsspezifischen Terminologie der Domänenebene benutzt.

Auf der Taskebene wird mit der Teilsprache P-KARL die Reihenfolge beschrieben, in der Inferenzaktionen auf der Inferenzebene aktiviert werden. P-KARL beinhaltet Schleife, Alternative und Sequenz als Kontrollkonstrukte.

Wissen auf der Inferenz- und Taskebene ist anwendungsunabhängig und beschreibt den Problemlösungsprozeß somit generisch.

2.3 Die Sprache KARL

KARL besteht aus den beiden Teilsprachen L-KARL und P-KARL. L-KARL dient zur darstellung (statischen) strukturellen Wissens und wurde auf Basis von F-Logik [KLW93] entwickelt. P-KARL dient zur Darstellung dynamischen (prozeduralen) Wissens und wurde auf der Basis von dynamischer Logik entworfen.

F-Logik und L-KARL bieten im Vergleich zu Prädikatenlogik zusätzliche Modellierungsprimitive, die zur Darstellung vieler Sachverhalte geeigneter sind, ohne jedoch die modelltheoretische Semantik von Prädikatenlogik zu verlassen. KARL integriert Ideen aus semantischen und objektorientierten Datenmodellen in einen logischen Ansatz.

Eine einfache Anweisung in P-KARL besteht aus dem Aufruf einer Inferenzaktion. An komplexen Anweisungen existieren zwei Schleifenvarianten, Verzweigung und Sequenz. Anweisungen können zu Subtasks zusammengefaßt werden. Der Aufruf der beiden Inferenzaktionen *verfeinere* und *selektiere* (vgl. Reihenfolgekontext in Bild 2) erfolgt in einer Schleife:

```
WHILE NOT EMPTY (aktuelle_Klassen)
    verfeinere
    selektiere
```

Solche Ausdrücke lassen sich unmittelbar in dynamische Logik übersetzten und erhalten so eine deklarative Interpretation.

Zur Erhöhung der Verständlichkeit und somit zur Unterstützung der Kommunikation mit dem Experten existieren für die meisten Modellierungsprimitive graphische Repräsentationen. Zur Darstellung der Domänenebene wird eine Form von Enhanced-Entity-Relationship (EER) [EIN89] Diagrammen verwendet. Zur Beschreibung der Input-/Outputbeziehungen der Inferenzaktionen werden Datenflußdiagramme [You89] eingesetzt. Die Abfolge der Aktivierung von Inferenzaktionen wird durch Programmflußdiagramme beschrieben. Alle graphischen Repräsentationen sind hierarchisch verfeinerbar.

In [Fen93] ist die Semantik von KARL modelltheoretisch und in [Ang93] operational definiert. Basierend auf dieser Semantik und optimierten Auswertungsalgorithmen wurde für KARL ein Interpreter mit einem integrierten Debugger entwickelt, so daß eine KARL-Spezifikation durch Testen validiert werden kann.

2.4 Das Designmodell

Das Modell der Expertise als Resultat der Analysephase in MIKE konzentriert sich auf die konzeptuelle Beschreibung des zur Problemlösung erforderlichen Wissens sowie der verwendeten Problemlösungsmethode. Nichtfunktionale Aspekte wie Effizienz der Realisierung, Wartungsfreundlichkeit usw., aber auch Einschränkungen durch die intendierte Zielumgebung werden während der Analysephase weitgehend außer Acht gelassen und sind im Modell der Expertise nicht erfaßt. Dieser Typ von Anforderungen wird erst in der Entwurfsphase genauer betrachtet, da Designentscheidungen vorrangig auf solche Anforderungen zurückzuführen sind.

Um u.a. die spätere Wartung des wissensbasierten Systems zu erleichtern, ist es erforderlich, im Designmodell nicht nur das Designprodukt als das Resultat des Designprozesses zu beschreiben, das als Grundlage für die eigentliche Implementierung dient, sondern auch die nichtfunktionalen Anforderungen und ihre Bezüge zu Designentscheidungen aufzuzeichnen, also auch Aspekte des Entwurfsprozesses zu erfassen.

Designentscheidungen im Zusammenhang mit nichtfunktionalen Anforderungen erfordern häufig die Einführung geeigneter Datenstrukturen und die Entwicklung effizienter Algorithmen sowie die Strukturierung des wissensbasierten Systems. Um diese Aspekte erfassen zu können, muß der Beschreibungsformalismus erweitert werden: DesignKARL [Lan94] ergänzt KARL u.a. um die Möglichkeit, Datenstrukturen und Algorithmen zu beschreiben und bietet zusätzliche Strukturierungprimitive wie Cluster und Module. Die Sprachmittel von KARL können in DesignKARL jedoch nach wie vor verwendet werden.

Daneben gestattet es DesignKARL, außer dem Designprodukt den Designprozeß selbst zu beschreiben, indem Sprachprimitive zur Verfügung stehen, mit denen Entwurfsentscheidungen dokumentiert und in Bezug zu den betroffenen Teilen des Designproduktes einerseits und den sie motivierenden nichtfunktionalen Anforderungen andererseits gesetzt werden können. Weiterhin besteht die Möglichkeit, Interaktionen zwischen Designentscheidungen (z.B. gegenseitiger Ausschluß oder Implikation) sowie Anforderungen (z.B. positive oder negative Korrelation) zu beschreiben.

Neben dem Vorteil, durch die Einbindung dieser Aspekte in das Designmodell die Wartung des Systems zu unterstützen, weil frühere Designentscheidungen gut dokumentiert sind, ist so auch nachvollziehbar, daß und wie sowohl funktionale als auch nichtfunktionale Anforderungen sich im Designmodell und der Implementierung niederschlagen.

3 Das Prozeßmodell von MIKE

Für MIKE ist in [Ang93] und [Neu93] ein Prozeßmodell beschrieben, das den Knowledge Engineer im gesamten Entwicklungsprozeß leitet. Das Prozeßmodell gibt bestimmte Aktivitäten und die Art der Abschlußdokumente vor; es beschreibt die Reihenfolge der Aktivitäten und die Kriterien zum Übergang in die nächsten Aktivität. Der gesamte Knowledge Engineering Prozeß unterteilt sich mehrere Phasen: die Wissensakquisition, die Entwurfsphase, die Implementierungsphase und die Evaluierungsphase. Diese Aktivitäten werden in dieser Reihenfolge zyklisch durchlaufen (vgl. [Boe88]). Jeder Durchlauf erzeugt einen Prototyp, der sich sukzessive an die gestellten Anforderungen annähert. Das endgültige System wird also durch evolutionäres Prototyping [Flo84] erstellt.

Die Wissensakquisition wird in die Aufgabenanalyse, die Modellbildung und die Modellevaluierung unterteilt. In der Aufgabenanalyse erfolgt die hierarchische Dekomposition der Aufgabe und die Beschreibung der Abhängigkeiten zwischen den Teilaufgaben. Zudem erfolgt die Auswahl der Teilaufgabe, die als nächstes angegangen werden soll. Die Modellbildung besteht aus Wissenserhebung, Interpretation, Formalisierung und Operationalisierung. In der Wissenserhebung erfolgt die Befragung des Experten und Erhebung mittels Literatur. Das Ergebnis ist eine Menge von Protokollen und wird im MIKE im Erhebungsmodell repräsentiert. Im Interpretationsschritt erfolgt die Konzeptualisierung und Strukturierung des Wissens und dessen semiformale Beschreibung im Strukturmodell. In einem Formalisierungsschritt wird das Strukturmodell in das Modell der Expertise in KARL überführt. Dieses Modell wird dann geeignet eingeschränkt, um ein operationales Modell zu erhalten. Das ausführbare Modell kann jetzt im Evaluierungsschritt gegenüber den Anforderungen des Experten und des Anwenders durch Testen anhand konkreter Fallbeispiele validiert werden. Die Schritte innerhalb der Wissensakquisition werden zyklisch durchlaufen, bis ein adäquates Modell entstanden ist. Die Entwicklung dieses Modell erfolgt also durch exploratives Prototyping [Flo84].

In der Entwurfsphase erfolgt zusätzlich die Erfassung, Beschreibung und Umsetzung nichtfunktionaler Anforderungen wie Effizienz der Ausführung, Wartbarkeit etc. Das Designmodell als Ergebnis dieses Schrittes wird mit DesignKARL beschrieben. Die Struktur des Modells der Expertise sollte im Designmodell weitgehend erhalten werden. Auch die Entwurfsphase wird in eine Anforderungsanalyse, eine Modellbildungsphase und eine Evaluierungsphase unterteilt, die zyklisch durchlaufen werden, und beinhaltet die Möglichkeit der Evaluierung von Designentscheidungen durch experimentelles Prototyping.

Nach der Entwurfsphase erfolgt die Implementierung des Systems durch eine teilautomatische Abbildung des Designmodells in eine Implementierung des Systems in C++.

4 Vergleich zu anderen Arbeiten

Eine Reihe weiterer Ansätze weist in vielen Punkten Ähnlichkeiten zu MIKE auf.

MIKE stützt sich auf KADS-I [SWB93] als Ausgangsbasis. Daher sind die Strukturen des Modells der Expertise nahezu identisch. MIKE legt jedoch mehr Gewicht auf die Formalisierung dieses Modells als KADS-I und integriert zusätzlich Prototyping in sein Lebenszyklusmodell. Obwohl Ähnlichkeiten der frühen Modelle von MIKE mit Modellen von KADS vorhanden sind, wird in KADS der fließende Übergang von informalen Repräsentationen bis zum endgültigen System weniger konsequent verfolgt.

Im Components of Expertise Ansatz [Ste93] wird das zu entwickelnde System aus anderen Bausteinen konfiguriert als in MIKE. Die Anwendung wird hinsichtlich der Aspekte Teilaufgaben, Informationsquellen und Methoden analysiert. Die entstehenden Strukturen ähneln denen des Strukturmodells in MIKE. Dieser Ansatz beschreibt Wissen allerdings nur semiformal und nicht formal wie in MIKE. Auch hier wird der fließende Übergang von informalen Beschreibungen zum endgültigen System nicht unterstützt.

Auch im Software Engineering oder im Information Systems Engineering existieren Ansätze, die das zu entwickelnde System in unterschiedlichen Repräsentationen beschreiben.

In der Strukturierten Analyse [You89] oder in der objektorientierten Methode OMT [RBP+91] wird das zu entwickelnde System aus unterschiedlichen Blickwinkeln betrachtet. Die funktionsorientierte Sicht betrachtet das System als einen Fluß von Daten zwischen Prozessen. Diese Sicht ähnelt somit der Inferenzebene. Die dynamische Sicht ähnelt der Beschreibung auf der Taskebene. Die Objektmodellierung entspricht der Modellierung statischen Wissens auf der Domänenebene. Allerdings erfolgt diese Beschreibung in SA oder OMT nur semiformal und ist deshalb mit dem Strukturmodell vergleichbar. Zwischen generischem und anwendungsspezifischem Wissen wird nicht unterschieden. In [FAL+93] wird gezeigt, wie KARL auch zur Formalisierung innerhalb der Strukturierten Analyse eingesetzt werden kann.

Im INCOME-Ansatz ([LNO+89], [NOS+92]) zur Entwicklung von Informationssystemen werden Glossare und Objektflußdiagramme zur semiformalen Beschreibung statischer und dynamischer Aspekte benutzt. Diese Beschreibung wird dann mit Hilfe von Petrinetzen und semantischen Datenmodellen formalisiert. Somit entspricht die erste Beschreibung dem Strukturmodell und die formalisierte Beschreibung dem Modell der Expertise in MIKE. Auch in INCOME wird nicht zwischen generischen und domänenspezifischen Teilen unterschieden. Außerdem werden durch Petrinetze Datenfluß und Kontrollfluß gemeinsam beschrieben. In [AFL92] findet sich ein detaillierterer Vergleich zwischen INCOME und MIKE.

5 Schluß

In MIKE erfolgt der Übergang von den informalen Wissensprotokollen zum endgültigen System sanft in mehreren aufeinanderfolgenden Schritten, was die Komplexität jedes einzelnen Schrittes entscheidend verringert. Die informaleren Repräsentationen eignen sich gut für die Kommunikation mit dem Experten und zur Dokumentation. Die formalen Repräsentationen beschreiben Wissen präzise unter Abstraktion von Implementierungsdetails. Die Verwendung generischer Teile in Form von Problemlösungsmethoden kann den Entwicklungsprozeß verkürzen und den gesamten Akquisitionsprozeß steuern. Die einzelnen Phasen des Entwicklungsprozesses werden durch eine Reihe von Werkzeugen unterstützt.

MIKE wurde inzwischen in verschiedenen Fallstudien sehr erfolgreich eingesetzt. Die größte Fallstudie war die Entwicklung eines Aufzugkonfigurationssystems [PFL+93]. Bei solchen Fallstudien hat sich allerdings auch gezeigt, daß vielfach Problemlösungen in stärkerer Kooperation mit dem Anwender erfolgen müssen. Die Sprache KARL, die den ersten ausführbaren Prototyp innerhalb des Akquisitionsprozesses beschreibt, bietet jedoch nur sehr rudimentäre Möglichkeiten zur Interaktion an.

Für die Sprache KARL wurde eine formale Semantik beschrieben. In Zukunft sollen darauf basierende Verifikationstechniken entwickelt werden.

Weiterhin sollen Aspekte der Wiederverwendbarkeit sowohl von Problemlösungsmethoden als auch von Domänenwissen [Pir93] noch genauer untersucht werden.

Momentan wird eine Werkzeugunterstützung für die Designphase entwickelt. Das Designmodell soll dabei ebenso zu dessen Validierung ausführbar sein.

Danksagung

Wir danken Rudi Studer für die vielfältige Unterstützung und Ideen von seiner Seite.

Literatur

[ABG+93] N. Aussenac, G. Boy, B. Gaines, M. Linster, J.-G. Ganascia und Y. Kodratoff (eds.): *Knowledge Acquisition for Knowledge-Based Systems*. Lecture Notes in Artificial Intelligence 723, Springer, Berlin, 1993

[AFL92] J. Angele, D. Fensel und D. Landes: Two languages to do the same? In R. Studer (ed.): *Informationssysteme und Künstliche Intelligenz*. Informatik- Fachberichte 303, Springer, 1992, 23-39

[AFL+93] J. Angele, D. Fensel, D. Landes, S. Neubert und R. Studer: Model-based and incremental knowledge engineering: the MIKE approach. In J. Cuena (ed.): *Knowledge Oriented Software Design.* North Holland, Amsterdam, 1993, 139-168

[Ang93] J. Angele: *Operationalisierung des Modells der Expertise mit KARL.* Dissertation, infix, St. Augustin, 1993

[Boe88] B.W. Boehm: A spiral model of software development and enhancement. In *ACM SIGSOFT 11(4)*, 1988, 21-42

[EIN89] R. Elmasri und S.B. Navathe: *Fundamentals of Database Systems*. Benjamin/Cummings, Houston, 1989

[FAL+93] D. Fensel, J. Angele, D. Landes und R. Studer: Giving structured analysis techniques a formal and operational semantics with KARL. In H. Züllighoven et al. (eds.)*: Requirements Engineering '93: Prototyping*. Berichte des German Chapter of the ACM Band 41, Teubner, Stuttgart, 1993, 267-285

[Fen93] D. Fensel: *The Knowledge Acquisition and Representation Language KARL.* Dissertation, Universität Karlsruhe, 1993. Erscheint bei Kluwer, Boston, 1995

[FeP94] D. Fensel und K. Poeck: A comparison of two approaches to model-based knowledge acquisition. In L. Steels et al. (eds.): *A Future for Knowledge Acquisition*. Lecture Notes in Artificial Intelligence 867, Springer, Berlin, 1994, 46-62

[Flo84] C. Floyd: A systematic look at prototyping. In R. Budde et.al. (eds.): *Approaches to Prototyping*, Springer, Berlin, 1984, 1-18

[JIS93] Sonderheft "Knowledge Acquisition as Modelling", Teil I und II, *International Journal of Intelligent Systems 8(1) bzw. 8(2)*, 1993

[KLW93] M. Kifer, G. Lausen und J. Wu: Logical foundations of object-oriented and frame-based languages. Forschungsbericht 93/06, Department of Computer Science, SUNY at Stony Brook, NY, April 1993. Erscheint in *Journal of the ACM*

[Lan94] D. Landes: DesignKARL - A language for the design of knowledge-based systems. In *Proc. 6th International Conference on Software Engineering and Knowledge Engineering SEKE'94*, Jurmala, Lettland, 1994, 78-85

[LaS94] D. Landes und R. Studer: Mechanisms for structuring knowledge-based systems. In D. Karagiannis (ed.): *Database and Expert Systems Applications*. Lecture Notes in Computer Science 856, Springer, Berlin, 1994, 488-497

[LNO+89] G. Lausen, T. Németh, A. Oberweis, F. Schönthaler und W. Stucky: The INCOME approach for conceptual modelling and prototyping of information systems. In *Proc. 1st Nordic Conference on Advanced Systems Engineering CASE'89*, Stockholm, Schweden, 1989

[Neu93] S. Neubert: Model construction in MIKE (Model Based and Incremental Knowledge Engineering). In [ABG+93], 200-219

[Neu94] S. Neubert: *Modellkonstruktion in MIKE - Methoden und Werkzeuge*. Dissertation, infix, St. Augustin, 1994

[NOS+92] T. Németh, A. Oberweis, F. Schönthaler und W. Stucky: INCOME: Arbeitsplatz für den Programmentwurf interaktiver betrieblicher Informationssysteme. Forschungsbericht 251, Institut für Angewandte Informatik und Formale Beschreibungsverfahren, Universität Karlsruhe, 1992

[PFL+93] K. Poeck, D. Fensel, D. Landes und J. Angele: Combining KARL and configurable role limiting methods for configuring elevator systems. In *Proc. 8th Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW´94)*, Banff, Kanada, 1994

[Pir93] T. Pirlein: Reusing a large domain-independent knowledge base. In *Proc. 5th International Conference on Software Engineering and Knowledge Engineering SEKE'93*, San Francisco, USA, 1993, 474-483

[PoG93] K. Poeck und U. Gappa: Making role-limiting methods more flexible. In [ABG+93], 103-122

[RBP+91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy und W. Lorensen: *Object-Oriented Modeling and Design*. Prentice Hall, 1991

[Ste93] L. Steels: The componential framework and its role in reusability. In J.-M. David et al. (eds.): *Second Generation Expert Systems*. Springer, Berlin, 1993, 273-298

[SWB93] G. Schreiber, B. Wielinga und J. Breuker (eds.): *KADS, A Principled Approach to Knowledge-Based System Development*. Academic Press, London, 1993

[You89] E. Yourdon: *Modern Structured Analysis*. Prentice-Hall, Englewood Cliffs, 1989