In Künstliche Intelligenz, no 4, 1994

Sinn und Unsinn formaler Spezifikationssprachen für wissensbasierte Systeme

Dieter Fensel
Department SWI, University of Amsterdam, 1018 πWB Amsterdam
e-mail: dieter@swi.psy.uva.nl

In den siebziger und frühen achtziger Jahren wurde im Software Engineering (SE) eine Debatte geführt, die nun mit Zeitverzögerung auch im Knowledge Engineering (KE) geführt wird. Es geht dabei um den Sinn und Unsinn formaler Spezifikationstechniken zur implementierungsunabhängigen Beschreibung von Softwaresystemen. Im KE-Bereich geht es dabei speziell um wissensbasierte Systeme bzw. Expertensysteme. Im folgenden soll zunächst kurz die Historie zusammengefaßt werden, die zur Entwicklung formaler Spezifikationstechniken im KE geführt hat. Damit werden zugleich die Gründe genannt, die die Entwicklung und die Verwendung formaler Spezifikationssprachen rechtfertigen. Danach werden einige Einwände gegen formale Spezifikationstechniken diskutiert, die Situation mit der im SE verglichen und im Anschluß versucht, einen versöhnenden Standpunkt aufzuzeigen.

1. Was führte zur Entwicklung formaler Wissensspezifikationssprachen?

Ein *Expertensystem* oder *wissensbasiertes System* ist ein Computerprogramm, das Aufgaben löst, für die

- (1) ein Mensch ein hohes Maß an Wissen und intellektuellen Fähigkeiten benötigt und
- (2) dieses *Wissen notwendig* zur Lösung der Aufgaben ist.[1]

Die letztere Bedingung fordert dabei, daß das Problem z.B. nicht einfach durch eine vollständige Suche im Raum aller möglichen Lösungen effektiv und effizient gelöst werden kann. Die Notwendigkeit, aufgaben- bzw. bereichsspezifischen Wissens zur effektiven und effizienten Problemlösung soll also eine inhärente Eigenschaft des Problems sein und nicht nur bei dessen Lösung durch einen menschlichen Problemlöser benötigt werden. Der Prozeß zur Erhebung, Interpretation und Beschreibung des Wissens zum Bau wissensbasierter Systeme wird *Wissensakquisition* genannt. Ursprünglich wurde Wissensakquisition wesentlich als Transferprozeß von bereits existierendem Wissen aufgefaßt:

> *"... transfer and transformation of problem-solving expertise from a knowledge source to a program is the heart of the expert-system development process." [HWL83]*

Wissen wurde als fixierte und objektiv bereits gegebene Größe aufgefaßt, welches die WissensingeneurIn aus dem Kopf der ExpertIn herauszufischen oder, noch besser, die die ExpertIn durch graphische Wissenseditoren direkt in das System einzugeben habe. Sehr schnell tauchte jedoch die Metapher des *knowledge acquisition bottleneck* auf, um die unerwarteten Probleme bei der Entwicklung wissensbasierter Systeme zu beschreiben. Einige seien hier beispielhaft aufgelistet (siehe [Ber87]): Die kognitiven Grundlagen menschlicher Problemlösefähigkeit sind ungenügend verstanden. Menschliche Expertise basiert auf unbewußtem Wissen bzw. Können und Fähigkeiten. Menschliches Wissen ist oft schwer zu formalisieren und extrem kontextsensibel. Experten neigen dazu, ihr eigenes Problemlösungsverhalten zu rationalisieren, und liefern so falsche bzw. unzureichende

1. Eine andere (soziologisch motivierte) Definition für Expertensysteme ist die folgende: *Expertensystem* ist der Name für ein Anwenderprogramm, das von Entwicklern erstellt wurde, die sich selbst der Gruppe der im Bereich Künstliche Intelligenz Tätigen zurechnen.

Begründungen. Experten sind Experten in der Lösung bestimmter Probleme, aber selten Experten in der Erklärung des Lösungsweges.

Alle diese “verschiedenen” Probleme sind Indikatoren für eine wesentliche Eigenschaft des Wissensgewinnungsprozesses: Wissensakquisition ist nicht ein Transferprozeß bereits gegebenen Wissens in ein Computerprogramm, sondern ein Prozeß der *Wissenserzeugung*.[2] Ein *Modell der Expertise*, d.h. der Problemlösung und des dabei verwendeten Wissens, wird während der Wissensakquisitionsphase *erzeugt* (vgl. [Mor87], [Fen92], [FBA+93]). Dieses Modell unterscheidet sich dabei signifikant von der kognitiven Basis, die der Problemlösung durch den menschlichen Experten zugrunde liegt. Jeder Modellierungsprozeß hat einige sehr unangenehme Eigenschaften: Er ist nur schrittweise möglich, bleibt immer unabgeschlossen, liefert nur Näherungslösungen und ist fehlerbehaftet. Das Modell ist dabei immer ebensosehr Resultat des gegenwärtigen Stands des Modellierungsprozesses, wie es in ihn selbst als Voraussetzung für die weitere Modellierung wieder eingeht.[3]

Eine frühe Antwort auf diesen zyklischen Charakter der Wissensgewinnung war die unmittelbare Implementierung des gewonnenen Wissens in einem Computerprogramm. Das laufende Programm kann dazu benutzt werden, das modellierte Wissens zu evaluieren. Diese naturwüchsig entstandene Entwicklungsmethode reflektiert den zyklischen Charakter des Modellierungsprozesses. Das laufende System gibt unmittelbare Rückkopplung für den Wissensakquisitionsprozeß und steuert den Prozeß der Revision und Verfeinerung des modellierten Wissens (vgl. [Bra89], [GuD89], [MBG90], [Par86], [Weu93]). Jedoch hat diese “Methode” einige signifikante Probleme: Die WissensingeneurIn hat eine Vielzahl verschiedener Tätigkeiten gleichzeitig durchzuführen. Sie muß die gegebene Information analysieren und zugleich das System entwerfen und implementieren. Das laufende System ist meist die einzige Dokumentation des Wissens und der gewählte Implementierungsformalismus bestimmt bzw. beschränkt die Sicht auf das zu modellierende Wissen, und Implementierungsdetails werden mit Aspekten des konzeptuellen Modells vermischt. *Das wesentliche Problem dieser Methode ist dabei die Vermischung zweier deutlich verschiedener Aspekte des Entwurfsprozesses*: Die Modellierung der Aufgabe, die das System lösen soll, und des dafür benötigten Wissens einerseits und andererseits der Entwurf eines spezifischen Softwareartefakts, der diese Aufgabe unter Verwendung des Wissens effizient löst. Folgen des unstrukturierten Entwicklungsprozesses sind zum einen die Unkalkulierbarkeit von Entwicklungskosten und -zeit und andererseits die Nichtplanbarkeit der Systemarchitektur, die sich bottom-up als Konglomerat von inkrementell hinzugefügten Systemfunktionalitäten und Verbesserungen schon realisierter Funktionalitäten ergibt. Im Ergebnis konnten mit dieser Entwicklungsmethode zwar schnell vielversprechende und eindrucksvolle Prototypen und kleine Systeme entwickelt werden. Andererseits scheiterten viele Expertensystemprojekte, wenn die Systeme eine bestimmte Größe überschritten bzw. über einen längeren Zeitraum hinweg im Einsatz waren und gewartet werden mußten.[4]

Diese Situation Ende der achtziger Jahre entsprach der Situation im Bereich der konventionellen Softwareentwicklung Ende der sechziger Jahre. Mit dem Begriff *Softwarekrise* wurde dort eine analoge Situation beschrieben. Programme, die eine bestimmte Größe überschritten, erwiesen sich als unkalkulierbar in ihren Entwicklungskosten, Entwicklungszeiten und Wartungskosten. Als Reaktion hierauf wurde das Fachgebiet Software Engineering (SE) in Leben gerufen und die Entwicklung von strukturierten Methoden, Vorgehensmodellen und Werkzeugen (z.B. formale Spezifikationssprachen) angestoßen, die die Entwicklung großer, zuverlässiger und wartbarer Software ermöglichen sollen (siehe beliebige Standardlehrbücher zu SE).

---

2. Wissen wird nicht abgebildet sondern konstruiert. Dieser *konstruktivistische* Blickwinkel kann natürlich auch für die Entwicklung konventioneller Software bzw. von Software überhaupt eingenommen werden, vgl. [Nau85] und [FZB+92].

3. Ein analoges Problem reflektiert z.B. der sogenannte hermeneutische Zirkel der Textinterpretation.

4. [GuD89], [MBG90], [Weu93].

In den letzten Jahren wurden ähnliche Anstrengungen im KE unternommen. Fundamental hierfür war die von [New82] vorgenommene Unterscheidung von *symbol level* und *knowledge level*. Auf dem sogenannten knowledge level wird Expertise implementierungsunabhängig in Form von Zielen, Operationen und Wissen, welches den Zusammenhang von Zielen und Operationen enthält, beschrieben. Erst auf dem symbol level wird eine spezielle berechenbare AgentIn implementiert, die die Problemlösung als Computerprogramm realisiert. Aus der Vielzahl von Ansätzen im KE (siehe [DKS93] für einen aktuellen Überblick) ist vor allem das KADS Projekt (KADS-I bzw. CommonKADS) [SWB93] hervorzuheben. Während sich die meisten anderen Ansätze in erster Linie mit der Entwicklung von Werkzeugen für einzelne Teilaspekte des gesamten Entwicklungsprozesses beschäftigen oder auf bestimmte Problemtypen beschränken, wurde im KADS Projekt eine umfassende Methode entwickelt, die die meisten Phasen und Aspekte des Entwicklungsprozesses wissensbasierter Systeme umfaßt und viele andere Ansätze als Spezialfälle enthält. Eines der wichtigsten Ergebnisse des KADS-I Projektes ist die Einführung strukturierter Lebenszyklusmodelle als Vorgehensmodelle für die Systementwicklung und hierbei besonders die Trennung der Wissensakquisition von der Design- und Implementierungsphase. Das akquirierte Wissen wird nicht mehr unmittelbar durch ein Computerprogramm beschrieben. Das *Modell der Expertise* à la KADS erlaubt die implementierungsunabhängige Beschreibung des Wissens und erfüllt eine analoge Aufgabe zum konzeptuellen Modell eines konventionellen Programms, wie es als Resultat der Spezifikationsphase entsteht. Das Modell der Expertise unterscheidet verschiedene Wissensarten, die auf getrennten Ebenen dargestellt werden. Für jede diese Ebenen wurden wiederum geeignete Primitive zur Darstellung des Wissens entwickelt. Auf dem *domain layer* wird vergleichbar mit einer deduktiven Datenbank eine deklarative, d.h. aufgabenunabhängige, Beschreibung des Wissens der ExpertIn über das Anwendungsgebiet dargestellt. Der domain layer definiert die anwendungsspezifische Terminologie als auch Wissen um die Beziehungen zwischen den Entitäten des Gebietes. Auf dem *inference layer* werden die Inferenzschritte der Problemlösungsmethode, der Datenfluß zwischen diesen Inferenzen und die Verwendung des auf dem domain layer spezifizierten Wissens für die Problemlösung definiert. Auf dem *task layer* wird erstens eine deklarative Beschreibung der Funktionalität des System, zweitens eine hierarchische Zerlegung des Problems in Teilprobleme und der Kontrollfluß zwischen den Inferenzschritten der Problemlösungsmethode definiert.

Tatsächlich wurde das Model der Expertise und die in ihm verwendeten Modellierungsprimitive nur informal definiert. *Informale*[5] oder *semiformale*[6] Spezifikationen von Softwareprodukten, denen informale oder semiformale knowledge level Beschreibungen von Expertensystemen entsprechen, haben jedoch einige wohlbekannte Nachteile:

- Dokumente in natürlicher Sprache[7] haben eine mehrdeutige und unpräzise Semantik. Für eine Spezifikation existieren so viele Bedeutungen wie Leser, und es ist eine Frage der Textinterpretation, ob ein System einer Spezifikation genügt.[8] Um Mißverständnisse zu vermeiden: Im allgemeinen ist Mehrdeutigkeit und Vagheit von Ausdrücken in natürlicher Sprache nicht ein Nachteil, sondern eine wesentliche Eigenschaft natürlicher Sprache.[9] Subjekte mit unterschiedlicher Wirklichkeitsinterpretation können so gemeinsame Interpretationsmuster entwickeln, ohne daß ihre hundertprozentige

5. Eine *informale Beschreibungstechnik* benutzt beliebige natürlichsprachliche Texte.

6. Eine *semiformale Beschreibungstechnik* hat eine definierte Syntax, eine Menge vordefinierter Primitive und erlaubt die Verwendung natürlicher Sprache in eingeschränkter Weise. Im Unterschied zu formalen Beschreibungstechniken verfügt sie jedoch nicht über eine definierte Semantik.

7. Die Ausdrücke natürliche Sprache, normale Sprache und Umgangssprache werden in dem Artikel als Synonyme behandelt.

8. Die Probleme natürlichsprachlicher Spezifikationen sind eindringlich illustriert in [Mey85]. Dieser schöne und ursprünglich 1979 erschienene Text macht eigentlich jeden weiteren Artikel “pro formale Spezifikation” unnötig.

Übereinstimmung schon unterstellt ist. Jedoch ist ein Computerprogramm (noch ?) kein autonomer Agent, der die Wirklichkeit interpretiert, sondern ein Produkt, welches präzise und genau beschrieben werden muß, wenn es verstanden und beherrscht werden soll.

- Es existiert eine große kognitive Distanz zwischen einer informalen oder semiformalen Spezifikation einerseits und einem implementierten Programm andererseits. Die Spezifikation ist daher kein geeignetes Mittel zur Evaluierung des Systems.
- Informale oder semiformale Spezifikationen sind weder ausführbar, d.h. sie unterstützen die Validierung nicht durch Testen, noch sind formale Analysetechniken anwendbar, die auf Basis der formalen Semantik mittels Deduktion die Spezifikation auf Vollständigkeit oder interner Korrektheit hin überprüfen.
- KADS unterstützt die Widerverwendung von Inferenzaktionen (elementare Schritte einer Problemlösung) durch eine vordefinierte Menge standaridisierter Inferenzen und von komplexen Problemlösungsmethoden durch Bibliotheken vordefinierter Bausteine. Beide sind jedoch derzeit nur informal bzw. semiformal beschrieben. [Abe92] zeigte, daß z.B. ein und dieselbe Standardinferenzaktion in völlig verschiedenen bis gegensätzlichen Bedeutungen von Anwendern verwendet wird. Ähnliches gilt für die nur informal definierten Problemlösungsmethoden wie die Erfahrung des Sysiphus-II Projektes zeigten.[10] Es werden daher nur Namen wiederverwendet und damit Inferenzen bzw. Problemlösungsmethoden mit völlig verschiedenen bis gegensätzlichen Bedeutungen bezeichnet. Durch tatsächliche Wiederverwendung vordefinierter Bausteine mit einer standardisierten Bedeutung durch ihre formale Definition könnten Entwicklungsaufwand und -kosten deutlich reduziert und durch Standardisierung Verständlichkeit und Zuverlässigkeit deutlich erhöht werden. In [Fen95b] wird exemplarisch gezeigt, welche Annahmen eine wiederverwendbare Problemlösungsmethode und ihre unterschiedlichen Varianten über Eigenschaften des verfügbaren Domänewissen machen. Die Gültigkeit dieser Annahmen für ein gegebenens Domänewissen bzw. die Ableitung der geeigneten Variante der Problemlösungsmethode auf der Basis der vorgefundenen Eigenschaften erfordern eine formale Spezifikation.

Das Ziel, diesen Mängeln abzuhelfen, hat zu einer wahren Sprachflut geführt.[11] Im engeren Sinn sind hierbei die Sprachen DESIRE (Freie Universität Amsterdam) [LPT93], FORKADS (IBM-Heidelberg) [Wet92], KARL (Universität KARLsruhe) ([Ang93], [Fen93b]), $K_{BS}SF$ (PTT Groningen) [JoS92], $(ML)^2$ (Universität Amsterdam und ECN Netherlands) [HaB92], Models (Universität Bukarest) [Bar93], OMOS (GMD-Bonn) [Lin92] und QIL (Universität Nottingham) [ARS92] zu erwähnen. Alle diese Sprachen sind Spezifikationssprachen, die eine *formale*[12] und/oder *ausführbare*[13] Beschreibung des Modells der Expertise erlauben. Neben diesen Spezifikationssprachen sind noch die beiden bei der GMD-Bonn entwickelten Sprachen Model-K [Kar93] und Momo [VoV93] zu erwähnen, die sich darum bemühen, die Struktur des Modells der Expertise auch für die Implementierung des Systems zu erhalten.

---

9. “If we begin with the assumption that vagueness is central to the social construction of meaning, we will not try to reduce human language to formal logic. Instead we will ask why natural signs in the form of words have so many multiple and inconsistent meanings and why being able to link them to one another in such a wide variety of ways, even inconsistent ways, is culturally advantageous.” [Joh93]

10. Das Sisyphus-II Problem besteht aus der Spezifikation und Implementierung eines wissensbasierten Systems zur Konfiguration von Aufzügen auf der Basis eines wirklichen Anwendungsfalles. In [GaM94] sind acht Lösungen dokumentiert, von denen sieben angeben, die Problemlösungsmethode Propose-and-Revise zu verwenden. Tatsächlich unterscheiden sich jedoch alle sieben benutzten Problemlösungsmethoden bzw. es ist überhaupt nicht klar, wie die verschiedenen Ansätze geeignet dokumentiert und verglichen werden können.

11. Insgesamt sind bis jetzt vier Dissertationen und eine Habilitation hierzu erschienen. Zumindest hierfür sind formale Spezifikationssprachen also hervorragend geeignet.

12. Eine *formale Beschreibungstechnik* hat eine in einem mathematischen Formalismus definierte Semantik. Diese Semantik kann nichtkonstruktiv sein, d.h. es muß kein Berechnungsverfahren hierfür existieren.

13. Eine *ausführbare Beschreibungstechnik* erlaubt die Berechnung von Ergebnisses als Resultat bestimmter Eingaben und setzt so eine konstruktive bzw. operationale Semantik voraus.

Ein Vergleich der meisten Sprachen finden sich in [FeH94] und [TrW93]. Mittlerweile haben sieben Workshops auf europäischer Ebene (siehe [TrW93], [Fen95a]) zum Vergleich und besseren Verständnis der jeweiligen Ansätze beigetragen. Weitere Workshops sind in Vorbereitung, z.B. in Zusammenhang mit dem nächsten European Symposium on Validation and Verification of Knowledge-based Systems (EUROVAV-95). All diesen Sprachen (mit Ausnahme von DESIRE) ist gemeinsam, daß sie das Modell der Expertise des KADS-I Projektes und die von ihm definierten Primitive anbieten, um so die Lücke zwischen symbol und knowledge level zu überbrücken.[14] Das Modellieren mit den Primitiven, die sich in den letzten zehn Jahren im Bereich des KE als adäquat zur konzeptuellen Beschreibung wissensbasierter Systeme herausgestellt haben, wird so auf wohldefinierte Weise möglich. Über ihre Gemeinsamkeiten hinweg, lassen sich diese Sprachen grob in drei Klassen einteilen:

- Sprachen, die eine *Operationalisierung des Modells der Expertise* erlauben. Diese Sprachen fügen die Vorteile der Evaluierung durch Testen zum Modellierungsprozeß hinzu. Eine Sprache wie z.B. Model-K enthält einerseits die Modellierungsprimitive des KADS-I Modells der Expertise und erlaubt andererseits durch Frames, Lisp oder Prolog eine ausführbare Semantik für sie zu definieren.
- Sprachen, die eine *Formalisierung eines Modells der Expertise* erlauben. Diese Sprachen ermöglichen eine präzise und detaillierte Beschreibung und erlauben z.B. Überprüfung auf Korrektheit oder Vollständigkeit (vgl. [Yue87]). Sie haben eine mathematisch definierte Semantik (meist eine modelltheoretische) für ihre Modellierungsprimitive und die WissensingeneurIn kann das Wissen deklarativ beschreiben, ohne es implementieren zu müssen. Sie muß nicht eine spezifische berechenbare AgentIn spezifizieren, die die Lösung berechnet, sondern sie kann auf allgemeine Weise die Aufgabe und das benötigte Wissen beschreiben. Wegen der definierten Semantik ist dies in einer eindeutigen und präzisen Art und Weise möglich. $(ML)^2$ bietet z.B. eine Kombination ordnungssortierter Prädikatenlogik, Metalogik und dynamischer Logik zur Beschreibung eines Modells der Expertise an.
- Sprachen, die eine *Formalisierung und Operationalisierung eines Modells der Expertise* erlauben. Diese Sprachen versuchen die Vorteile von formalen und ausführbaren Sprachen zu integrieren. Im Vergleich zu den reinen Formalisierungssprachen erreichen sie dies im wesentlichen durch eine Einschränkung der Sprachmächtigkeit, um so eine effektive und effizientere Berechnung der Semantik zu ermöglichen. Im Vergleich zu $(ML)^2$ schränkt KARL z.B. Prädikatenlogik auf Hornlogik mit stratifizierter Negation und “kleinen” minimalen Herbrandmodellen ein.

Mittlerweile liegen rund hundert Anwendungen bzw. Fallstudien dieser Sprachen im akademischen Bereich vor (vgl. [FeH94]). Einige kommerzielle Anwendungen von $(ML)^2$ haben begonnen. Für die älteste Sprache DESIRE (sie befindet sich im Alter von vier) existieren bereits durchgeführte kommerzielle Anwendungen. Natürlich spüren auch diese Sprachen den Wind, der derzeit der KI bzw. der Expertensystemtechnik in das Gesicht weht. Gegenwärtige Forschungsvorhaben zu formalen Spezifikationssprachen zielen auf die Entwicklung formaler Validierungs- und Verifikationstechniken für diese Sprachen und auf die Spezifikation von Bibliotheken wiederverwendbarer Inferenzaktionen und Problemlösungsmethoden (vgl. [Abe93]). Hierbei können formale Schlußfolgerungsverfahren z.B. dazu eingesetzt werden, das sogenannte Indexing- bzw. Retrievalproblem zu lösen. Im wesentlichen müssen hierfür jedoch noch einerseits Schlußfolgerungskalküle für diese Sprachen auf der Basis ihrer formalen Semantiken entwickelt werden und andererseit die relevanten Eigenschaften der Problemlösungsmethoden erfaßt und geeignet beschrieben werden.

14. Für eine ausführlichere Diskussion siehe [Fen93a].

2. Unsinn formaler Spezifikationen?

Die Vorteile formaler Wissensspezifikationssprachen sind leicht zu verstehen (vgl. [Mey85], [Win90], [Hal90], [AHS93]):

- Sie erlauben eine präzise und eindeutige Beschreibung des Wissens, ohne diese mit Design- und Implementierungsaspekten zu vermischen. Sie überbrücken also den Gegensatz von informalen und implementierungsunabhängigen Beschreibungen einerseits und überpräzisen, weil Implementierungsdetails festlegenden, Beschreibungen andererseits.
- Sie erlauben die Evaluierung des spezifizierten Wissens durch formale Techniken (Operationalisierung, symbolische Evaluierung, Beweise von Vorbedingungen, Nachbedingungen und Invarianten, etc.).[15]
- Sie bauen eine Brücke zwischen der Vorstellung über das Wissen im Kopf der WissensingeneurIn und dem wirklich implementierten System und erleichtern so den Übergang in beide Richtungen.
- Durch die formale Spezifikation wiederverwendbarer Inferenzaktionen, Problemlösungsmethoden und Ontologien wird der Entwicklungsaufwand wissensbasierter Systeme verringert und durch Standardisierung Verständlichkeit und Zuverlässigkeit erhöht.

Trotzdem ist ihr Sinn nicht unbestritten. Jeder Formalisierungsprozeß eines Bereiches sieht sich dabei mehr oder weniger kleinen Variationen der gleichen Art von Argumenten gegenüber. Dies galt für die Mathematik Anfang des 20. Jahrhunderts (vgl. [Meh90] und [Hei93]) genauso wie für SE in den siebziger und frühen achtziger Jahren (vgl. z.B. [Nau82], [Mey85], dessen Orginalfassung 1979 erschien, [EFN85], und [Hal90]) oder den Bereich der Wissensrepräsentationsformalismen in den Achtzigern. Insbesondere die Argumente aus dem SE für und wider formale Spezifikationen können dabei unmittelbar auf das KE übertragen werden. Sie bekommen nur ein zusätzliches Gewicht. Während im SE der Nutzen formaler Spezifikationstechniken vorallem darin gesehen wird, funktionale Anforderungen an ein System zu präzisieren, kommt im KE der zusätzliche Aspekt des Problemlösewissens der ExpertIn hinzu. Im KE hat man daher nicht nur mit der Vagheit funktionaler Anforderungen zu kämpfen, sondern mit dem Hauptproblem, daß wesentliche Teile des benötigten ExpertInnenwisssens als Können, implizites Wissen, vages Wissen, heuristisches Wissen etc. existiert [Ber87]. Eine zweite wichtige Eigenschaft von Expertensystementwicklung ist die Trennung von aufgabenspezifischen Problemlösungsmethoden und Domänewissen. Problemlösungsmethoden können für ähnliche Aufgaben in unterschiedlichen Domänen widerverwendet werden und Domänewissen kann zur Lösung unterschiedlicher Aufgaben herangezogen werden. Jede Problemlösungsmethode macht jedoch spezifische Annahmen über die Art des verfügbaren Wissens und der vorliegenden Aufgabe und nur auf Basis einer formalen Spezifkation kann über deren Gültigkeit entschieden werden.[16]

Obwohl daher in neuem Gewand immer wieder auch die alte Debatte geführt wird sollen einige Einwände gegen formale Spezifikationssprachen diskutiert werden. Die Notwendigkeit hiefür ergibt auch dadurch, daß diese Debatte selbst im SE noch nicht zu Ende ist. So wurden z.B. auf der letzten International Conference on Software Engineering (ICSE´94) in Papern und Penaldiskussion insbesondere die Relevanz formaler Spezifikationstechniken für die tatsächliche kommerzielle Softwareentwicklung kontrovers diskutiert.

---

15. Vgl. [Lyd92].

16. Natürlich gilt hier auch das allgemeine Argument der Softwarewiederverwendung, daß Wiederverwendung formale Spezifikation und Verifikation möglich - da bezahlbar - macht und umgekehrt nötig macht, da nur so fremden Bausteinen vertraut werden kann.

Das Wissen des Experten kann nicht bzw. nicht adäquat formal beschrieben werden.

Dies mag durchaus richtig sein, gilt jedoch erstens in noch viel höherem Maße für die spätere Implementierung, da ein ausführbares Computerprogramm aus prinzipiellen Gründen eine eingeschränktere Ausdrucksmächtigkeit hat als z.B. Prädikatenlogik erster Stufe. Zweitens stellt die natürlichsprachliche Beschreibung der Expertise selbst schon eine wesentliche Reduktion dar und man könnte mit derselben Berechtigung fragen, ob vor allem auf Können beruhende menschliche Expertise hinreichend durch natürliche Sprache beschrieben werden kann.

> *"Programs, however, are formal objects, susceptible to formal methods ... Thus, programmers cannot escape from formal methods. The question is whether they work with informal requirements and formal programs, or whether they use additional formalism to assist them during requirements specification." [Win90]*

Formale Sprachen sind schwierig zu lernen.

Dies ist das alte Argument der Assemblerprogrammierer gegen Fortran und läuft im Grunde auf die Frage hinaus, was einfacher zu verstehen ist: low-level oder high-level Programmiersprachen. Jede ausführbare und im eingeschränkten Sinn auch jede formale Spezifikationssprache kann als Programmiersprache auf einer sehr hohen konzeptuellen Ebene angesehen werden. Der Aspekt der effizienten und maschinennahen Implementierung tritt hier hinter dem Anspruch zurück, Sprachkonstrukte auf einem hohen konzeptuellen Niveau anzubieten. Schon [Zav91], die wesentlich an der Entwicklung der operationalen Spezifikationssprache PAISley mitwirkte, wies darauf hin: *"an executable specification language is a specialized programing language."* Das Verhältnis von Spezifikationssprachen und Programmiersprachen ist dabei ständigem Wandel unterworfen. Oder anders formuliert: Die Spezifikationssprachen der Gegenwart sind die Programmiersprachen der Zukunft, da sie durch das höhere konzeptuelle Niveau eine effizientere, weil verständlichere Programmierung erlauben.

Gerade die hier diskutierten Sprachen benutzen das Modell of Expertise als konzeptuelles Modell. Sie erlauben eine graphische Repräsentation der meisten Modellierungsprimitive und die formale Spezifikation kann so an schon erstellte semiformale Modellierungen anknüpfen und diese verfeinern. Die graphischen Modellierungsprimitive des KADS-Modells werden dabei z.B. um Petrinetze oder erweiterte Entity-Relationship Diagramme (EER) ergänzt. Natürlich ist hier nicht in erster Linie die graphische Repräsentation, sondern die Verwendung geeigneter Modellierungsprimitive und die Darstellung des Systems mit Hilfe eines konzeptuellen Modells wichtig.

Formale Spezifikationen sind zu komplex und zu schwer zu erstellen bzw. zu verstehen.

Im wesentlichen sind hier die Aspekte der *Präzision* und der *Eindeutigkeit* diskutiert. Da formale und ausführbare Spezifikationen eine detaillierte Formulierung des modellierten Wissens erzwingen, werden formale Spezifikationen schnell groß und unübersichtlich. Strukturierung, Hierarchisierung und Modularisierung sind bekannte Techniken, um dieses Problem abzumildern. Gerade auch hier bietet das verwendete konzeptuelle Modell dieser Sprachen deutliche Vorteile, da es klar zwischen verschiedenen Wissenarten bzw. Ebenen unterscheidet und die gesamte Spezifikation in kleine Bestandteile mit klar definiertem Zweck unterteilt (z.B. eine elementare Inferenzaktion, ein domain view einer Inferenzaktion, eine Task, eine knowledge role, etc.). Übrigens ist dies ein deutlicher Vorteil der hier diskutierten Sprachen gegenüber den general-purpose Spezifikationssprachen des SE mit ihrem schwächeren konzeptuellen Modell. Dies wird natürlich nur dadurch möglich, daß sich im KE auf eine enge Klasse von Systemen beschränkt wird, was daher speziellere Ansätze ermöglicht.

Es muß immer auch betont werden, daß eine formale Spezifikation eine informale Spezifikation nicht ersetzen, sondern verfeinern, d.h. also präzisieren soll. Es ist fast immer

leichter, eine Idee durch das Lesen eines natürlich-sprachlichen Textes zu verstehen, als zu versuchen, diese mühsam aus einer Menge von Formeln zu extrahieren. Andererseits führt der Versuch einer präzisen und vollständigen natürlichsprachlichen Definition eines Sachverhalts meist zu unlesbaren und unverständlichen Schachtelsätzen, die doch immer wieder Zweideutigkeiten, Redundanzen, Lücken und Wiedersprüche enthalten.

> *"The main advantage of natural language texts is their understandability. One should concentrate on this asset rather than trying to use natural language for precision and rigor, qualities for which it is hopelessly inadequate." [Mey85]*

Ein anderer Einwand betrifft die Eindeutigkeit formaler Spezifikationen. Formale Spezifikationssprachen haben zwar eine exakt definierte mathematische Semantik, nur verstehe und verwende diese außer den Sprachschöpfern kaum jemand sonst. Tatsächlich sind die formalen Semantiken dieser Spezifikationssprachen meist ein komplexes und nicht ganz einfaches Stück Mathematik. Richtig gewählt verlieren sie jedoch viel von ihrem Schrecken. Ein Beispiel dafür ist die Art, wie in F-Logic (siehe [KLW93]) Welltyping auf deklarative Weise durch die Verwendung partieller anti-monotoner Funktionen mit nach oben geschlossenen Teilmengen als Bildern definiert wird. Das komplexe und scheinbar schwer zu verstehende mathematische Gebilde definiert jedoch leicht verständliche, weil intuitive Bedingungen für die Mehrfachvererbung von Attributen und deren Wertbereichsbeschränkungen. Formale Semantiken sind also dann leicht zu verstehen, wenn sie intuitiver Semantik entsprechen und diese nur exakt definieren. Ein anderes Beispiel ist die Interpretation von Hornklauseln als *feuernde Regeln* oder *Prozeduren*, die für Variablenbelegungen der Prämissen Variablenbelegungen der Konklusionen erzeugen, als Resultat der minimalen bzw. perfekten Herbrandmodellsemantik (siehe [Llo87]).

Nicht übersehen werden darf hierbei auch die doppelte Funktion, die formale Spezifikationen haben (vgl. [FBA+93]):

- *Mediating Representation:* Die Spezifikation vermittelt die Kommunikation zwischen ExpertIn und WissensingeneurIn. Die formale Spezifikation definiert dabei den technischen Kern der Kommunikationsgrundlage. Sie erlaubt der WissensingeneurIn eine präzise Beschreibung ihrer Interpretation des Expertenwissens.
- *Intermediate Representation:* Die Spezifikation überbrückt die Distanz zwischen einer informalen Systembeschreibung einerseits und dem implementierten System andererseits, d.h. es vermittelt die Kommunikation zwischen WissensingeneurIn und ProgrammiererIn.

"Verständlichkeit" formaler Spezifikationen muß also bezüglich unterschiedlicher Personengruppen diskutiert werden.

Formale Spezifikationen sind zu teuer.

> *"The greatest benefit in applying a formal method often comes from the process of formalizing rather than from the end result." [Win90]*

Tatsächlich verursacht es auf den ersten Blick zusätzliche Kosten, eine Spezifikation zu formalisieren. Andererseits ist dieser Einwand wohl der vordergründigste. Jedes Standardlehrbuch des SE ist voll von Ermahnungen, daß je später ein Spezifikations- oder Programmierfehler im Softwareentwicklungsprozeß gefunden wird, dessen Beseitigung umso teurer wird. Eine informale Spezifikation mit ihrer prinzipiellen Unvollständigkeit zwingt den Programmierer geradezu, Lücken durch seine Phantasien zu überbrücken. Und nicht immer trifft er dabei die Erwartungen der späteren Benutzer bzw. das fehlende Expertenwissen.

Der überwiegende Teil des Aufwands beim Erstellen einer formalen Spezifikation ist also nicht der Verwendung einer formalen Spezifikation geschuldet, sondern dem Ziel, eine informale Spezifikation zu präzisieren sowie Mehrdeutigkeiten und Widersprüche zu beseitigen. Soll das System implementiert werden, muß dieser Aufwand sowieso getrieben werden. Dann jedoch nicht mehr in einer geeigneten Phase und von den geeigneten Personen.

Auch zusätzlicher Aufwand für Beweisobligation und deren Beweis dienen ja nur dazu, daß Produkt besser zu verstehen.

In diesem Zusammenhang ist auch wichtig, Spezifikationen nicht nur als nützliches *Produkt*, sondern das Schreiben von Spezifikationen als wichtigen *Prozeß* zu betrachten. In diesem Prozeß werden Aussagen präzisiert, Mehrdeutigkeiten und Widersprüche erkannt und beseitigt. Dieser Prozeß kann durch Beweiswerkzeuge oder Interpreter unterstützt werden, aber schon die bloße formale Definition z.B. einzelner Inferenzschritte eines wissensbasierten Systems offenbart in aller Regel Schwächen ihrer informalen Definitionen (vgl. [RHA94]). Die erkannten Fehler und Probleme können dazu benutzt werden, die informale Spezifikation zu verbessern. So ist manchmal vielleicht gar keine formale Spezifikation als Abschlußdokument notwendig, sondern lediglich der Prozeß der Formalisierung und die durch ihn ermöglichte Verbesserung der informalen Spezifikation.

> *"The fact is that writing a formal specification decreases the cost of development." [Hal90]*

Auch die Semantik formaler Spezifikationen gründet in der natürlichen Sprache.

> *„Die Wissenschaft baut nicht auf Felsengrund. Es ist eher ein Sumpfland, über dem sich die kühne Konstruktion ihrer Theorien erhebt; sie ist ein Pfeilerbau, dessen Pfeiler sich von oben her in den Sumpf senken - aber nicht bis zu einem natürlichen, 'gegebenen' Grund. Denn nicht deshalb hört man auf, die Pfeiler tiefer zu hineinzutreiben, weil man auf eine feste Schicht gestoßen ist: wenn man hofft, daß sie das Gebäude tragen werden, beschließt man, sich vorläufig mit der Festigkeit der Pfeiler zu begnügen." [Pop35]*

Ein letzter und wesentlicher Einwand gegen formale Sprachen sei noch erwähnt. Formale Sprachen haben nur vordergründig eine exakt definierte Semantik. In der Regel wird die Semantik einer formalen Sprache durch die Verwendung einer zweiten formalen Sprache definiert, deren Semantik als bekannt vorausgesetzt wird.[17] Sehr schnell wird man hier also in den unendlichen Regreß verwiesen. Und es kommt noch schlimmer. Jeder Definitionsversuch der einen formalen Sprache durch die Verwendung einer zweiten formalen Sprache kommt nicht ohne die Verwendung natürlicher Sprache aus. Tatsächlich ruhen die Stützpfeiler der Kristallpaläste der formalen Semantik also im unendlichen Regreß bzw. auf den trüben Sümpfen der *normalen* bzw. *Umgangssprache*. Verfolgt man diesen Regreß weiter stößt man natürlich schnell auf ein weiteres bzw. das *eigentliche* Problem: Worin gründet eigentlich die Bedeutung der Umgangssprache?

[Mea34] zeigt die entwicklungsgeschichtliche Bedeutung der *vokalen* Geste für die menschliche Kommunikation. Im Unterschied zur nichtvokalen Geste kann hier der Sender gleichzeitig: erstens die eigene *Geste wahrnehmen*; und so zweitens die *Wirkung wahrnehmen*, die diese in ihm selbst hervorruft; und er kann drittens die *Reaktion* eines anderen Empfängers *wahrnehmen*. Beim Sender kann so über die gleichzeitige Wahrnehmung von innerem Reiz und äußerer Reaktion eine Vorstellung über die Bedeutung der Geste entstehen. Bewußtsein über die Bedeutung einer Geste wurde notwendig, als unsere Vorfahren den engen Kreis instinktiv fixierten Verhaltens zur Natur und zueinander verließen, und so der Instinkt keine ausreichende Grundlage mehr für die Koordination der Tätigkeit und des *Verstehens* bot. Menschen haben so schon immer gehandelt, bevor sie verstanden und gedacht haben. Sprache dient der Synchronisation von gemeinschaftlichen Unternehmungen, und wenn der Einzelne sich dabei adäquat als Glied des Zusammenhanges setzen kann, hat er die *Bedeutung* der Ausdrücke *verstanden*. Wird die natürliche Sprache so als Reflex des praktischen Verhaltens der Individuen verstanden, werden die Schwierigkeiten maschineller Sprachverarbeitung verstehbar [WiF86]. Ein Sprachakt hat seine unmittelbare

17. Oder man macht es wie [Spi88] und definiert die Semantik der Sprache Z mit Hilfe der Sprache Z. Nicht umsonst interpretiert das Herbrand Model einer prädikatenlogischen Formelmenge Konstanten und Funktionen durch sich selbst.

Bedeutung nur in seinem Kontext. Erst Standardisierung d.h. Übereinkunft weist natürlichsprachlichen Ausdrücken eine bestimmte feste Bedeutung zu. Dabei ist der Versuch, diese feststehende Bedeutungen natürlichsprachlich zu definieren demselben schon erwähnten logischen Zirkel unterworfen. Die zyklische Definition von Ausdrücken durch sich selbst wird z.B. durch die Erfahrungen des CYC-Projektes [LeG90] belegt, welches Alltagswissen aus Lexikas extrahieren und formalisieren will.

Eine formale Sprache treibt diesen Standardisierungsprozeß voran. Ihre formale Semantik dient dazu, Ausdrücken der Sprache eine feste und kontextfreie Bedeutung zuzuweisen. Die formale Semantik definiert einen (hoffentlich verständlichen) Standard für die Bedeutung der Ausdrücke der Sprache. Die Definition von kontextfreien Bedeutungen für Sprachausdrücke vereinfacht es, Sachverhalte so zu formulieren, daß sie eine eindeutige und präzise definierte bzw. verstehbare Bedeutung haben. Sowohl zur Definition ihrer Bedeutung und daher auch zu ihrem Verständnis ist man jedoch noch immer auf normale Sprache angewiesen. Formale Sprachen sind nur ein geeigneteres Kommunikationsmittel für bestimmte Zwecke. Aufgrund ihrer standardisierten und daher kontextfreien Semantik sind formale Sprachen übrigens wesentlich leichter zu lernen und zu verstehen als normale Sprachen. Und in Analogie zur normale Sprachen hat jemand die mathematische Semantik einer formalen Spezifikationssprache verstanden, wenn er mit dieser Sprache geeignet arbeiten kann.[18]

Zusätzlich ist es durch die standardisierte Bedeutung einer formalen Sprache möglich, einen Schlußfolgerungskalkül zu entwickeln, der gültige Eigenschaften von Mengen von Ausdrücken ableitet. Mit diesen Kalkülen lassen sich z.B. Fragen der Korrektheit und Vollständigkeit von formalen Spezifikationen bezüglich formal definierter Ziele überprüfen. Auch hier bleibt jedoch das Problem erhalten, daß niemals die Übereinstimmung der formalen Spezifikation mit einer informalen Spezifikation bzw. mit der Intention des Systemanwenders *formal* gezeigt werden kann. Korrektheit und Vollständigkeit einer Spezifikation bezüglich einer informalen Aufgabenbeschreibung kann also nur außerhalb des formalen Systems selbst entschieden werden (siehe [Fet88]). Die formale Spezifikation liefert für diesen Prozeß aber einen wohldefinierten Bezugspunkt. Darüberhinaus verhalten sich beide Seiten - wie schon erwähnt - allerdings sowieso nicht so statisch wie hier diskutiert, da die Erstellung einer formalen Spezifikation meist ein Prozeß ist, in dem die Anforderungen überhaupt erst präzissiert werden. Daher ändern sich die Anforderungen (sie entstehen zum Teil erst) im Prozeß ihrer Formalisierung. Die Frage, ob eine formale Spezfikation der ursrünglichen Intention des späteren Systembenutzer entspricht, ist in der Regel also bloße Scholastik. Umgekehrt soll sie dabei helfen, diese realistischer zu machen.

3. Ein Vergleich mit dem Software Engineering

Die Entwicklung formaler und ausführbarer Spezifikationssprachen hat eine lange, fast fünfundzwanzigjährige, Tradition im SE--in kleinerem Umfang und kürzerem Zeitraum gilt dies auch für den Bereich der Informationssystementwicklung. Mittlerweile hat dies zu einer kaum überschaubaren Fülle von Sprachen geführt (von A wie *ATOL* und Z wie *Z* sind dem Autor weit mehr als 30 Sprachen bekannt).[19] Sprachen wie z.B. VDM und Z (siehe [BHL90]) sind das Ergebnis langjähriger (immer noch laufender) Forschungsprojekte mit einer Vielzahl kommerzieller Anwendungen (vgl. [BFL+93], [WoL93]). Dabei fällt auf, daß erst in letzter Zeit, d.h. ungefähr in den letzten 5-7 Jahren, exakte formale Semantiken für diese Sprachen definiert wurden und ihre Axiomatisierung, d.h. die Entwicklung von mechanischen Schlußfolgerungskalkülen auf Basis einer formalen Semantik, daher aktueller Forschungsgegenstand ist (vgl. [AGM93]). Wichtig in diesem Zusammenhang sind Arbeiten im Bereich der Programmverifikation wie z.B. das System KIV (Karlsruhe Interactive

18. Oder um mit Wittgenstein zu sprechen, wenn er die in ihr formulierten Befehle versteht, d.h. wenn er sie richtig ausführt.
19. Eine Auflistung formaler Methoden zur Systementwicklung ist enthalten in [RyS93].

Verifier, siehe z.B. [HRS90]), das die Verifikation prozeduraler Programme unter Verwendung eines Theorembeweisers für dynamische Logik ermöglicht. Eine wissenschaftliche Evaluierungsstudie einiger kommerzieller Anwendungen formaler Spezifikationstechniken findet sich in [CGR93].[20]

Als wesentlicher Unterschied beim Vergleich von Sprachen des SE und des KE fällt auf, daß den Sprachen des KE i.d.R. ein stärkeres konzeptuelles Modell des zu beschreibenden Systems unterliegt. Sie benutzen z.B. meist das KADS Modell der Expertise und sind daher viel näher an einer konzeptuellen und informalen bzw. semiformalen Beschreibung der Expertise als *low-level* bzw. *general-purpose* Sprachen wie z.B. *Z*, die beliebige Programme mit Hilfe der mathematischen Mengentheorie beschreiben. KE Sprachen nutzen dagegen aus, daß sie eine spezielle Klasse von Software beschreiben. Eine formale Spezifikation ist so kein unübersehbarer Formelwust, sondern sie ist zum überwiegenden Teil mit der semiformalen Spezifikation identisch. Die semiformale Spezifikation definiert eine Schablone, deren Lücken durch formale Definitionen angereichert werden. Im SE gibt es viele bekannte Ansätze, die, wie z.B. Structured Analysis oder Object-Oriented Analysis, rein auf semiformalen Methoden basieren. Umgekehrt thematisieren viele formale Methodenentwickler nicht die sinnvolle Kombination mit informalen oder semiformalen Methoden. Hierbei wäre nicht nur die Definition eines graphischen Repräsentationsformalismus gefordert, sondern die Beantwortung der Frage, wie die konzeptuelle Modellierung des System mit seiner Beschreibung als partieller mathematischer Funktion korrespondiert. Die enge Integration konzeptueller und mathematischer Beschreibungstechniken könnte vielleicht etwas sein, was das SE umgekehrt vom KE lernen könnte. Dabei ist dies für den Bereich des KE natürlich leichter, da es durch die beschränktere Klasse von Software, die modelliert werden soll, unterstützt wird.[21]

Ein zweiter Unteschied betrifft das Vorhandensein von Schlußfolgerngskalkülen und automatischen Beweisern für diese Spezifikationssprachen. Hier muß den Sprachen aus dem SE neidvoll ein Entwicklungsvorsprung anerkannt werden. Für einige der Sprachen aus dem KE wie DESIRE, KARL und $(ML)^2$ werden jedoch derzeit Methoden und Werkzeuge entwickelt bzw. angepaßt, die die Deduktion von formal spezifizierten Eigenschaften einer Spezifikation ermöglichen sollen.

4. Die stufenweise Überwindung der Distanz zwischen menschlicher Expertise und implementiertem Expertensystem

In den letzten Jahren wurde im KE intensiv daran gearbeitet, semiformale und formale bzw. konzeptuelle und mathematische Spezifikationen wissensbasierter Systeme zu kombinieren, statt sie als abstrakten Gegensatz zu diskutieren (vgl. z.B. [Ste90], [AFL+93], [SWB93]).[22] Hierzu wurde eine Reihe miteinander kombinierter Beschreibungsebenen eingeführt (siehe Abb. 1). Als Resultat des ersten Schrittes (Knowledge Elicitation) liegt das zu modellierende Wissen meist in der Form *natürlichsprachlich formulierter Wissensprotokolle* als Resultat von Literaturstudien, Interviews oder der Transkription von Protokollen vor. Gegenüber der menschliche Expertise stellt dies schon eine wesentliche Reduktion und Strukturierung dar. In einem zweiten Schritt (Knowledge Interpretation) wird das Wissen auf der begrifflichen

20. Eine repräsentative Evaluierungstudie müßte gleichartige Projekte einmal mit und einmal ohne formale Spezifikationsmethoden durchführen und dies in einer statistisch ausreichenden Wiederholungszahl. Um Plazeboeffekte zu vermeiden dürften dabei in der Form des Blindtestes den beteiligten Gruppen nicht einmal bekannt sein, ob sie formale Methoden benutzen oder nicht. Statistisch fundierte Evaluierungsstudien scheiden daher also aus. Was bleibt sind Einzelfallstudien mit begrenzter Verallgemeinerungsfähigkeit.

21. Dabei scheinen allerdings die Grenzen nicht sehr eng gezogen zu sein, da wie in [FAL+93] gezeigt z.B. KARL unmittelbar zur Formalisierung und Operationalisierung der Beschreibungstechniken (z.B. der Datenflußdiagramme) der Structured Analysis benutzt werden kann.

22. Damit wurde einer Forderung entsprochen, die z.B. [Flo85] für das SE gestellt hatte.

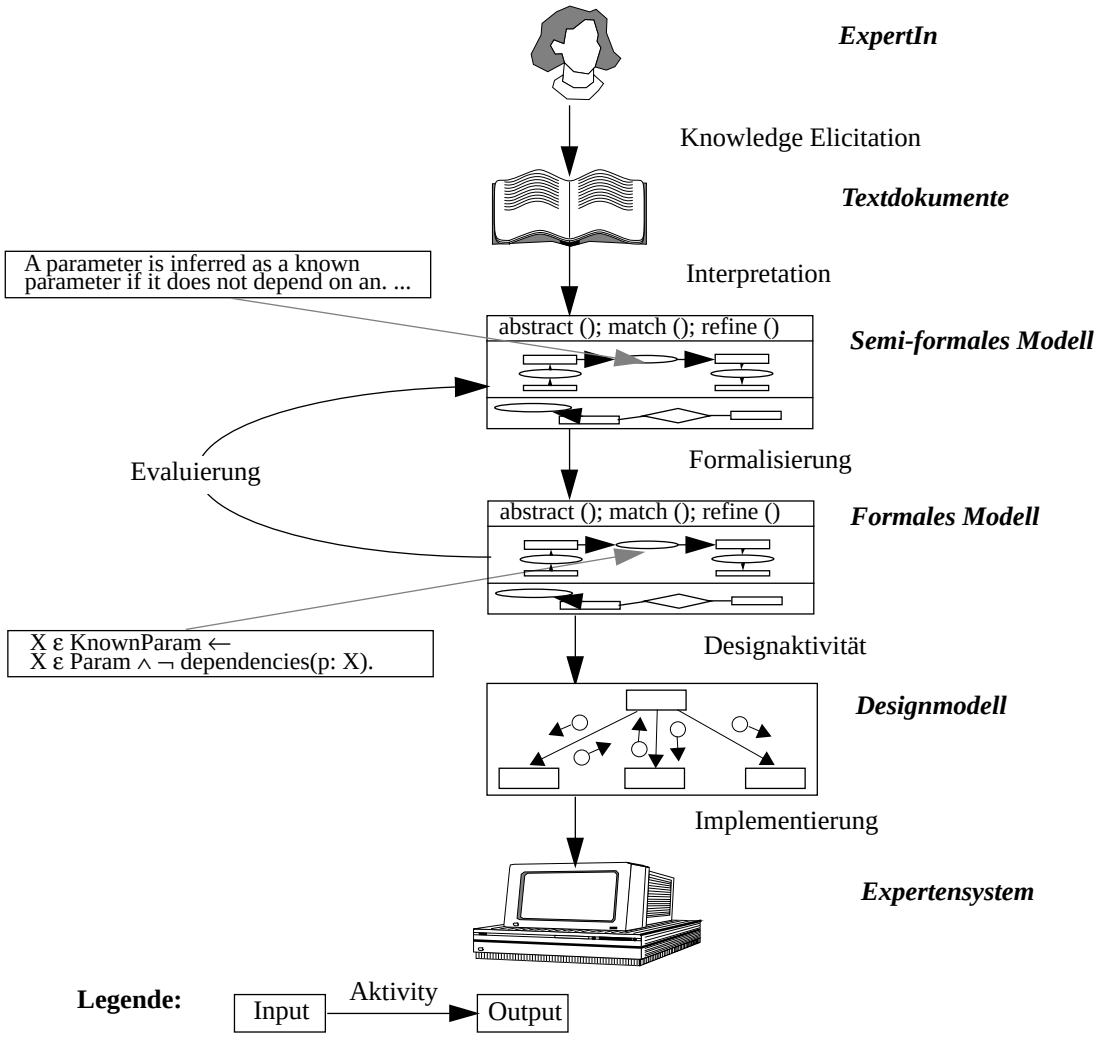

Abbildung 1. Modellbasierte Wissensakquisition

Ebene strukturiert und durch einen *semiformalen Beschreibungsformalismus* wie z.B. das KADS Model der Expertise repräsentiert. In einem dritten Schritt (Knowledge Formalization) werden die informalen, d.h. natürlich-sprachlichen, Beschreibungen durch *formale Ausdrücke* ergänzt. Dies ermöglicht einen *weichen* Übergang von semiformaler zu formaler Beschreibung (vgl. [FeN94]). Letztere ersetzt nicht die informalen Bestandteile der semiformalen Beschreibung, sondern ergänzt diese durch Präzision und Eindeutigkeit. Die Struktur der Modellierung, d.h. die semiformale Beschreibung des Modells der Expertise, und genauso dessen graphische Repräsentation, bleibt dabei erhalten. Das semiformal und formal spezifizierte Modell der Expertise ist dann der Ausgangpunkt für die Evaluierung des modellierten Wissens durch Inspektion, Ausführung und Beweistechniken. Die verschiedenen Aktivitäten Knowledge Elicitation, Knowledge Interpretation, Knowledge Formalization und Knowledge Evaluation wechseln sich dabei solange zyklisch ab, bis der Modellierungsprozeß ein Ergebnis der geforderten Güte geliefert hat.[23]

> *"This then, is the conclusion of the present discussion, that programs should be supported and specified by documentation of any kind, the overriding concern in producing this documentation being clarity to the people who have to deal with it.*

23. Weitere hier nicht thematisierte Aktivitäten sind Design und Implementierung des Systems.

For achieving clarity any formal mode of expression should be used, not as a goal in itself, but wherever it appears to be helpful to authors and readers alike." [Nau82]

Danksagung

Ich danke Jürgen Angele, Ernst-Erich Doberkat, Frank van Harmelen, Dieter Landes, Frank Maurer, Susanne Neubert, Rudi Studer, Jan Treur, Angi und Hans Voss sowie zwei anonymen Gutachtern für die Fülle hilfreicher Kommentare zu diesem Artikel.

Literatur

[Abe92] M. Aben: On the Specification of Knowledge Model Components. In *Proceedings of the 7th Knowledge Acquisition for Knowledge-Based Systems Workshop KAW´9*2, Banff, Canada, October 11-16, 1992.

[Abe93] M. Aben: Formally Specifying Reusable Knowledge Model Components. In *Knowledge Acquisition Journal*, vol 5, no 2, June 1993.

[AFL+93] J. Angele, D. Fensel, D. Landes, S. Neubert, and R. Studer: Model-Based and Incremental Knowledge Engineering: The MIKE Approach. In J. Cuena (ed.), *Knowledge Oriented Software Design*, IFIP Transactions A-27, Elsevier, Amsterdam, 1993.

[AFS90] J. Angele, D. Fensel, and R. Studer: Applying Software Engineering Methods and Techniques to Knowledge Engineering. In D. Ehrenberg et al. (eds.), *Wissensbasierte Systeme in der Betriebswirtschaft, Reihe betriebliche Informations- und Kommunikationssysteme, no 15*, Erich Schmidt Verlag, Berlin, 1990.

[AGM93] D. J. Andrews, J. F. Groote, C. A. Middelburg (eds.): *Preliminary Proceedings of the International Workshop on Semantics of Specification Languages SoSL*, Utrecht, The Netherlands, October 25-27, 1993.

[AHS93] H. Akkermans, F. van Harmelen, G. Schreiber, and B. Wielinga: A Formalisation of Knowledge-Level Models for Knowledge Acquisition. In *International Journal of Intelligent Systems, Special Issue Knowledge Acqisition as Modeling*, part II, no 2, vol 8, 1993.

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

[ARS92] S. Aitken, H. Reichgelt, N. Shadbolt: *Representing KADS models in QIL*, AI Group, University of Nottingham, Working Paper WP-006, 1992.

[Bar93] M. Barbuceanu: Models: Towards Integrated Knowledge Modeling Environments. In *Knowledge Acquisition*, vol 5, no 3, 1993.

[Ber87] D. C. Berry: The Problem of Implicit Knowledge. In *Expert Systems*, vol 4, no 3, August 1987.

[BFL+93] J. C. Bicarregui, J. S. Fitzgerald, P. A. Lindsay, R. Moore, B. Ritchie: *Proof in VDM: A Practitioner´s Guide*, Springer Verlag, Berlin, 1993.

[BHL90] D. Bjørner, C. A. R. Hoare, and H. Langmaack (eds.): *VDM´90. VDM and Z - Formal Methods in Software Development*, Lecture Notes in Computer Science, no 428, Springer-Verlag, Berlin, 1990.

[Bra89] I. Bratko: Fast Prototyping of Expert Systems Using Prolog. In G. Guida et al. (eds.), *Topics in Expert System Design, Methodologies and Tools*, North Holland, Amsterdam, 1989.

[CGR93] D. Craigen, S. Gerhart, and T. Ralston: *An International Survey of Industrial Applications of Formal Methods, vol 1 and 2*, U.S. Department of Commerce, National Institute of Standards and Technology, Gaithersburg, report NISTGCR 93/626, March 1993. Beziehbar über ftp nemo.ncsl.nist.gov, pub/ahis/formal_methods.

[DKS93] J.-M. David, J.-P. Krivine, and R. Simmons (eds.): *Second Generation Expert Systems*, Springer-Verlag, Berlin, 1993.

[EFN85] H. Ehrig, C. Floyd, M. Nivat, and J. Thatcher (eds.): *Formal Methods and Software Development, Proceedings of the International Joint Conference on Theory and Practice of Software Development (TAPSOFT)*, Berlin, March 25-29, 1985, Lecture Notes in Computer Science, no 186, vol 2, Springer-Verlag, Berlin, 1985.

[FAL+93] D. Fensel, J. Angele, D. Landes, and R. Studer: Giving Structured Analysis Techniques a Formal and Operational Semantics with KARL. In *Proceedings of Requirements Engineering ´93 - Prototyping* -, Bonn, April 25 - 27, 1993, Teubner Verlag, Stuttgart, 1993.

[FeN94] D. Fensel and S. Neubert: Integration Of Semiformal and Formal Methods For Specification of Knowledge-Based Systems. In *Proceedings of the GI-Fachgespräch Workshop-F1 Integration of Semiformal and Formal Methods, IFIP´94*, Hamburg, August 31, 1994.

[FBA+93] K. M. Ford, J. M. Bradshaw, J. R. Adams-Webber, and N. M. Agnew: Knowledge Acquisition as a Constructive Modeling Activity. In *International Journal of Intelligent Systems, Special Issue Knowledge Acqisition as Modeling*, part I, no 1, vol 8, 1993.

[FeH94] D. Fensel and F. v. Harmelen: A Comparison of Languages which Operationalize and Formalize

KADS Models of Expertise. In *The Knowledge Engineering Review*, vol 9, no 2, 1994.

[Fen92] D. Fensel: Knowledge Acquisition and the Interpretative Paradigm. In F. Schmalhofer et al. (eds.), *Contemporary Knowledge Engineering And Cognition*, First Joint Workshop, Kaiserslautern, Germany, February 21-22, 1991, in *Lecture Notes in Artificial Intelligence, no 622*, Springer-Verlag, Berlin, 1992

[Fen93a] D. Fensel: *The Reconciliation of Symbol and Knowledge Level*. In Forschungsbericht, no 266, Institut für Angewandte Informatik und Formale Beschreibungsverfahren, Unversität Karlsruhe, 1993.

[Fen93b] D. Fensel: *The Knowledge Acquisition and Representation Language KARL*, Ph.D thesis, University of Karlsruhe, 1993. To appear Kluwer Science Publ., 1995.

[Fen95a] D. Fensel: ECAI´94 Workshop on Formal Specification Methods for Knowledge-based Systems. To appear in *The Knowledge Engineering Review.*

[Fen95b] D, Fensel: Assumptions and Limitations of a Propose-and-Revise like Problem-Solving Method. Submitted 1995.

[Fet88] J.H. Fetzer: Program Verification: The Very Idea. In *Communications of the ACM*, vol 31, no 9, September 1988.

[Flo85] C. Floyd: On the Relevance of Formal Methods to Software Development. In [EFN85], pp. 1-11.

[FZB+92] C. Floyd, H. Züllighoven, R. Budde, R. Keil-Slawik: *Software Development and Reality Construction*, Springer-Verlag, Berlin, 1992.

[GaM94] B. Gaines and M. Musen (eds.): Proceedings of the 8th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop, vol 3, Banff/Canada, January 30 - February 4, 1994.

[GMB94] S. Greenspan, J. Mylopoulos, and A. Borgida: On Formal Requirements Modeling Languages: RML Revisited. In [ICSE94].

[GuD89] G. Guida and C. Tasso: Building Expert Systems: From Life Cycle to Development Methoddology. In G. Guida et al. (eds.), *Topics in Expert System Design, Methodologies and Tools*, North Holland, Amsterdam, 1989.

[HaB92] F. v. Harmelen and J. Balder: $(ML)^2$: A Formal Language for KADS Conceptual Models. In *Knowledge Acquisition*, vol 4, no 1, 1992.

[Hal90] A. Hall: Seven Myths of Formal Methods. In *IEEE Software*, vol 7, no 9, September 1990.

[Hei93] B. Heintz: *Die Herrschaft der Regel: zur Grundlagengeschichte des Computers*, Campus Verlag, Frankfurt am Main, 1993.

[HRS90] M. Heisel, W. Reif, and W. Stephan: Tactical Theorem Proving in Program Verification. In *Proceedings of the 10th International Conference on Automated Deduction*, Kaiserslautern, July 24-27, 1990, Lecture Notes in Artificial Intelligence, no 449, Springer Verlag, Berlin, 1990.

[HWL83] F. Hayes-Roth, D. Waterman, and D. Lenat (eds.): *Building Expert Systems*, Addison-Wesley, London, 1983.

[ICSE94] *Proceedings of the 16th International Conference on Software Engineering (ICSE´94)*, Sorrento, Italy, May 16-21, 1994.

[Joh93] S. R. Johansson: The Brain´s Software: The Natural Languages and Poetic Information Processing. In H. Haken et al. (eds.), *The Machine as Metaphor and Tool*, Springer-Verlag, Berlin, 1993.

[JoS92] W. Jonker and J.W. Spee: Yet Another Formalisation of KADS Conceptual Models. In *Proceedings of the 6th European Knowledge Acquisition for Knowledge-Based Systems Workshop (EKAW-92)*, May 18-22, Heidelberg/Kaiserslautern, T. Wetter et al. (eds.), *Current Developments in Knowledge Acquisition*, Lecture Notes in Artificial Intelligence, no 599, Springer-Verlag, Berlin, 1992.

[Kar93] W. Karbach: *MODEL-K: Modellierung und Operationalisierung von Selbsteinschätzung und -Steuerung durch Reflexion und Metawissen*, Ph. D. thesis, University of Bielefeld, Germany, 1993.

[KLW93] M. Kifer, G. Lausen, and J. Wu: *Logical Foundations of Object-Oriented and Frame-Based Languages*, Technical Report 93/06, Department of Computer Science, SUNY at Stony Brook, NY, April 1993. To appear in *Journal of the ACM*.

[LeG90] D. B. Lenat and R. V. Guha: *Building Large Knowledge-Based Systems: Representation and Inference in the CYC Project*, Addison-Wesley, Menlo Park, 1990.

[Lin92] M. Linster: *Knowledge Acquisition Based on Explicit Methods of Problem Solving*, Ph. D, thesis, University of Kaiserslautern, February 1992.

[Llo87] J.W. Lloyd: *Foundations of Logic Programming, 2nd Editon*, Springer-Verlag, Berlin, 1987.

[LPT93] Izak van Langevelde, A. Philipsen, and J. Treur: A Compositional Architecture for Simple Design Formally Specified in DESIRE. In [TrW93].

[Lyd92] T. J. Lydiard: Overview of Current Practice and Research Initiatives For The Verification And Validation of KBS. In *The Knowledge Engineering Review*, vol 7, no 2, 1992.

[MBG90] P. Mertens, V. Borkowski, and W. Geis: *Betriebliche Expertensystem-Anwendungen*, 2nd edition, Springer-Verlag, Berlin, 1990.

[Mea34] G.H. Mead: *Mind, Self and Society. From the standpoint of social behaviorist*, University of Chicago Press, 1934.

[Meh90] H. Mehrtens: *Moderne Sprache Mathematik: Eine Geschichte des Streits um die Grundlagen des Disziplin und des Subjekts formaler Systeme*, Suhrkamp Verlag, Frankfurt am Main, 1990.

[Mey85] B. Meyer: On Formalism in Specifications. In *IEEE Software*, vol 2, no 1, 1985.

[Mor87] K. Morik: Sloppy Modeling. In K. Morik (ed.), *Knowledge Representation and Organisation in Machine Learning*, Lecture Notes in Artificial Intelligence, no 347, Springer-Verlag, Berlin, 1987.

[Nau82] P. Naur: Formalization in Program Development. In *BIT*, vol 22, 1982.

[Nau85] P. Naur: Programming as Theory Building. In *Microprocessing and Microprogramming*, vol 15, 1985.

[New82] A. Newell: The Knowledge Level. In *Artificial Intelligence*, vol 18, 1982.

[Par86] D. Partridge: *Artificial Intelligence. Applications in the Future of Software Engineering*, Ellis Horwood, Chichester, 1986.

[Pop35] K. Popper: *Logik der Forschung*, Tübingen, 1989 (Originally appeared 1935). Zitiert nach [Hei93].

[RHA+94] F. Ruiz, F. van Harmelen, M. Aben, and G. van de Plaffche: Evaluating a Formal Modeling Language. In *Proceedings of the European Knowledge Acquisition Workshop (EKAW´94)*, Hoegaarden, Belgium, September 26-29, 1994, Lecture Notes in Aritificial Intelligence (LNAI), no 867, Springer Verlag, Berlin, 1994.

[RyS93] P. Ryan and C. Sennett: *Formal Methods in Systems Engineering*, Springer Verlag, Berlin, 1993.

[Spi88] J. M. Spivey: *Understanding Z: A Specification Language and its Formal Semantics*, Cambridge University Press, Cambridge, 1988.

[Ste90] L. Steels: Components of Expertise. In *AI Magazine*, vol 11, no 2, 1990.

[SWB93] G. Schreiber, B. Wielinga, and J. Breuker (eds.): *KADS. A Principled Approach to Knowledge-Based System Development*, Knowledge-Based Systems, vol 11, Academic Press, San Diego, 1993.

[TrW93] J. Treur and Th. Wetter (eds.): *Formal Specification of Complex Reasoning Systems*, Ellis Horwood, Chichester, 1993.

[VoV93] H. Voss and A. Voss: Reuse-Oriented Knowledge Engineering with MoMo. In *Proceedings of the 5th International Conference on Software Engineering and Knowledge Engineering (SEKE´93)*, San Fransisco Bay, June 14-18, 1993.

[Wet92] T. Wetter: *FORKADS: An Executable Language for the KADS Conceptual and Interpretation Models*, Habilitationsschrift, University of Kaiserslautern, Germany, 1992.

[Weu93] H. Weule: Expertensysteme im industriellen Einsatz. In *Proceedings 2. Deutsche Tagung Expertensysteme (XPS-93)*, Hamburg, February 17-19, 1993, F. Puppe et al. (eds.), *Expertensysteme 93*, Informatik aktuell, Springer-Verlag, Berlin, 1993.

[WiF86] T. Winograd and F. Flores: *Understanding Computers and Cognition*, Ablex Publishing Corporation, 1986.

[Win90] J. M. Wing: A Specifier´s Introduction to Formal Methods. In *IEEE Computer*, vol 23, no 9, September 1990.

[WoL93] J. C. P. Woodcock and P. G. Larsen (eds.): *FME´93: Industrial-Strength Formal Methods, Proceedings of the First International Symposium of Formal Methods Europe*, Odense, Denmark, April 19-23, 1993, Lecture Notes in Computer Science, no 670, Springer-Verlag, Berlin, 1993.

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

[Yue87] K. Yue: What Does It Mean To Say That A Specification Is Complete? In *Proceedings of the 4th International Workshop on Software Specification and Design*, April 3-4, Monterey, CA, USA, 1987.

[Zav91] P. Zave: An Insider´s Evaluation of PAISLey. In *IEEE Transactions on Software Engineering*, vol 17, no 3, March 1991.