Institut für Produktion und Industrielles Informationsmanagement

Universität Essen
Fachbereich 5: Wirtschaftswissenschaften
Universitätsstraße 9, D – 45141 Essen
Tel.: ++49 (0) 201/ 183–4006, Fax: ++49 (0) 201/ 183–4017

KOWIEN–Projektbericht 3/2002

Sprachen zur Repräsentation von Wissen

- eine untersuchende Darstellung -

Dipl.-Ing. Lars Dittmann

E-Mail: Lars.Dittmann@pim.uni-essen.de

KOWIEN
(“Kooperatives Wissensmanagement in Engineering-Netzwerken”)
wird mit Mitteln des Bundesministeriums für Bildung und Forschung
(BMBF) gefördert.
Förderkennzeichen Hauptband 02 PD1060.
Die Mitglieder des Projektteams danken
für die großzügige Unterstützung ihrer Forschungs- und Transferarbeiten.

Juli 2002

Inhaltsüberblick:

Seite

1. Definition Wissensrepräsentation 1

2. Allgemeine Ansätze zur Gliederung von Wissensrepräsentationen aus der KI 3

3. Spracharten 6

3.1. Natürliche Spracharten 6
3.1.1. Textuelle Spracharten 6
3.1.2. Ikonische Spracharten 7
3.1.3. Mischformen textuell/ikonisch 8

3.2. Formale Spracharten 9
3.2.1. Logische Sprachen 10
3.2.2. Mathematische Sprachen 18
3.2.3. Computersprachen 18
3.2.4. Programmiersprachen 24

3.3. Semi-formale Spracharten 26

4. Repräsentationssprachen für spezielle Wissensarten 26

4.1. Repräsentation von rein deklarativem Wissen 26
4.1.1. Objekt-Attribut-Wert Tripel 26
4.1.2. Prädikatenlogik 27
4.1.3. Semantische Netze 29
4.1.4. Frames (Rahmen) 30

4.2. Repräsentation von überwiegend prozeduralem Wissen 32

4.3. Repräsentation von Mischformen aus deklarativem/prozeduralem Wissen 32
4.3.1. ERM 32
4.3.2. Ereignisgesteuerte Prozessketten (EPKn) 34
4.3.3. Petri-Netze 36
4.3.4. Semantisches Objektmodell (SOM) 40

4.4. Kritische Betrachtung der Repräsentationen 42

5. Repräsentationssprachen in speziellen Verwendungszusammenhängen 43

5.1. Systematisierung von (Geschäftsprozess-)Modellen 43

5.2. Referenzmodelle aus betriebswirtschaftlicher Sicht 43

6. Anforderungen an Repräsentationssprachen aus Sicht von KOWIEN 44

7. Praktische Empfehlungen an Partnerunternehmen 45

8. Literaturverzeichnis 46

9. Anhang 50

Abbildungsverzeichnis

Abbildung 1: Wissensarten und Repräsentationsarten 4
Abbildung 2: Überblick Wissensrepräsentationsarten 5
Abbildung 3: Semantik der Aussagenlogik 12
Abbildung 4: Semantische Äquivalenzen 13
Abbildung 5: Exemplarische Axiom-Spezifikation in F-Logic 17
Abbildung 6: Exemplarische Anfrage-Spezifikation in F-Logic 17
Abbildung 7: Systematisierung Spezifikationssprachen 19
Abbildung 8: Einfaches Semantisches Netz 29
Abbildung 9: Frame eines Dreiecks 31
Abbildung 10: Einfaches Entity-Relationship-Diagramm in Anlehnung an Chen 33
Abbildung 11: Ablaufbeschreibung mit der EPK 35
Abbildung 12: Wirkung des Schaltens einer Transition 39
Abbildung 13: Interaction schema of business process distribution (4th level) 41

1. Definition Wissensrepräsentation

Wissen entspricht der Gesamtheit der Kenntnisse und Fähigkeiten, die Akteure zur Lösung von Problemen einsetzen.

Nachdem Wissen akquiriert worden ist, stellt sich die Frage, wie es repräsentiert werden soll, damit weiteres Wissen gewonnen werden kann bzw. erhalten werden kann. Als Wissensrepräsentation wird die sprachliche Rekonstruktion von Wissen und ihre Implementierung verstanden.[1] Die Repräsentation umfasst die Beziehung eines Begriffs zu dessen semantischer/pragmatischer Bedeutung als Objekt in der Realität oder Fiktion. Die Darstellungsart Formalisierung ist kennzeichnend für Wissensrepräsentation.

HAUN zählt die Wissensrepräsentation neben Problemrepräsentation, heuristischen Suchverfahren, Inferenzmaschinen und Lernen/Wissenserwerb zu den elementaren Methoden der Künstlichen Intelligenz (KI).[2]

In Bezug auf das Projekt KOWIEN und seine betriebswirtschaftliche Aufgabenstellung wird der Wissensrepräsentation im Folgenden die Aufgabe der Rekonstruktion und rechnergestützten Nutzbarmachung von explizitem und implizitem Wissen zugeschrieben.

Die in der Arbeitsdefinition für Wissen benutzten Begriffe, *Kenntnisse* und *Fähigkeiten* dienen zur Einteilung in zwei Wissensrepräsentationsextrema, die ihren Ursprung in einer Unterscheidung von RYLE[3] in „knowledge-that“ und „knowledge-how“ haben.

Das Wissen-Was des Akteurs entspricht *deklarativem* (Fakten- und Fach-)Wissen. Es umfasst die Kenntnisse einer Person (knowledge by aquaintance) über Begriffe, Objekte und ihre Relationen. Die deklarative Wissensrepräsentation lässt sich als Sammlung statischer Elemente verstehen. Das beschriebene Objektwissen kann hierarchisch oder netzwerkartig strukturiert sein. Vorteilhaft sind einfaches Ergänzen, Ändern und Entfernen von Elementen der Repräsentation. Nachteilig erscheinen jedoch die begrenzten Darstellungsmöglichkeiten, da beispielsweise eine chronologische Verknüpfung nicht stattfindet.

---

1) Vgl. GÖRTZ (1991) S.91.

2) Vgl. HAUN (2000) S. 21.

3) Vgl. RYLE (1949).

Demgegenüber steht das Wissen-Wie des Akteurs, das einen prozeduralen Charakter aufweist, denn es umfasst das Wissen über Handlungen/Aktivitäten. Die prozedurale Wissensrepräsentation lässt sich als Sammlung dynamischer Elemente zur Beschreibung von Handlungen (Methoden) und Aktivitäten (Abläufen) verstehen. Es umfasst das Wissen eines Akteurs[4] über „Anwendungshinweise“ des Wissens bereits in der Darstellung selbst. Es werden mit Hilfe des prozeduralen Wissens aufbauend auf Objektwissen Aufgaben gelöst und Ergebnisse abgeleitet.[5] Der Gebrauch des Wissens steht somit im Vordergrund dieser Repräsentationsform.

4) Akteur kann sowohl eine natürliche Person als auch eine Unternehmensorganisation sein.

5) Vgl. POCSAI (2000) S.18.

2. Allgemeine Ansätze zur Gliederung von Wissensrepräsentationen aus der KI

In Anlehnung an PUPPE[6] werden verschiedene Repräsentationsansätze kurz unterschieden:

- *Logik*. In der (Prädikaten-)Logik entsteht die Idee eines Kalküls daraus, dass man Objekte und Zustände der realen Welt durch Aussagen des Kalküls beschreiben und daraus mit allgemeingültigen Ableitungsregeln andere Aussagen herleiten kann, die dann wieder auf Objekte bzw. Zustände der Welt bezogen werden.[7]
- *Produktionsregeln.* Regeln bestehen aus einer „Vorbedingung" und einer „Aktion". Ist die Vorbedingung erfüllt, so wird die Aktion ausgeführt. Grundsätzlich lassen sich zwei Aktionsweisen unterscheiden. Zum einen wird durch „Induktionen" und „Deduktionen" der Wahrheitsgehalt einer Feststellung ermittelt, und zum anderen wird mit „Handlungen" ein Zustand verändert. Zur Anwendung von Regeln lassen sich zwei Möglichkeiten feststellen, die Vorwärtsverkettung (Forward-Reasoning) und die Rückwärtsverkettung (Backward-Reasoning). Das zu erhaltende Wissen stellen die Regeln dar. Die Datenbasis ist oftmals eine unstrukturierte und passive Menge von Fakten.
- *Framebasierter Ansatz.* In der Literatur auch bekannt als objektorientierter Ansatz. Im Gegensatz zum regelbasierten Ansatz spielen die Formalismen zur Strukturierung der Fakten die tragende Rolle, mit dem Ziel, Vererbungshierarchien, zugeordnete Prozeduren und Defaultwerte zu ermöglichen aus Gründen der Vereinfachung und Nachvollziehbarkeit. Ein „Frame" ist eine formularähnliche Objektbeschreibung (Datenstruktur) mit Merkmalen (Synoym: Objekteigenschaften, „slots") und Merkmalsausprägungen (Synonym: Werte, „values").[8]

Ostermayer unterscheidet, neben einer Unterteilung von deklarativ versus prozedural, vier Paradigmen der Wissensrepräsentation:[9]

---

6) PUPPE (1993).

7) POCSAI (2000), S. 23.

8) POCSAI (2000), S. 24.

9) OSTERMAYER (2001), S. 46 ff.

- *Assoziative Netze* mit den Vertretern Semantische Netze, Begriffliche (Conceptual) Graphen und Begriffliche Abhängigkeiten.
- *Strukturierte Objekte* mit den Vertretern Frames, Schemata, Units und Objekte.
- *Logikbasierte Repräsentationen* mit den Formen Aussagenlogik, Prädikatenlogik, Modale Logik, Mehrwertige (Multivalued) Logik und Unscharfe (Fuzzy) Logik sowie weitere Unterarten.
- *Prozedurale Repräsentation* wie zum Beispiel Produktionsregeln, Constraints und Scripts.

HAUN nennt in seiner Klassifikation in einer Art Baumstruktur in der obersten Ebene deklarativ und prozedural (sowie eingeschränkt die Hybris aus beiden). Die deklarativen Methoden unterteilt er wiederum in traditionelle datenmodellbasierte Methoden, objektorientierte Wissensrepräsentationsmethoden, regelbasierte Methoden und lexikalische Methoden. Zur prozeduralen Wissensrepräsentation zählt er insbesondere Agenten und Programme.[10]

Die Abbildung 1 zeigt eine weitere Differenzierungsmöglichkeit. Es wird hierbei hinsichtlich der Wissensarten, die repräsentiert werden sollen, unterschieden.

Wissensart	Repräsenatationsart
Faktenwissen	Texte, Tabellen, Diagramme, Schaubilder, Videos
Strukturwissen	Wissensbestands- und Wissensstrukturkarten, Organigramme
Personenwissen (Wissen über Wissen)	Wissensträgerkarten
Strategiewissen	Cognitive Maps (Kausalkarten)
Projektwissen	Vorgehensmodelle
Prozesswissen	Petrinetze, EPK, Ablauf- und Interaktionsdiagramme, Referenzprozessmodelle

Abbildung 1: Wissensarten und Repräsentationsarten

10) Vgl. HAUN (2000) S. 42.

Die Abbildung 2 (Seite 5) zeigt qualitativ eine mögliche Einteilung von Wissensrepräsentationsarten hinsichtlich ihres Verhältnisses von deklarativ zu prozedural. Hinzu kommt eine Unterscheidung in natürlichsprachlich und formalsprachlich. Es wird der Versuch unternommen, die genannten Wissensrepräsentationsarten hinsichtlich ihrer Spezifizierbarkeit in natürlichsprachliche respektive formalsprachliche Arten zu unterscheiden.[11]

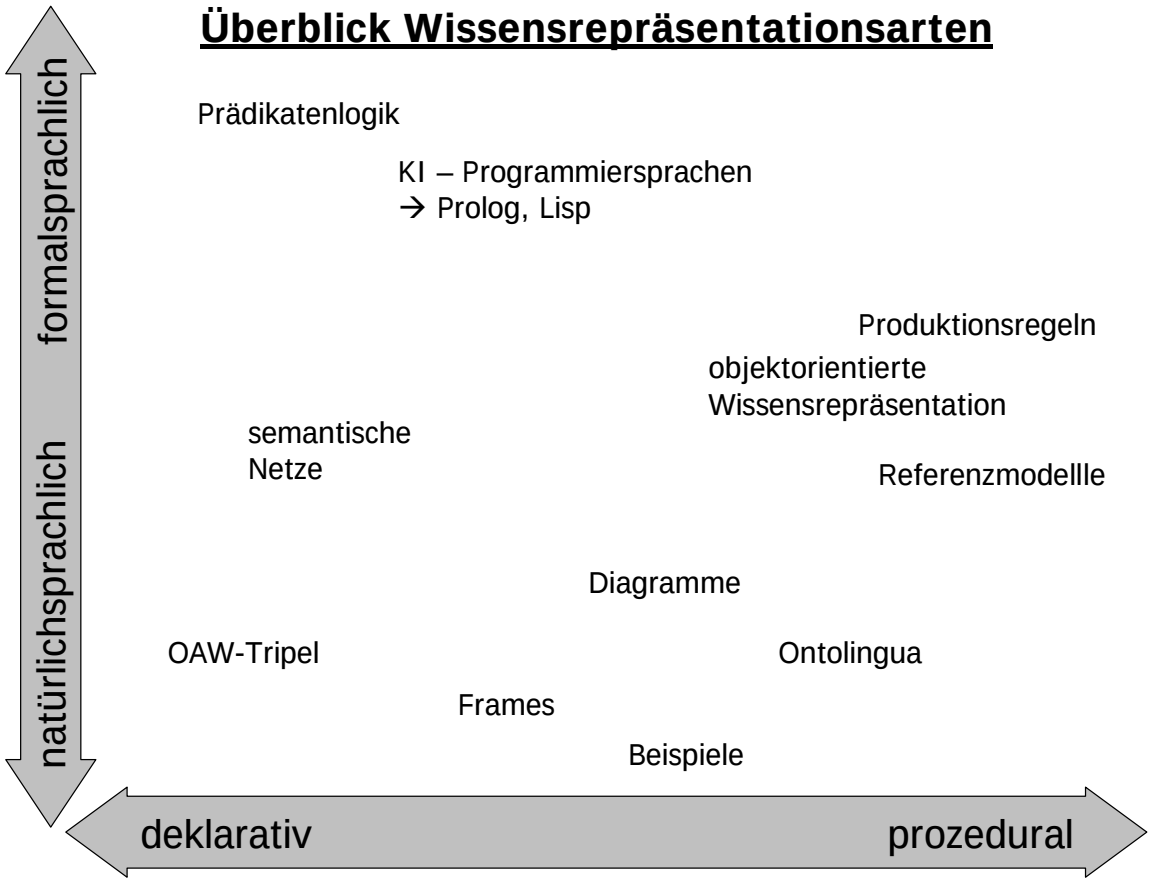

Abbildung 2: Überblick Wissensrepräsentationsarten

11) Zur Unterscheidung von natürlichen und formalen Spracharten siehe Kapitel 3 ab Seite.

3. Spracharten

Grundsätzlich ist jede Sprache durch zwei Aspekte charakterisiert: ihre Syntax und ihre Semantik. Die Syntax befasst sich zum einen mit dem Aufbau von Grundzeichen oder Grundausdrücken einer Sprache. Zum anderen mit der Erzeugung von Sätzen aus diesen Grundzeichen oder Grundausdrücken. Demgegenüber beschäftigt sich die Semantik mit der Bedeutung der Grundzeichen oder Grundausdrücke und den Bedingungen unter denen Sätze dieser Sprache als wahr angenommen werden.[12]

3.1. Natürliche Spracharten

Die natürlichen Spracharten werden in diesem Kapitel in textuelle und ikonische sowie deren Verbindung unterschieden. Denkbare wäre eine Ergänzung hinsichtlich mimisch-pantomimischer Bewegungen. Jedoch wurde hinsichtlich der KOWIEN-spezifischen Anwendbarkeit, die insbesondere eine maschinelle Erschließbarkeit voraussetzt, auf diese und mögliche weitere Arten (wie etwa eine mentale Darstellung) verzichtet.

Will man natürliche Sprache bis zu einem Detaillierungsgrad verstehen, der ihre mechanische Beherrschung erlaubt, so muss man sie als Formalismus mit einer bestimmten Semantik zu begreifen versuchen.[13] Dabei stößt man schnell auf die Problematik der Mehrdeutigkeit und der Ableitung von Semantik aus syntaktischen Aufbauten. Bibel weißt darauf hin, dass die Verarbeitung natürlicher Sprache ein eigenständiges Forschungsgebiet darstellt mit etwa gleicher Ausdehnung wie die Wissensrepräsentation.[14] Da jedoch die natürliche Sprache als Basis für jegliche Formen der Wissensrepräsentation angesehen werden muss, soll sie an dieser Stelle kurz erläutert werden.

3.1.1. Textuelle Spracharten

Die textuellen Spracharten findet man vorwiegend in gesprochener oder geschriebener Form.

---

12) Vgl. BECKERMANN (1997) S. 51.

13) Vgl. BIBEL (1993) S. 22.

14) Vgl. BIBEL (1993) S. 23.

Natürliche textuelle Sprache lässt sich in einem ersten Ansatz in Wissenschaftssprache und Alltagssprache unterscheiden. Die Eigenschaften der Alltagssprache ermöglichen eine Flexibilität, die Grundvoraussetzung von Sprache als ein Kommunikationsmittel im Alltag ist. Beispielsweise sollen Empfänger vom Sender bewusst im Unklaren gelassen werden in bestimmten Situationen. Dies geschieht durch Mehrdeutigkeit, unscharfe Bedeutung oder Synonyme. Wissenschaftssprache hingegen muss eindeutig und genau sein. Hierfür wird eine festgelegte Terminologie verwendet, die Eindeutigkeit, Verkürzung und Übersicht ermöglicht. Die Wissenschaftssprache hat die Alltagssprache zur Grundlage. Die Gesamtheit der sprachlichen Mittel einer Wissenschaft mit den Regeln für deren Gebrauch nennt man Wissenschaftssprache.[15]

Eine symbolisierte Sprache ist eine Sprache, deren Zeichen (=Symbole) künstlich geschaffen oder mit einer bestimmten neuen Bedeutung versehen wurden.[16] Formalisierte Sprache enthält darüber hinaus eine festgelegte, wohlgeformte Syntax die festgelegten Operationsregeln folgt und eine eindeutige Semantik umfasst. Insbesondere in Teilgebieten der formalen Logik und der Mathematik werden formale Sprachen eingesetzt (siehe hierzu auch Kapitel 3.2, Seite 9).

Textuelle Spracharten findet man sowohl alltagssprachlich als auch wissenschaftssprachlich in Freitexten (Reports, Handbüchern, Gebrauchs-/Verfahrensanweisungen usw.) und strukturierten Texten (Konten, Bilanzen usw.). Nicht immer lassen sich Texte einwandfrei einer der vorgenannten Textklassen zuordnen. So muss an dieser Stelle darauf hingewiesen werden, dass zum Beispiel Tabellen einer Kombination von freiem und strukturiertem Text entsprechen.

3.1.2. Ikonische Spracharten

Die visuelle Repräsentation in Form von ikonischer Darstellung gilt als die ursprünglichste aller Darstellungsarten, wie zum Beispiel die Höhlemalereien, die Wissen über Jagdmethoden enthalten, aus der Steinzeit belegen. Die Informationsmenge, die der Mensch bei Anblick einer Grafik aufnimmt übersteigt signifikant die Menge an Informationen, die man textuell aufnehmen kann. Aufgrund ihres Informationsgehaltes spielen ikonische Spracharten eine wichtige Rolle bei der Wissensrepräsentation

15) Vgl. KLAUS (1971).

16) Vgl. KLAUS (1971)

und insbesondere bei der Wissensakquisition (siehe hierzu Projektbericht Nr. 2 des Projekts KOWIEN). Nachfolgend werden gängige ikonische Darstellungsformen aufgelistet und charakterisiert, um den nachfolgenden Arbeitspaketen eine möglichst vollständige Aufzählung zur Verfügung zu stellen.[17]

▪ Piktogramme	→ Wörtlich übersetzt: Bildzeichen.
▪ topographische Karten	→ Eine topographische Karte ist eine verkleinerte Projektion der Erdoberfläche auf eine horizontale Ebene und stellt die Oberflächengestalt der Erde dar (Morphologie). Außerdem werden anthrophogene und natürliche Inhalte die sich auf die Erdoberfläche befinden dargestellt.
▪ thematische Karten	→ Stellen verschiedenste Sachverhalte in einen geographischen Bezug (z.B. Arbeitslosenstatistik oder Mitspiegel).
▪ Konstruktionszeichnungen	→ Hierunter fallen alle ingenieursmäßig erstellten Konstruktionszeichnungen, die gängigen Normen und Standards genügen (DIN, ISO,...).
▪ Graphen	→ Netze, Bäume.
▪ Schaltpläne für elektrotechn. Produkte	→ Ähnlich Konstruktionszeichnungen, jedoch um eine Fluß-Dimension erweitert.
▪ Schaubilder/ Diagramme	→Dienen zur Veranschaulichung von Sachverhalten (z.B. aus der Statistik).
▪ Fotografien	→ Halten die reale Darstellung eines Objekts/Sachverhalts aus der Vergangenheit vor.

3.1.3. Mischformen textuell/ikonisch

Die Mischformen enthalten sowohl textuelle Elemente als auch ikonische. Sie sind als Synthese anzusehen und versuchen ein Optimum an eindeutigen, intersubjektiv nachvollziehbaren Informationen darzustellen. Wo die alleinige ikonische Darstellung nur bedingt ausreicht, wird textuell geholfen um alle gewünschten Informationen abzubilden.

17) Eine umfangreichere Liste, die sich nicht auf ikonische Darstellungsarten beschränkt, mit der Zielstellung der Computerverarbeitbarkeit findet sich im Anhang dieses Berichts.

▪ beschriftete Karten	→ Es werden hier Karten verstanden, die einen übermäßig großen (>= 50%) textuellen Teil besitzen und ohne diesen unverständlich bleiben.
▪ Organigramme	→ Dienen zur Darstellung eines Organisationsgefüges i. d. R. mittels grafischer Knoten und Kanten.
▪ Mind Maps	→ Einzelne textuelle Informationen oder Daten werden mittels grafischer Hilfsmittel in einen Kontext gesetzt.

Ferner sind an dieser Stelle auch Formate wie etwa Videos (also alle Bewegtbildaufnahmen mit Ton) zu subsumieren. Aufgrund ihrer äußerst schwierigen Maschinenverarbeitbarkeit wird an dieser Stelle jedoch nicht weiter auf diese Möglichkeiten der Repräsentation von Wissen eingegangen werden.

3.2. Formale Spracharten

Unter formalen Spracharten wird die Kombination von logischen und nicht-logischen Symbolen zu Formeln, deren Bedeutung nur von der äußeren Gestalt (der „Form“) der Symbole/Formeln abhängen, verstanden. Die inneren Bedeutungszuweisungen bleiben hiervon ausgenommen.

Ein formales System stellt ein Symbolsystem dar, in dem vordefinierte Operationen informationstechnisch ausgeführt werden können, ohne dass man als Benutzer wissen muss, wofür die Symbole in Wirklichkeit stehen.

Die Sprache innerhalb eines formalen Systems wird Kalkül genannt. Es wird in interpretiert und uninterpretiert unterschieden. Ein Kalkül wird uninterpretiert genannt, dessen Formationsregeln oder syntaktische Regeln, festgesetzt, jedoch dessen Ausdrücke (Semantik) noch nicht interpretiert worden sind. Ein interpretiertes Kalkül beinhaltet hingegen sowohl Regeln für die Syntax als auch für die Semantik. Eine formalisierte Sprache entspricht einem interpretiertem Kalkül.

Die formalen Spracharten werden an dieser Stelle in logische Sprachen, mathematische Sprachen und Computersprachen differenziert. Die logischen Sprachen sind untergliedert in Aussagenlogik, Prädikatenlogik, Tensorlogik, Modallogik und F-Logic. Aussagenlogik aufgrund ihrer Basisbedeutung und F-Logic aufgrund ihrer speziellen Bedeutung hinsichtlich Computerverarbeitbarkeit (und nicht zuletzt ihrer Bedeutung für das Projekt KOWIEN) werden jeweils ausführlicher dargestellt. Die mathemati-

schen Sprachen werden untergliedert in Arithmetik, Mengenlehre, Differentialkalkül und Algebraen. Ausführlicher werden die Computersprachen behandelt, sie werden in einer ersten Gliederungsebene in Spezifikationsprachen, Programmiersprachen und Annotationssprachen unterschieden.

Das anschließende Kapitel widmet sich den vornehmlich semi-formalen Spracharten.

3.2.1. Logische Sprachen

Zu den logischen Sprachen gehört insbesondere die Aussagenlogik, die im Folgenden eingehender vorgestellt wird. Anschließend wird F-Logic als die ihr am nächsten stehende computerverarbeitbare Sprache behandelt. Weitere Anwendungen der Logik sind die Tensorlogik und die Modallogik. Auf die Prädikatenlogik und ihre Besonderheit für das Projekt KOWIEN wird in Kapitel 4.1.2 (S. 27) näher eingegangen.

Die Aussagenlogik

Aussagen sind Grundbestandteil der Aussagenlogik. Hierunter fallen sämtliche Sätze, die *wahr* oder *falsch* sein können. Entsprechend wird die Aussagenlogik als *2-wertig* bezeichnet.[18] Der Wahrheitsgehalt einer Aussage ergibt sich aus der Vorlage des ausgedrückten Sachverhaltes. Die Untersuchung der Aussagenlogik wird hier zum einen hinsichtlich ihrer *Syntax* erfolgen. Hierunter fallen die Fragen nach den *Grundzeichen* und dem *Aufbau* der Aussagen aus den Grundzeichen. Zum anderen erfolgt eine Untersuchung der *Semantik* der Aussagenlogik. Untersuchungsgegenstand sind hierbei die *Bedeutung* der Grundzeichen und die *Bedingungen* für den Wahrheitsgehalt der Aussagen.

Hinsichtlich der verwendeten Grundzeichen lässt sich eine Unterscheidung treffen in *deskriptive* und *logische* Zeichen.[19] Unter die deskriptiven Grundzeichen fallen sämtliche Zeichen, die einen Sachverhalt repräsentieren. So kann z.B. der Sachverhalt,

---

18) Vgl. RAUTENBERG (1996) S. 1. Die *Fuzzy-Logik* stellt eine Sonderform der Logik dar, die ohne das Axiom der 2-wertigkeit auskommt. Vgl. hierzu z.B. HEINSOHN/AMBROSIUS (1999) S. 177 ff; HENNINGS (1991) S. 103 ff. Im Gegensatz zur 2-wertigen Logik erstreckt sich hierbei die Interpretation einer Aussage auf den Wertbereich [0.0,1.0]. Aufgrund ihrer als zu gering zu erwartenden Bedeutung der Fuzzy-Logik im Rahmen des KOWIEN-Projektes wird hier nicht näher darauf eingegangen werden. Dennoch werden im Projekt die Optionen freigehalten werden, im weiteren Verlauf eine Integration des *probabilistischen* und *possibilistischen Schließens* in die Forschungsarbeiten aufzunehmen.

19) Zusätzlich zu den deskriptiven und logischen Zeichen werden in der Aussagenlogik die verwendeten Klammern `(´ und `)´ als *Hilfszeichen* bezeichnet. Vgl. BECKERMANN (1997) S. 52.

dass der Mitarbeiter Müller die Programmiersprache Java beherrscht, durch *M* dargestellt werden.[20] Unwichtig ist hierbei die Buchstabenwahl. Der dargestellte Sachverhalt hätte ebenso durch das Zeichen *J* dargestellt werden können. Lediglich bei dem Versuch mehrere Sachverhalte in einem Satz auszudrücken ist darauf zu achten, nicht das gleiche Zeichen für verschieden Sachverhalte zu benutzen. Es ließe sich z.B. der Satz "Müller kann Java und C++" als *M* formalisieren. Versucht man allerdings die Aussage in zwei Teilaussagen zu unterteilen, müssen die beiden Teile durch verschiedene Zeichen dargestellt werden (z.B. *J und C*). Die Benennung beider Teilaussagen mit *M* und die daraus resultierende Konstellation *M und M* ist somit für diesen auszusagen versuchten Sachverhalt ungültig. Die in der Aussagenlogik verwendeten logischen Zeichen werden auch als *Junktoren* bezeichnet. Diese sind:

1. die Negation `¬´ ("nicht"),
2. die Konjunktion `∧´("und"),
3. die Disjunktion `∨´ ("oder"),
4. die Subjunktion `→´ ("wenn...dann...") und
5. die Bisubjunktion `↔´ ("genau dann... wenn...").[21]

Die Junktoren stellen logische Konstanten dar, mit denen *atomare Formeln* verknüpft werden können, um komplexere Aussagen zu erhalten. Die einheitliche Interpretation komplexer Aussagen wird gesichert durch *Bindungs- bzw. Vorrangregeln.*[22] Demnach werden die Junktoren in der obigen Reihenfolge interpretiert. So ist beispielsweise die Formel $A \vee B \wedge C$ zu interpretieren als "A oder (B und C)". Ebenso dürfen Außenklammern der Form $(A \circ B)$[23] weggelassen werden. Durch *Rechtsklam-*

---

20) Es wird abgesehen von teilweise unterschiedlich verwendeten Notationen. So ist z.B. in BECKERMANN (1997) S. 52 vorgesehen, die Satzbuchstaben mit einem tiefgestellten Index zu versehen.

21) Weitere Junktoren, die sich auf die bereits vorgestellten Junktoren zurückführen lassen, werden hier nicht explizit vertieft. (Vgl. RAUTENBERG (1996). S. 3 ff; HEINSOHN/SOCHER-AMBROSIUS (1999) S. 76) Hierunter fallen die Antivalenz (`+´, "entweder...oder..."), die Nihilition (`↓´, "weder...noch...") und die Unverträglichkeit (`↑´, "nicht zugleich...und..."). Ihre besondere Bedeutung erhalten diese Junktoren im Rahmen der Konstruktion von Schaltungsentwürfen in der Digitaltechnik. Aufgrund ihres häfigen Einsatzes in der Literatur wurde allerdings die Subjunktion eingeführt, obwohl auch sie sich mit den anderen Junktoren darstellen ließe. So entspricht die Subjunktion (1) $A \rightarrow B$ der Disjunktion (2) $\neg A \vee B$. Zur Verdeutlichung: Die Aussage "Wenn der Mitarbeiter die Java-Schulung besucht hat, dann hat der Mitarbeiter Kenntnisse in Java" ist *semantisch äquivalent* mit der Aussage "Der Mitarbeiter hat die Java-Schulung nicht besucht oder der Mitarbeiter hat Kenntnisse in Java".

22) Vgl. HEINSOHN/ROCHER-ABROSIUS (1999) S. 77; RAUTENBERG (1996) S. 5.

23) Das Zeichen `∘´ stelle hier einen beliebigen Junktor dar.

merung wird erreicht, dass eine Formel der Form $A \rightarrow B \rightarrow C$ gelesen wird als $A \rightarrow (B \rightarrow C)$.[24]

Es lässt sich zusammenfassend festhalten, dass zum Aufbau von Sätzen entweder die Bedingung erfüllt sein muss, dass das verwendete Zeichen selbst eine atomare Formel darstellt oder dass sich die Formel aus einer Negation, Konjunktion, Disjunktion, Subjunktion oder Bisubjunktion von zwei anderen Zeichen ergibt.

Die Semantik der Aussagenlogik gibt zum einen eine Antwort auf die Frage nach der Bedeutung der Zeichen. Zum anderen fallen hierunter die Bedingungen, unter denen die Sätze wahr sind. Die Bedeutung der deskriptiven Zeichen entspricht ihrer *Interpretation I*. In umgangssprachlichem Gebrauch entspricht der Abbildung dessen, was der "Satz besagt". Sie entspricht der Abbildung der atomaren Aussagen auf die möglichen Wahrheitswerte *wahr* und *falsch*. Wenn *I* z.B. dem Zeichen *M* das zuordnet, was der Satz "Meier kann Java" besagt, dann heißt das, dass *M* bezüglich *I* dasselbe aussagt, wie das Meier Java kann. Somit liegt die Bedeutung der in der Aussagenlogik verwendeten Zeichen geknüpft an die *Übereinstimmung* der Sätze mit den "Tatsachen".[25]

Die *extensionalen* Definitionen zusammengesetzter Aussagen können in Wahrheitstafeln[26] oder –matrizen[27] erstellt werden. Die nachfolgende Abbildung gibt die möglichen Interpretationen für zusammengesetzte Aussagen wieder.

A	¬A
0	1
1	0

A	B	A∨B	A∧B	A→B
0	0	0	0	1
0	1	1	0	1
1	0	1	0	0
1	1	1	1	1

Abbildung 3: Semantik der Aussagenlogik

24) Ebenso ist es möglich, zusammengesetzte Aussagen durch die polnische Notation ohne Klammern auszudrücken (Vgl. BUCHER (1987) S. 127). Demnach kann z.B. die Konjunktion ($A \wedge B$) ausgedrückt werden durch *Kab*. Dieser Ansatz wird hier nicht weiter verfolgt werden, da seine Vorteile sich größtenteils bei der Programmierung in älteren Programmiersprachen entfalten.

25) Das hier zu Grunde gelegte *Korrespondenz-Konzept* für *Wahrheit* geht zurück auf Aristoteles. In späteren Arbeiten u.a. Wittgensteins wurde eine Verfeinerung des Begriffs der *Übereinstimmung* vorgenommen. Verworfen wurde hierbei u.a. die später als naiv postulierte Hypothese der "ganzheitlichen Abbildungsbeziehung" zwischen Satz und Tatsachen(Vgl. CZAYKA (2000). S. 68 f). Stattdessen nahm der Begriff der "Struktur-Identität" eine Sonderstellung ein.

26) Vgl. CZAYKA (2000) S. 7; BUCHER (1987) S. 61 ff; HEINSOHN/ROCHER-AMBOSIUS (1999) S. 76.

27) Vgl. RAUTENBERG (1996) S. 3.

Sind die Interpretation *I* einer Aussage stets wahr, wird sie als *allgemeingültig* oder *tautologisch* bezeichnet. Die Formel F ∨ ¬F ist z.B. stets wahr. Belegt werden kann dies dadurch, dass in der Wahrheitstafel unter der Formel immer eine 1 sein wird. Der entgegen gesetzte Fall entspricht einer Wahrheitstafel, in der unter der Formel nur Nullen wären. Dieser Fall wird als eine *Kontradiktion* bezeichnet. Ein Beispiel solch einer Formel stellt A ∧ ¬A dar.

Die – bereits oben angesprochene – semantische Äquivalenz syntaktisch unterschiedlicher Formeln ist dann gegeben, wenn sämtliche Interpretationen der Formel identisch sind. Ihre Bedeutung erlangt die semantische Äquivalenz insbesondere bei der automatischen Formalisierung natürlichsprachlich vorliegender Sätze oder der Umformung bereits formalisierter Sätze. Die im Rahmen der Aussagenlogik bekannten semantischen Äquivalenzen[28)] lassen sich der Abbildung 4 entnehmen.[29]

$(A \rightarrow B)$	$\equiv$	$\neg A \vee B$	Implikation
$\neg(A \vee B)$	$\equiv$	$\neg A \wedge \neg B$	Gesetze von DeMorgan
$\neg(A \wedge B)$	$\equiv$	$\neg A \vee \neg B$	
$\neg\neg A$	$\equiv$	A	Doppelte Negation
$A \vee A$	$\equiv$	A	Idempotenzgesetze
$A \wedge A$	$\equiv$	A	
$A \wedge (A \vee B)$	$\equiv$	A	Absorptionsgesetze
$A \vee (A \wedge B)$	$\equiv$	A	
$A \wedge B$	$\equiv$	$B \wedge A$	Kommutativgesetze
$A \vee B$	$\equiv$	$B \vee A$	
$A \wedge (B \wedge C)$	$\equiv$	$(A \wedge B) \wedge C$	Assoziativgesetze
$A \vee (B \vee C)$	$\equiv$	$(A \vee B) \vee C$	
$A \wedge (B \vee C)$	$\equiv$	$(A \wedge B) \vee (A \wedge C)$	Distributivgesetze
$A \vee (B \wedge C)$	$\equiv$	$(A \vee B) \wedge (A \vee C)$	

Abbildung 4: Semantische Äquivalenzen

28) Die semantischen Äquivalenzen – hier dargestellt durch das Symbol `≡´ entsprechen einer bikonditionalen Verknüpfung von Formeln durch `↔´.

29) Vgl. HEINSOHN/ROCHER-AMBOSIUS (1999) S. 79.

Zwecks Überprüfung der semantischen Äquivalenz bietet sich u.a. die Umformung in eine *Normalform* an. Eine *konjunktive Normalform* ist gegeben bei einer Konjunktion von Disjunktionen von Literalen.[30] Die Formel $(A \vee B) \wedge (C \wedge D)$ ist z.B. in konjunktiver Normalform. Analog hierzu ist die *disjunktive Normalform* bei einer Disjunktion von Konjunktionen von Literalen gegeben .

Die Aussagenlogik gestattet es, anhand der logischen *Struktur* der Sätze gültige Folgerungen aus ihnen abzuleiten. Die "Konservierung" der Wahrheit bei Schlussfolgerungen wird durch *Deduktion*[31)] erreicht. Einige Regeln der Deduktion lauten:

1. Modus (ponendo) ponens

Regel	Beispiel
$A \rightarrow B$	Wenn der Mitarbeiter an der Java-Schulung teilgenommen hat, dann hat er Kenntnisse in Java.
A	Der Mitarbeiter hat an der Java-Schulung teilgenommen
B	Der Mitarbeiter hat Kenntnisse in Java.

2. Modus (tollendo) tollens

Regel	Beispiel
$A \rightarrow B$	Wenn der Mitarbeiter an der Java-Schulung teilgenommen hat, dann hat er Kenntnisse in Java.
$\neg B$	Der Mitarbeiter hat keine Kenntnisse Java.
$\neg A$	Der Mitarbeiter hat nicht an der Java-Schulung teilgenommen

3. Simplifikation

Regel	Beispiel
$A \wedge B$	Der Mitarbeiter hat Kenntnisse in Java und der Mitarbeiter hat Kenntnisse in C++
A	Der Mitarbeiter hat Kenntnisse in Java

30) Als *Literale* werden die atomaren Formeln oder ihre Negationen verstanden. Komplementät sind Literale der Form A und $\neg A$. Entsprechend gilt z.B. $\bar{A} = \neg A$.

31) Vgl. Bucher (1987) S. 87; Hennings (1991) S. 44;

4. Konjunktion

Regel	Beispiel
A	Der Mitarbeiter hat Kenntnisse in Java
B	Der Mitarbeiter hat Kenntnisse in C++
$A \wedge B$	Der Mitarbeiter hat Kenntnisse in Java und der Mitarbeiter hat Kenntnisse in C++

5. Hypothetischer Syllogismus

Regel	Beispiel
$A \rightarrow B$	Wenn der Mitarbeiter an der Java-Schulung teilgenommen hat, dann hat er Kenntnisse in Java.
$B \rightarrow C$	Wenn der Mitarbeiter Kenntnisse in Java hat, dann ist er einsetzbar in Projekt X.
$A \rightarrow C$	Wenn der Mitarbeiter an der Java-Schulung teilgenommen hat, dann ist er einsetzbar in Projekt X.

6. Disjunktiver Syllogismus

Regel	Beispiel
$A \vee B$	Der Mitarbeiter hat Kenntnisse in Java oder der Mitarbeiter hat Kenntnisse in C++.
$\neg A$	Der Mitarbeiter hat keine Kenntnisse in Java.
B	Der Mitarbeiter hat Kenntnisse in C++.

7. Konstruktives Dilemma

Regel	Beispiel
$(A \rightarrow B) \wedge (C \rightarrow D)$	Wenn der Mitarbeiter an der Java-Schulung teilgenommen hat, dann hat er Kenntnisse in Java und wenn der Mitarbeiter an der C++-Schulung teilgenommen hat, dann hat er Kenntnisse in C++.
$A \vee C$	Der Mitarbeiter hat an der Java-Schulung teilgenommen oder der Mitarbeiter hat an der C++ Schulung teilgenommen
$B \vee D$	Der Mitarbeiter hat Kenntnisse in Java oder der Mitarbeiter hat Kenntnisse in C++.

8. Destruktives Dilemma

Regel	Beispiel
$(A \rightarrow B) \wedge (C \rightarrow D)$	Wenn der Mitarbeiter an der Java-Schulung teilgenommen hat, dann hat er Kenntnisse in Java und wenn der Mitarbeiter an der C++-Schulung teilgenommen hat, dann hat er Kenntnisse in C++.
$\neg B \vee \neg D$	Der Mitarbeiter hat keine Kenntnisse in Java oder der Mitarbeiter hat keine Kenntnisse in C++.
$B \vee D$	Der Mitarbeiter hat Kenntnisse in Java oder der Mitarbeiter hat Kenntnisse in C++.

- F-Logic

F-Logic ist eine Sprache zur Repräsentation von Wissen über Konzepte und ihre Relationen. Die Spezifikation von F-Logic entstammt der Forschung aus dem Themengebiet der deduktiven Datenbanken.[32] Eine besondere Bedeutung hat F-Logic für das Projekt KOWIEN, da die im Rahmen des Projektes verwendete Inferenzmaschine *Ontobroker* und die Ontologie-Entwicklungsumgebung *OntoEdit* in ihren derzeitigen Version über sie kompatibel sind.[33] F-Logic stellt eine Kombination der Repräsentationsarten Frames (Siehe hierzu Seite 30) und Prädikatenlogik (Siehe hierzu Seite 27) dar.[34] Es kann in einer Faktenbasis Wissen über die Domäne abgelegt werden, anhand derer mit Inferenzregeln "neues" Wissen expliziert werden kann. Die Syntax von F-Logic lässt sich wie folgt darstellen:

- Is-a Beziehungen zwischen Konzepten in einer Domäne können durch zwei aufeinanderfolgende Doppelpunkte dargestellt werden. So kann z.B. die Kenntnis das "Mitarbeiter" eine Subklasse von "Mensch" ist durch "Mitarbeiter::Mensch" dargestellt werden.
- Klassenzugehörigkeiten von Instanzen können durch Doppelpunkte dargestellt werden. Die Zugehörigkeit von "Schmitz" zur Klasse "Mitarbeiter" wird entsprechend mit "Schmitz:Mitarbeiter" dargestellt.

32) Vgl. KIFER /LAUSEN (1995)

33) Vgl. o.V. (2002). Es ist geplant, in zukünftigen Versionen von OntoEdit den Export von Ontologien in *Resource Description Framework* (RDF) zu ermöglichen.

34) Vgl. DECKER (1998).

- Prädikate, die auf eine ganze Klasse angewendet werden, können in der Form O[P=>>S] modelliert werden, wobei O die Klasse der Objekte angibt, auf die das Prädikat P mit den Extension der Klasse S anwendbar ist.
- Prädikate, die auf ein Element (Instanz) einer Klasse angewendet werden, können analog in der Form I[P->>B].

Um aus der nach obigem Schema definierten Wissensbasis implizites Wissen inferieren zu können, können in F-Logic *Regeln* (Axiome) angewandt werden. So kann z.B. die Regel, dass angenommen werden kann, jeder Mitarbeiter, der an einer Java-Schulung teilgenommen hat müsse Kenntnisse in objektorientierten Programmiersprachen haben wie folgt definiert werden:

```
FORALL X X[Kenntnis_in->>Java] <--
X:Mitarbeiter[Teilnahme_an_Schulung->>Y] and Y:Schulung[Themengebiet->>Java].
```

Abbildung 5: Exemplarische Axiom-Spezifikation in F-Logic

Durch den Ontobroker kann nun eine explizite Anfrage an die Wissensbasis gestellt werden, obwohl das Wissen nicht explizit in der Wissen spezifiziert wurde. So würde beispielsweise die Anfrage aus Abbildung als Ergebnis den Mitarbeiter "Schmitz" angeben, wenn in der Wissensbasis lediglich seine Teilnahme an der Java-Schulung deklariert wurde.

```
FORALL X,Y <-- X:Mitarbeiter[Kenntnis_in->>Java].
```

Abbildung 6: Exemplarische Anfrage-Spezifikation in F-Logic

Wie zu sehen ist, stellt auch die Anfrage (Query) eine Inferenzregel dar.

Die Mächtigkeit der Sprache F-Logic liegt in den durch sie gebotenen Möglichkeiten der Vererbung von Klasseneigenschaften auf Subklassen. So wird das Prädikat *Name* der Klasse *Mitarbeiter* auf sämtliche Subklassen vererbt. Das heißt, das bei einer weiteren Untergliederung der Klasse *Mitarbeiter* in *wissenschaftliche Mitarbeiter* und *administrative Mitarbeiter* beide Klassen das Prädikat *Name* erben würden. Weiterführend erlaubt F-Logic allerdings auch multiple Vererbungen, so das die Klasse *studentische Hilfskraft* sowohl die Eigenschaften der Klasse *Mitarbeiter* (z.B. Eintrittsdatum) als auch die der Klasse *Studenten* (z.B. Matrikelnummer) übernimmt.

Durch *Parameter* ist es möglich, Prädikate die auf Objekte angewendet werden, mit Zusatzinformationen zu versorgen. Möchte man z.B. die Reihenfolge der Publikatio-

nen angeben, die ein Autor herausgegeben hat, so kann das in folgender Form geschehen: X[hat_Publikation@(1)->>Y; hat_Publikation@(2)->>Z].

Als nachteilig an F-Logic ist die mangelnde Kardinalitätsfunktion anzusehen, so kann lediglich festgelegt werden, ob eine Relation 1:n und n:n mal existiert. Eine Angabe 2:5 ist demnach nicht möglich.

3.2.2. Mathematische Sprachen

Die Gruppe der mathematischen Sprachen umfasst ohne die Formen der Logik insbesondere die folgenden Felder.

- Arithmetik
- Mengenlehre
- Differentialkalkül
- Algebraen

Sie sind jedoch für das Projekt KOWIEN, insbesondere wenn man den Fokus Kompetenzprofile anführt, nur von untergeordneter Rolle und werden deshalb an dieser Stelle nur der Vollständigkeit halber erwähnt.

3.2.3. Computersprachen

Der bewusst allgemein gehaltene Begriff „Computersprachen“ wird in diesem Abschnitt untergliedert in Spezifikationssprachen, Programmiersprachen und Annotationssprachen.

Spezifikationssprachen

Eine erste Gliederung von Spezifikationssprachen kann anhand des zu spezifizierenden Objekts vorgenommen werden. Es wird dabei in Anforderungen an Informationssysteme, Abläufe in Informationssystemen und die Spezifikation von Wissensinhalten unterschieden. Die Abbildung ?? zeigt eine tabellarische Übersicht hierzu. Die bereits erwähnte Dreiteilung wird als Matrix in Level, Sparte und Sprachen aufgespannt. Als Level wird die Ebene des einzelnen Ansatzes bezeichnet. Die Anforderungen an Informationssysteme betreffen somit die Ebene der Organisation innerhalb der ein (wissensbasiertes) System erstellt werden soll. Die Sparte bezeichnet das Feld der Informatik innerhalb derer die anzuwendenden Techniken entwickelt und vorgehalten werden. In der letzten Zeile werden beispielhaft einige Sprachen die hierbei Anwendung finden aufgezählt.

Auf SADT (Structured Analysis and Design Technique) 1977 von ROSS[35] vorgestellt und andere Sprachen bzw. Methodologien die Sprachen enthalten, wird an dieser Stelle nicht weiter eingegangen, weil der Fokus des Requirements Engineering nur wenige Vorteile für das Projekt KOWIEN verspricht.[36]

	Anforderungen an Informations-systeme	**Abläufen in Informations-systemen**	**Spezifikation von Wissensinhalten**
Level	organizational level	symbol processing level	knowledge level
Sparte	Requirements Engineering	Software Engineering	Knowledge Engineering
Sprachen	SADT	Petri-Netze ERM	allgemeine vs. spezielle Spezifikations-sprachen

Abbildung 7: Systematisierung Spezifikationssprachen

Petri-Netze und ERM werden bereits im Kapitel 4.3 ausführlich dargelegt hinsichtlich ihrer Möglichkeiten für KOWIEN. Aus den gleichen Gründen (nur mit umgekehrtem Vorzeichen) wird im folgenden Text näher auf die allgemeinen und die speziellen Spezifikationssprachen eingegangen. Bei den allgemeinen Spezifikationssprachen wird insbesondere auf CommonKADS, KIF, XML, RDF und UML eingegangen. Die speziellen Spezifikationssprachen werden mit der Forderung „ontologiebasiert“ (Kernthema des Projekts KOWIEN) ausgewählt und vorgestellt. Weitere Spezifikationssprachen für Zustandsveränderungen (term rewriting systems, Produktionsregeln) oder Konzeptzusammenhänge (Entity Relationship-Modell, conceptual schemata, conceptual graphs) werden entweder an anderer Stelle (Entity Relationship-Modell, Kap. 4.3) oder aufgrund zeitlicher Restriktionen gar nicht erläutert.

35) Vgl. ROSS (1977).

Allgemeine Spezifikationssprachen:

- CommonKADS (Common Knowledge Analysis and Design Support)

CommonKADS stellt im eigentlichen Sinne keine allgemeine Spezifikationssprache dar, vielmehr wird es als eine modellbasierte Methodologie angesehen. Es bedeutet eine Sammlung informeller Darstellungen und Definitionen verschiedener Konzepte.[37] In seiner langen Entwicklungsgeschichte wurden jedoch einige Sprachen speziell für die KADS-Thematik entwickelt. So beschreiben Fensel und van Harmelen bereits 1994 acht formale Sprachen und vergleichen diese miteinander.[38] Es sind dies FORKADS, KARL, $K_{BS}SF$, $(ML)^2$, Model-K, Momo, OMOS und QIL. Die Autoren kommen zu dem Schluß, dass, die den Sprachen zugrunde gelegten unterschiedlichen Zielsetzungen und ihre Komplementarität auf formaler und operationaler Ebene, es zulässt, die Sprachen zu kombinieren und bei verschiedenen Stufen innerhalb des Knowledge-Engineering Prozesses anzuwenden.[39]

- Knowledge Interchange Format (KIF)

Knowledge Interchange Format (KIF) wurde zum Austausch von *Wissen* zwischen ungleichartigen Programmen entwickelt. Es enthält eine deklarative Semantik (die Bedeutung der Ausdrücke in der Repräsentation wird verstanden ohne zuvor auf einen Interpreter zurückgreifen zu müssen), beinhaltet Elemente der Prädikatenlogik erster Ordnung , es unterstützt die Repräsentation von Metawissen und unterstützt die Definition von Objekten, Funktionen und Relationen. Die Sprache soll als Mediator zwischen anderen Sprachen eingesetzt werden und somit eine Übersetzungsfunktion erfüllen. Die Sprachbeschreibung enthält sowohl eine Spezifikation für die Syntax als auch für die Semantik. Der KIF-Kern ähnelt sehr stark F-Logic.[40]

---

36) Zur Thematik des Requirements Engineering siehe auch: PARTSCH (1991).

37) Siehe hierzu auch: SCHREIBER (2001).

38) Vgl. FENSEL (1994).

39) FENSEL/HARMELEN (1994) S. 144ff.

40) Für weitere Informationen siehe auch: http://www-ksl.stanford.edu/knowledge-sharing/kif/, und http://logic.stanford.edu/kif/kif.html, Zugriff am 9.4.2002.

- eXtensible Markup Language (XML)

XML (eXtensible Markup Language) ist aus SGML hervorgegangen und vom W3C Consortium als Standard empfohlen.[41] Sie stellt eine Metasprache, die insbesondere den Erfordernissen des WWW Rechnung trägt, dar. SGML (und somit auch XML) dient als Metasprache zur Definition von Auszeichnungssprachen zur Erstellung von Dokumenten. Eine Auszeichnungssprache dient lediglich zur Formatierung von Texten. Metasprachen gehen darüber hinaus, da sie auch Inhalte strukturieren. Ein in einem XML-Dokument enthaltenes Element muss in einer Document Type Definition (DTD) aufgeführt und definiert werden. Die DTD kann dabei intern auch als extern zum Dokument enthalten sein. Es besteht die Möglichkeit Tags zu definieren, die anschließend einem Agenten ermöglichen die Semantik einzelner Dokumentpassagen zu erfassen und weiterzuverarbeiten. Eine bekannte Anwendung von XML stellt RDF dar, das im folgenden Abschnitt definiert wird.

- Resource Description Framework (Schema)

RDF(S)[42] ist eine Entwicklung und Standardisierung des World Wide Web Consortium (W3C). Ziel der Entwicklung war es die unterschiedlichen Standards, hinsichtlich der Syntax von Metadaten und möglichen Beschreibungssprachen für Schemas, zu vereinen auf dem Weg von der Maschinenlesbarkeit zur Maschinenverstehbarkeit. RDF enthält eine Syntax-Spezifikation (RDF) und eine Schema-Spezifikation (RDF-S). RDF besitzt seit 1999 den Status einer W3C Recommendation, während RDF-S bis heute lediglich als Recommendation Candidate eingestuft wird. Zusammengenommen bildet RDF(S) eine Infrastruktur zur Codierung, den Austausch und die Wiederverwendung von strukturierten Metadaten. Insbesondere ist dies sinnvoll für Suchmaschinen, intelligente Agenten, Informationsbroker, Browser und nicht zuletzt für menschliche Nutzer. Es bietet die Möglichkeit semantische Informationen maschinell zu verarbeiten. RDF ist eine Applikation von XML. RDF-Schema erlaubt Entwicklern Klassen (*Classes*) von Ressourcentypen (*Resource Types*) und Eigenschaften (*Properties*) zu spezifizieren um Beschreibungen dieser Klassen, Beziehungen zwischen Eigenschaften und Klassen und Einschränkungen (*Constraints*) bezüg-

41) Vgl. http://w3.org/XML/ , für Tutorials siehe: http://www.programmingtutorials.com/tutorial.asp?id=XML, Zugriff am 9.4.2002.

42) Vgl. http://w3.org/RDF/, Zugriff am 9.4.2002.

lich erlaubter Kombinationen von Klassen, Eigenschaften und Werten zu übermitteln.[43]

- Universal Modeling Language (UML)

Die Frage: Was ist UML? beantwortet die Object Management Group (OMG), die UML standardisiert hat, folgend:[44]

UML defines twelve types of diagrams, divided into three categories: Four diagram types represent static application structure; five represent different aspects of dynamic behavior; and three represent ways you can organize and manage your application modules.

Structural Diagrams include the Class Diagram, Object Diagram, Component Diagram, and Deployment Diagram.

Behavior Diagrams include the Use Case Diagram (used by some methodologies during requirements gathering); Sequence Diagram, Activity Diagram, Collaboration Diagram, and Statechart Diagram.

Model Management Diagrams include Packages, Subsystems, and Models.

UML soll als einheitlicher Standard zur Modellierung in einer objektorientierten Sichtweise fungieren. Mittels XMI (XML Metadata Interchange, einem weiteren OMG-Standard), ist es möglich, die in einem Tool erstellten Daten in ein weiteres einzulesen und weiterzuverarbeiten. UML enthält weiterhin die Sprache Object Constraint Language (OCL), auf die aus Zeitgründen nicht näher eingegangen werden kann.

Spezielle Spezifikationssprachen

Spezielle Spezifikationssprachen dienen insbesondere zur Spezifikation terminologischer Apparate/ sprachlicher Ausdrucksmittel u.ä.. An dieser Stelle werden einige beispielhaft aufgeführt die besonders in Hinsicht auf ontologische Unterstützung angewandt werden (die somit auch die Abbildung von Ontologien gemäß der Definition von ZELEWSKI erlauben). Es sind dies Ontolingua, Loom, CycL, DAML+OIL. RDF(S) und F-Logic finden sich an anderer Stelle der in diesem Arbeitsbericht vorgenommenen Klassifikation (Seite 21 und Seite 16), weil sie entweder keine Spezifikationssprache darstellen sondern ein Framework und somit eine allgemeine Sprache darstellen oder weil sie näher zu den logischen Sprachen gezählt werden und im eigentlichen Sinne nicht ausschließlich zur Konstruktion von Ontologien genutzt werden.

---

43) Vgl. http://www.w3.org/TR/rdf-schema/ , Zugriff am 4.4.2002.

Ontolingua

Ontolingua[45] wurde 1992 vom Knowledge Systems Lab (KSL) der Stanford University vorgestellt. Die Sprache basiert auf KIF. Sie unterstützt die Repräsentation von Konzepten, Taxonomien von Konzepten, Relationen, Funktionen, Axiomen, Instanzen und Prozeduren. Die große Ausdrucksstärke führte zu großen Problem hinsichtlich der Erstellung von Schlußfolgerungsmechanismen. Auch ein kürzlich erschienener Theorem-Beweiser für KIF-Ausdrücke konnte hieran nichts ändern.[46]

Loom

Loom (LOOM 1991[47]) ist ein LISP-basierter Spross der KL-ONE Familie. Es gilt als sehr ausdrucksstarkes und schnelles System. Es wurde an der University of South California entwickelt. Es basiert auf Description Logics und Produktionsregeln. Es erlaubt die Repräsentation von Konzepten, Taxonomien, Relationen, Funktionen, Axiomen und Produktionsregeln.

CycL

CycL[48] ist eine formale Sprache deren Semantik sich urprünglich aus der Prädikatenlogik erster Ordnung ergab, mittlerweile jedoch weit darüber hinaus geht. In order to express real-world expertise and even just plain old common sense knowledge, however, it goes far beyond first order logic. The vocabulary of CycL consists of terms: semantic constants, non-atomic terms (NATs), variables, numbers, strings, etc. Terms are combined into meaningful CycL expressions, ultimately forming meaningful closed CycL sentences (with no free variables).[47]

---

44) Vgl. http://www.omg.org/gettingstarted/what_is_uml.htm, Zugriff am 9.4.2002.

45) Vgl. http://ontolingua.stanford.edu, Zugriff am 9.4.2002.

46) Vgl. CORCHO (2002), S.52.

47) Vgl. LOOM GUIDE (1991).

48) Vgl. http://www.cyc.com/cycl.html, Zugriff am 10.4.2002.

DAML+OIL

DAML+OIL[49] stellt eine spezielle semantische Markup Language für Ressourcen des Webs dar. DAML+OIL stellt eine gemeinsame Entwicklung für das Semantic Web dar.

Es baut auf RDF(S) auf und erweitert diese mit mehr Modellierungsmöglichkeiten. Es enthält Modellvorgaben die vornehmlich in framebasierten und description logic Sprachen Verwendung finden. Es erlaubt die Repräsentation von Konzepten, Taxonomien, binären Relationen, Funktionen und Instanzen. Die neueste Version enthält Verarbeitungsmöglichkeiten von XML Schema datatypes.[44]

3.2.4. Programmiersprachen

Programmiersprachen können in einem ersten Konzept in implementierungsunabhängig und implementierungsabhängig unterschieden werden. An dieser Stelle werden jedoch nur die beiden implementierungsunabhängigen Sprachen LISP und PROLOG kurz erläutert, weil sie weitreichende Bedeutung für die Repräsentation von Wissen erlangt haben.

1. LISP

LISP[50] (LISt Processing) wurde Ende der 50er Jahre von John McCarthy entwickelt. 1994 wurde Common Lisp der erste offizielle ANSI-Standard und gilt heute neben SCHEME als die weitverbreiteste Variante. Allgemein gilt LISP als eine programmierbare Programmiersprache die im Laufe ihrer Evolution sogar objektorientierte Möglichkeiten herausbilden konnte. LISP ist weiterhin sehr verbreitet und kompatibel zu den meisten gängigen Programmiersprachen. An dieser Stelle soll jedoch nicht weiter auf LISP eingegangen werden, weil die Kernkompetenzen von LISP nicht in der Darstellung von Wissen gesehen werden.

2. PROLOG

Ein PROLOG-Programm (PROgramming in LOGic) besteht aus Fakten und Regeln, die die Wissensbasis darstellen. Fakten beschreiben hierbei wahre Aussagen in einer spezifischen Domäne. Regeln beschreiben jeweils logische Implikationen. In einem

---

49) Vgl. http://www.daml.org/language, Zugriff am 10.4.2002.

50) Um einen umfassenden Einblick zu erhalten siehe: http://www.lisp.org.

zweiten Schritt werden Anfragen an die Wissensbasis aus dem PROLOG-System gestellt. Die Antwort wird durch einen Resolutionsbeweiser zu beweisen versucht und mit yes oder no falsifiziert. Als weitere Hauptbestandteile eines PROLOG-Systems sind die Unifikation, die Rekursion, Listen, Operatoren, Negation und Identität zu nennen.

Annotationssprachen

Um Wissen innerhalb eines Computercodes über diesen selbst verfügbar zu machen, bedient man sich in der Programmierung der Annotation. Ziel der Annotation kann sowohl Mensch als auch Maschine sein. In der Regel wird bei der Zielrichtung Mensch versucht den Code zu einem späteren Zeitpunkt einem Dritten nachvollziehbar zu erhalten, d.h. es werden beispielsweise Erklärungen zu verwendeten Abkürzungen annotiert. In jüngerer Zeit rückt der Computer oder auch der nichtmenschliche Agent in den Fokus der Zielrichtung. So entwickelte zum Beispiel MÜLLER[51] 1995 die Annotationssprache SALSA für die objektorientierte Programmiersprache SATHER auf der Grundlage der Design-by-Contract-Theorie für den Softwareentwurf. Annotationen sind hierbei Vor- und Nachbedingungen von Methoden der objektorientierten Programmiersprache sowie Klasseninvarianten. Sie sind somit formalisierte, in den Programmcode eingebundene Verträge.[52] Weitere Annotationssprachen ähnlicher Art sind LARCH und EIFFEL.

Interessanter für das Projekt KOWIEN erscheinen die Annotationssprachen für die Skriptsprachen des Internets und hier besonders die ontologiebasierten Sprachen. Das Projekt ONTOWEB[53] untersucht unter anderem ontologiebasierte Annotations-Software mit den Annotationssprachen[54] DAML+OIL, RDF, F-Logic und SHOE. Es wird hierbei versucht dem Computer „Semantik“ beizubringen, er soll in die Lage versetzt werden einer Zeichenkette eine Bedeutung zuzuweisen, so dass beispielsweise eine verbesserte Suchfunktion zu erwarten ist. Weitere Annotationssprachen in diesem Bereich stellen A-HTML und OCL dar.

---

51) Vgl. MÜLLER (1995).

52) Vgl. MÜLLER (1995) S.8.

53) Siehe hierzu http://www.ontoweb.org, Zugriff am 3.4.2002.

54) Nach Ansicht der Autoren ist die Subsumierung der aufgelisteten Sprachen unter dem Oberbegriff Annotationsprachen nicht zulässig, da es sich strenggenommen nicht um solche handelt.

Die oben kurz erwähnte Zielrichtung Mensch spielt für das Projekt KOWIEN lediglich eine untergeordnete Rolle und wird an dieser Stelle nicht weiter verfolgt.

3.3. Semi-formale Spracharten

Als dritten und letzten Punkt gilt es die semi-formalen Spracharten zu untersuchen. Sie stellen eine Zwischenform der zwei bereits Behandelten dar. Grundsätzlich lassen sich wiederum zwei Gruppen differenzieren. Zum einen die natürlichen Sprachen mit formalsprachlichen Ergänzungen, hierzu zählen etwa Fachtexte mit Formeln und HTML-Dokumente.

Zum anderen deren Umkehrung, die formale Sprachform mit natürlichsprachlichen Ergänzungen, hierzu zählen etwa Computerprogramme mit Erläuterungstexten und beispielsweise Petrinetze mit natürlichsprachlichen Kanten-/Knotenbeschriftungen.

Für das Projekt KOWIEN und die Zielstellung dieses Arbeitsberichts werden die semi-formalen Spracharten als vernachlässigbar angesehen, weil sie gerade aus natürlichsprachlichen und formalsprachlichen Elementen bestehen und somit schon an anderer Stelle ausreichend behandelt wurden.

4. Repräsentationssprachen für spezielle Wissensarten

4.1. Repräsentation von rein deklarativem Wissen

Die wichtigsten Methoden der deklarativen Wissensrepräsentation sind die Prädikatenlogik, semantische Netze und Frames.[55] Diese Aufzählung wird an dieser Stelle noch um die Objekt-Attribut-Wert Tripel ergänzt, die eine grundlegende Systematisierung darstellen, die in der elektronischen Datenverarbeitung und der Modellierung quasi einen Basisstandard darstellen.

4.1.1. Objekt-Attribut-Wert Tripel

Objekt-Attribut-Wert Tripel dienen zur Darstellung des Informationsgehalts von Fakten. Objekte repräsentieren hierbei Gegenstände oder Begriffe. Attribute beschreiben die spezifischen Eigenschaften der Objekte. Werte geben die spezifische Ausprägung der Attribute wieder.

Analog repräsentieren Semantische Netze und (relationale) Datenbanken einen Sachverhalt als eine Menge von Objekt-Attribut-Wert Tripeln. Beispielsweise hat das

55) HAUN (2000), S. 83.

das Objekt *Monitor* das Attribut *Bildschirm* mit dem Wert *TFT*. In der Programmiersprache PROLOG ließe sich dieses Tripel in der Form Monitor (Bildschirm, TFT) darstellen.

4.1.2. Prädikatenlogik

Obwohl die Aussagenlogik es erlaubt, teilweise komplexe Aussagen zu modellieren, ist es mit ihrer Hilfe nicht möglich, Aussagen über eine ganze Klasse von Objekten zu treffen. Versuchte man beispielsweise die Sätze "Schmitz kann Java", "Jeder, der Java kann, hat Kenntnisse in objektorientierten Programmiersprachen" und "Schmitz hat Kenntnisse in objektorientierten Programmiersprachen" zu modellieren, so kommt man mit den Zeichen A,B und C aus. Allerdings ist es nicht möglich, von dem Inhalt der Aussagen A und B auf C zu schließen. Die Prädikatenlogik stellt eine Sprache dar, die aufbauend auf den Erkenntnissen der Aussagenlogik u.a. diese Möglichkeiten bietet. Analog zu der Vorgehensweise bei der Aussagenlogik wird auch hierbei zunächst die Syntax der Prädikatenlogik und anschließend ihre Semantik erläutert werden.

Die syntaktischen Elemente der Prädikatenlogik sind *Konstanten*, *Variablen*, *Quantoren*, und *Prädikate*.[56] Konstanten werden in der Prädikatenlogik zur Kennzeichnung von Individuen aus der Objektklasse benutzt. Sie werden hier mit den Buchstaben a,b,c,..., gekennzeichnet. Mit Variablen (x,y,z,) können ebenso Aussagen über die ganze Klasse gemacht werden. Sie können unterschieden werden in freie und gebundene Variablen. Die Gebundenheit der Variablen ergibt sich aus ihrer Vorkommnis in einer Teilformel einer Formel, der ein Quantor vorgestellt ist. Quantoren sind die Zeichen `∀´ und `∃´. `∀´ heißt *Allquantor*, was etwa dem deutschen "für alle" entspricht. Der *Existenzquantor* `∃´ kann übersetzt werden mit "es gibt mindestens ein". Die bei der Anwendung der Quantoren entstehen Sätze werden entsprechend als *Allsätze* bzw. *Existenzsätze* bezeichnet[57]. Prädikate entsprechen etwa dem umgangssprachlichen Gebrauch von *Prädikat*. Mit ihnen werden Aussagen *über* Objekte ge-

56) Vgl. HEINSOHN/ROCHER-AMBOSIUS (1999) S. 94. Hinzuweisen ist hierbei darauf, das Prädikate, Quantoren und Funktionen auch Variablen i.e.S. darstellen. Somit reduziert sich die Menge der verwendeten Elemente auf Konstanten und Variablen.

57) Vgl. CZAYKA (2000) S. 36.

macht bzw. "prädiziert".[58] Dies kann in Form von Verben, Substantiven oder Adjektiven erfolgen. Ebenso ist eine Kombination dieser möglich.

Anhand von Prädikaten können einstellige Aussagen über Objekte getroffen werden. So kann beispielsweise die Aussage "Schmitz ist ein Mitarbeiter" durch die Formel *Ms* ausgedückt werden. Um das eingangs vorgestellte Problem lösen zu können bietet die Prädikatenlogik aber auch die Möglichkeit der *mehrstelligen* Prädikate. So können z.B. Aussagen wie "Schmitz ist Mitarbeiter im Projekt X" durch eine Formel der Form *Msx*[59] ausgedrückt werden. Im weitern können Aussagen mit Hilfe der logischen Junktoren verbunden werden. So kann z.B. der Satz "Schmitz ist Mitarbeiter in den Projekten X und Y" ausgedrückt werden durch die Konjunktion Msx $\wedge$ Msy. Die gleichen Möglichkeiten gelten auch für die restlichen Junktoren. Die Formalisierung der Schlussfolgerung kann wie folgt aussehen:

1. Js	"Schmitz kann Java"
2. $(\forall x)$ (Jx $\rightarrow$ Kxo)	"Jeder, der Java kann, hat Kenntnisse in objektorientierten Programmiersprachen"
3. Kso	"Schmitz hat Kenntnisse in objektorientierten Programmiersprachen" (Schlussfolgerung aus Modus ponens)

Die Semantik der Prädikatenlogik ergibt sich – analog zu der Aussagenlogik – aus der Interpretation ihrer Sätze. So wird die Aussage obige *Msx* dann wahr, wenn Schmitz "tatsächlich" im Projekt X arbeitet. Jedes Prädikat kann in der Prädikatenlogik *extensional*[60] mittels Konstanten, auf die sich das Prädikat bezieht, definiert werden. Das Prädikat *Mitarbeiter* ist in der *Diskurswelt* (Domäne) eines bestimmten Unternehmens extensional definiert durch alle Mitarbeiter, die im Unternehmen beschäftigt sind. Allgemein lässt sich festhalten, das der Wahrheitsgehalt eines Satzes Mx, wobei M das Prädikat und x eine Konstante darstelle, dann gegeben ist, wenn das von x bezeichnete Objekt ein Element der Extension des Prädikats M ist. Analog lässt sich dies auf mehrstellige Prädikate anwenden.

---

58) Vgl. RUPPEN (1997) S. 157.

59) Teilweise wird in der Literatur eine andere Notation verfolgt: die Prädikate in atomaren Sätzen mit einem hochgestellten Index, der die jeweilige Stellenzahl angibt und einem tiefgestellten Index, der das Unterscheidungskriterium darstellt (Vgl. BECKERMANN (1997) S. 170; MATES (1978) S. 68).

60) Die Extension eines Prädikats entspricht der Menge aller existenten Objekte die durch das Prädikat beschrieben werden können. Hingegen die Intension eines Prädikats beinhaltet sämtliche möglichen Objekte (Vgl. SCHWEIZER S. 48 ff).

4.1.3. Semantische Netze

Ein Semantisches Netz[61] ist ein Modell, das aus einer definierten Menge von begrifflichen Entitäten und der zwischen diesen bestehenden Beziehungen besteht.[62] Es verfügt über lediglich zwei formale Elemente, den Knoten und den gerichteten Kanten. Mit Hilfe der Knoten werden Objekte, Konzepte oder Ereignisse dargestellt, die Kanten repräsentieren die Beziehungen zwischen diesen. Abhängig vom Anwendungsgebiet werden verschiedenartige Beziehungen dargestellt. Besonders hervorzuheben sind jedoch die „ist-ein“ und die „hat“-Beziehung, die eine Spezialisierungs- oder eine Elementbeziehung ausdrücken.[63] Der Satz: „Afghanistan ist ein Land und hat ein Problem.“ wird in Abbildung 8 als ein einfaches semantisches Netz zur Veranschaulichung verdeutlicht.

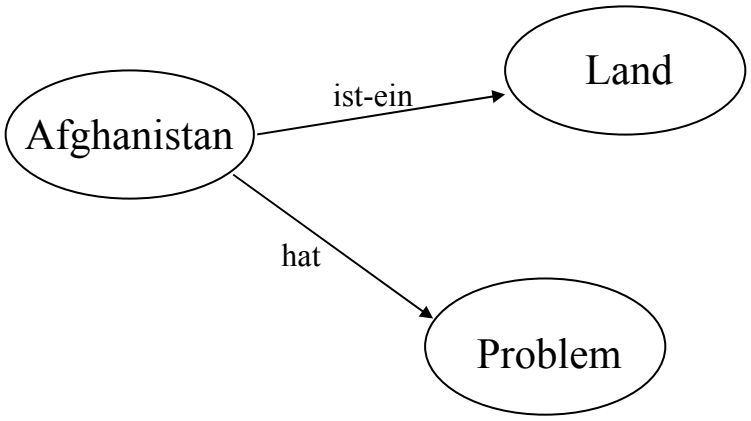

Abbildung 8: Einfaches Semantisches Netz

Semantische Netze und prädikatenlogische Formeln repräsentieren die gleiche Information in unterschiedlichem Format (Knoten entsprechen Termen und markierte gerichtete Kanten entsprechen Prädikaten).[64] Der Nachteil Semantischer Netze liegt in der schnell zu umfangreichen und unübersichtlichen Darstellung komplexerer Sachverhalte. Dieses Problem verspricht das Konzept der Frames im nachfolgenden Abschnitt zu lösen.

---

61) BIBEL (1993) spricht in diesem Zusammenhang auch von assoziativen Netzen.

62) Vgl. HAUN (2000) S. 66.

63) Vgl. Silberbusch (1990) S. 167.

64) Vgl. WACHSMUTH (2000) S. 3.

4.1.4. Frames (Rahmen)

MINSKY schreibt als Quintessenz seines *Frame*-Konzepts: Wenn jemand auf eine neue Situation stößt (oder jemand vollzieht einen substantiellen Sichtwechsel auf ein gegenwärtiges Problem), so selektiert er aus seinem Gedächtnis eine Struktur die *Frame* bezeichnet wird. Es handelt sich hierbei um einen Rahmen (Framework) aus der Erinnerung, der adaptiert wird um der Realität zu entsprechen durch die Veränderung so vieler Details wie nötig.[65]

HAUN definiert einen Frame als eine strukturierte Repräsentation einer Entität oder einer Klasse von Entitäten im Sinne einer Verallgemeinerung semantischer Netzwerkmodelle.[66] Frames tragen dem Bedürfnis wissensbasierter Systeme nach einer strukturierten und einheitlich kodierten Wissensbasis Rechnung um die Maschinenverarbeitbarkeit zu gewährleisten. Als Objekte enthalten Frames eine definierte Anzahl, mit speziellen Instanzen oder Daten gefüllte, Slots. Slots können auch mit Default-Werten und Verweisen auf andere Frames gefüllt werden. Ein Objekt in diesem Sinne ist eine vom Benutzer definierte Datenstruktur einschließlich der Operationen, die auf diese Daten ausgeführt werden.[67] Frames sind somit sehr ähnlich den Objekten aus der objektorientierten Programmierung (OOP).

Die Abbildung 9 zeigt den Frame eines Dreiecks (entnommen HAUN[68]).

Da Slots die charakteristischen Eigenschaften der modellierten Objekte beschreiben, unterscheiden sich Frames von Semantischen Netzen, da bei letzteren die Knoten nicht weiter unterteilt werden. Des weiteren sind Frames auch in Netzen organisiert, sie weisen hierbei jedoch eine Hierarchie auf, die bei Semantischen Netzen nicht vorkommt.

65) Vgl. MINSKY (1975).

66) Vgl. HAUN (2000) S. 103.

67) Vgl. SILBERBUSCH (1990) S. 172.

68) Vgl. HAUN (2000) S. 104.

Frame	Dreieck
Unterframe_von	Polygon
Anzahl_eckpunkte	drei
Anzahl_kanten	drei
Umfang	berechne
Farbe	unbekannt

Abbildung 9: Frame eines Dreiecks[48]

Programmtechnisch stellen Frames komplexe Datenstrukturen dar und können mit den OAW-Tripeln und den benutzerdefinierten Strukturen in prozeduralen Programmiersprachen verglichen werden (siehe hierzu Kapitel ?? und Kapitel ??).[69]

Ausgehend vom Frame-Konzept nach MINSKY entstandene Frame-Repräsentationssprachen (Auszug):

- FRL (Frame Representation Language, 1977)
- LISP
- KRL
- KL-ONE

69) Vgl. HAUN (2000), S. 103.

4.2. Repräsentation von überwiegend prozeduralem Wissen

Für die Abbildung von Abläufen und damit für die **Repräsentation von prozeduralem Wissen** kann man grundsätzlich zwischen aktivitätsorientierten, ereignisorientierten und objektorientierten Methoden unterscheiden.[70]

Aktivitätsorientierte Methoden bilden die Aktivitäten eines Prozesses, deren Verzweigungen und zeitliche Anordnung ab, können aber keine Ereignisse zwischen den einzelnen Aktivitäten darstellen. Beispielhaft für aktivitätsorientierte Methoden sind Programmablaufpläne.

Ereignisorientierte Methoden stellen mit Hilfe von Ereignissen, Zuständen und Aktivitäten den konditionalen Prozessablauf dar. Bei den ereignisorientierten Methoden sind vor allem Ereignisgesteuerte Prozessketten (EPK) und Petri-Netze von herausragender Bedeutung.

Objektorientierte Methoden stellen Objekte in den Mittelpunkt der Modellierung. Die Darstellung des Prozessablaufs erfolgt mit Hilfe von Nachrichten bzw. Transaktionen, die zwischen den Objekten ausgetauscht werden, wie beispielsweise im Semantischen Objektmodell (SOM) angewendet.

4.3. Repräsentation von Mischformen aus deklarativem/prozeduralem Wissen

Da mittlerweile in den Partnerunternehmen eine Prozesssicht hinsichtlich der betrieblichen Leistungserstellung überwiegt, soll sich im weiteren Verlauf bei den Mischformen aus deklarativem und prozeduralem Wissen auf solche Sprachen konzentriert werden, die insbesondere eine Abbildung von Prozessen unterstützen. Es werden das Entity-Relationship-Modell, die ereignisgesteuerten Prozessketten, Petri-Netze und das semantische Objektmodell untersucht.

4.3.1. ERM

Das auf CHEN[71] zurückgehende Entity-Relationship-Modell (ERM) unterscheidet hauptsächlich in Entities und Relationships. Als Entity wird ein „Ding“ verstanden,

---

70) Die Überschrift verdeutlicht bereits die Meinung des Verfassers, dass es die eigentliche *reine* Darstellung von prozeduralem Wissen (ohne deklarative Inhalte) nicht gibt. Deshalb wird an dieser Stelle weitergegangen zu den Mischformen, die die soeben genannten Aussagen aufgreifen. Es werden hierbei jedoch nur die ereignisorientierten Methoden und die objektorientierten Methoden betrachtet, da die Erfahrungen mit den Partnerunternehmen bereits frühzeitig verdeutlichten, dass aktivitätsorientierte Methoden (Programmablaufpläne) keine Rolle spielen würden.

71) Vgl. CHEN (1976)

dass sich deutlich identifizieren lässt, z.B. eine spezifische Person, Unternehmen oder Ereignis. Ein Relationship charakterisiert hingegen die Verbindung zwischen Entities (zum Beispiel kann „Vater-Sohn“ ein Relationship zwischen zwei „Person“-Entities darstellen).[72] Gleichartige Entities werden zu Klassen bzw. Typen zusammengefasst und gleichartige Relationships zu Relationshiptypen. Durch Attribute werden Entities und deren Eigenschaftswerte beschrieben. Durch ein oder mehrere Schlüsselattribute werden Entities eindeutig identifizierbar gemacht. Kanten verbinden die unterschiedlichen Informationsobjekte des ER-Ansatzes, zur Kennzeichnung der Relation zwischen den Entitytypen tragen die Kanten Kardinalitäten.[73] Grundsätzlich unterscheidet CHEN in 1:1, 1:n und m:n Beziehungen. Die Abbildung 10 zeigt die Darstellung des ER-Modells als ein einfaches Diagramm in Anlehnung an CHEN.[74]

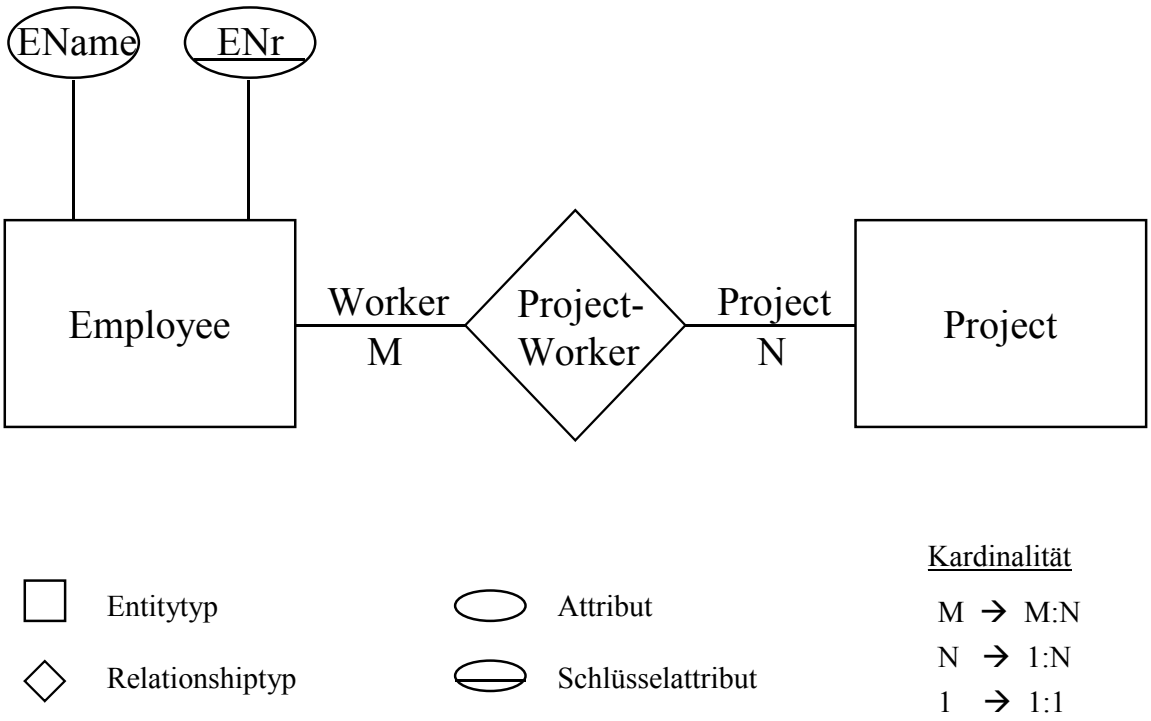

Abbildung 10: Einfaches Entity-Relationship-Diagramm in Anlehnung an Chen 1976, S. 19

72) Vgl. CHEN (1976)

73) Vgl. SCHÜTTE (1998), S. 94.

74) Vgl. Chen (1976), S.19

Es lässt sich nicht immer jeder Sachverhalt eindeutig darstellen und es ist durchaus denkbar, dass ein Relationship auf einer anderen Aggregationsstufe[75] beispielsweise zu einem Entity wird. Festzuhalten bleibt die starke Abhängigkeit der Modellentwicklung vom Auge des Betrachters.

4.3.2. Ereignisgesteuerte Prozessketten (EPKn)

Für ereignisgesteuerte Prozessketten (EPKn) ist die Gliederung eines Prozesses in Ereignisse und Funktionen typisch. Ereignisse lösen Funktionen aus und durchgeführte Funktionen erzeugen wiederum Ereignisse. Jeder Prozess beinhaltet mindestens ein Startereignis und ein Endereignis. Funktionen werden als aktive Informationsobjekte, die mit einem Substantiv (für das Objekt) und ein die Aktivität charakterisierendes Verb benannt werden, aufgefasst und Ereignisse demgegenüber als passive Informationsobjekte, die mit einem Substantiv (für das Objekt) und einem Verb zur Formulierung des Zustands benannt werden. Es können nur unterschiedliche Informationsobjekte miteinander verbunden werden. Neben direkten Verbindungen kommen auch Konnektoren zum Einsatz. Hauptsächlich lassen sich drei Arten von Konnektoren unterscheiden. Diese können auch kombiniert angewendet werden. Es werden Konnektoren verwendet, die „konjunkt“, „adjunkt“ und „disjunkt“ wirken, das heißt sie wirken als „und“, „inklusives und“ und als „exklusives und“. Verschiedene Prozesse können mit Prozesswegweisern verbunden werden. Zwangsläufig entsprechen dann die Endereignisse des vorlaufenden Prozesses den Anfangsereignissen des nachfolgenden Prozesses, dies entspricht einer vertikalen Dekomposition (Hierarchie). Des weiteren gibt es die horizontale Segmentierung, die, miteinander über Prozessschnittstellen verbundene, einzelne Prozessmodelle nebeneinander abbildet. Die Abbildung 11 zeigt beispielhaft die Ablaufbeschreibung einer Thematik des Handels.

75) zur Problematik der Aggregation siehe auch SCHÜTTE (1998), S.98.

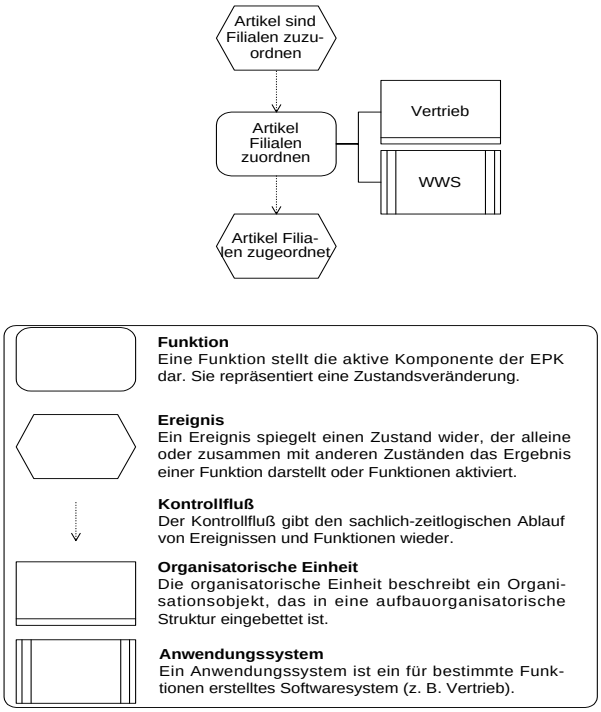

Abbildung 11: Ablaufbeschreibung mit der EPK[76]

76) SCHÜTTE (2001), S. 2.

4.3.3. Petri-Netze

Petri-Netze (PN) sind benannt nach Carl Adam Petri, der sie 1962 in seiner Dissertation als Modelle für nebenläufige (es besteht die Möglichkeit des parallelen Verlaufs) und nicht-deterministische Prozesse eingeführt hat. In der Wissenschaft sind verschiedene Varianten von PN bekannt wie Stelle/Transition-Netze, farbige PN, hierarchische PN, Bedingungs/Ereignisnetze, stochastische und zeitbehaftete PN, (erweiterte) Prädikate/Transitions-Netze und Kanal/Instanzen-Netze.

PN eignen sich insbesondere zur Repräsentation von Wissen, bei dem diskrete Ereignisse auftreten, die von diskreten Objekten verbraucht oder erzeugt werden. Ein abgebildetes System wird insbesondere durch seine passiven Elemente (Stellen) und seine aktiven Elemente (Transitionen), die in einem kausal-logischem Zusammenhang stehen, gekennzeichnet.

Strukturell ist ein Petrinetz ein geordnetes 3-Tupel PN=(S;T;F) für das gilt:[77]

- Die Stellenmenge $S=\{s_m:m=1,...,M\}$ mit $M\in N_0$ ist eine endliche Menge aus atomaren formalen Objekten s_m der Objektart "Stelle".
- Die Transitionenmenge $T=\{t_n:n=1,...,N\}$ mit $N\in N_0$ ist eine endliche Menge aus atomaren formalen Objekten t_n der Objektart "Transition".
- Die Flussrelation $F\subseteq((S\times T)\cup(T\times S))$ ist eine endliche Menge von zusammengesetzten formalen Objekten, die jeweils Paare (kn_x,kn_y) aus artverschiedenen atomaren formalen Objekten darstellen.
- Disjunktheitsbedingung IB_D: $S\cap T=\varnothing$.
- Existenzbedingung IB_E: $S\cup T\neq\varnothing$ [i)].
- Verknüpftheitsbedingung IB_V: $S\cup T=VB(F)\cup NB(F)$.

Die Klasse der Stelle/Transition-Netze stellt eine Art Standard[78] und wird deshalb an dieser Stelle näher erläutert um die Repräsentationsmethodik zu verdeutlichen.

Ein Stelle/Transition-Netz ist ein geordnetes 6-Tupel $STN=(S,T;F,K,W,M_0)$, für das gilt:

---

77) Vgl. ZELEWSKI, (1995a) S. 25. Siehe hier auch zu näheren Erläuterungen.

78) Vgl. MÜLLER-CLOSTERMANN (2001) S. 2-4.

- Die Stellenmenge $S=\{s_m:m=1,...,M\}$ ist eine nicht-leere, endliche Menge aus atomaren formalen Objekten s_m der Objektart "Stelle" mit $M \in N_+$.
- Die Transitionenmenge $T=\{t_n:n=1,...,N\}$ ist eine nicht-leere, endliche Menge aus atomaren formalen Objekten t_n der Objektart "Transition" mit $N \in N_+$.
- Die Flußrelation $F \subseteq ((S \times T) \cup (T \times S))$ ist eine nicht-leere, endliche Menge von zusammengesetzten formalen Objekten, die jeweils Paare (kn_x,kn_y) aus artverschiedenen atomaren formalen Objekten darstellen.
- Die Kapazitätsfunktion $K: S \rightarrow N_+ \cup \{\omega\}$[79] ordnet jeder Stelle s_m eine Markenkapazität $K(s_m)$ zu.
- Gewichtsfunktion $W: F \rightarrow N_0$ bildet jedes Element (kn_x,kn_y) aus der Flussrelation auf das Kantengewicht $W(kn_x,kn_y)$ ab.

$$W(kn_x,kn_y) = \begin{cases} W^*(kn_x,kn_y); & \text{für } (kn_x,kn_y) \in F \\ 0; & \text{für } (kn_x,kn_y) \notin F \end{cases}$$

- Die Markierungsfunktion $M_0: S \rightarrow N_0$ schreibt jeder Stelle s_m eine Markierung $M_0(s_m)$ zu.
- Disjunktheitsbedingung IB_D: $S \cap T = \varnothing$.
- Existenzbedingung IB_E: $S \cup T \neq \varnothing$.
- Verknüpftheitsbedingung IB_V: $S \cup T = VB(F) \cup NB(F)$.

Die STN lassen sich sowohl als Matrizen als auch als Graphen darstellen. Aus Anschauungsgründen soll hier kurz auf die graphische Darstellung der Repräsentation von Wissen anhand eines Beispiels eingegangen werden. Die Stellen/Transition-Netze lassen sich – in der hier gebotenen Kürze – grob als gerichtete, beschriftete, bipartite Graphen charakterisieren.[80]

STN setzen sich Grundelementen zusammen:

Stellen (Zustände, Bedingungen), Transitionen (Übergänge, Ereignisse), Inputrelationen, Outputrelationen und Markierungen (Marken, Tokens), die den verschiedenen Zuständen zugeordnet werden. Stellen werden dabei durch Kreise und Transitionen mittels Rechtecken dargestellt. Die Input- und Outputrelationen ergeben sich durch

79) Mit „ω“ wird eine beliebige, unbeschränkt große Zahl bezeichnet.

80) ZELEWSKI, S.: (1996) S. 369-381.

gerichtete Kanten zwischen Input-Zustand und Übergang bzw. Übergang und Output-Zustand. In Abbildung 12 bedeuten die Zahlen an den Kanten die jeweiligen Input- und Outputrelation einer Schaltung. Die Stellen S_{1-3} sind die Objekte die benötigt werden um eine Transition durchzuführen. Die Abbildung[81] zeigt zwei aufeinanderfolgende Zustände M_r (Referenzmarkierung) und M_f (Folgemarkierung). Es ist zu erkennen, dass die, durch die Inputrelationenangabe geforderten Marken entfernt werden und die durch die Outputrelationenangabe geforderten Marken hinzukommen. Zur Veranschaulichung[82] kann man sich den Prozess in der Abbildung als die Montage zweier Baugruppen vorstellen, die die Komponenten S_{1-3} in der geforderten Stückzahl {1;3;1} als Bauteile benötigt und nach erfolgreicher Montage die Baugruppen S_{4-5} in den angegebenen Stückzahl vorhält (die Baugruppe S_5 liegt hierbei aus unerklärlichen Gründen bereits einmal auf Halde). M_r stellt dabei den „Vorher-Zustand“ dar und M_f den „Nachher-Zustand“.

81) ZELEWSKI (1995c) S. 46.

82) Zur weiteren Veranschaulichung vgl. Baumgarten (1990), Rosenstengel (1991), zur Beurteilung Zelewski (1995b).

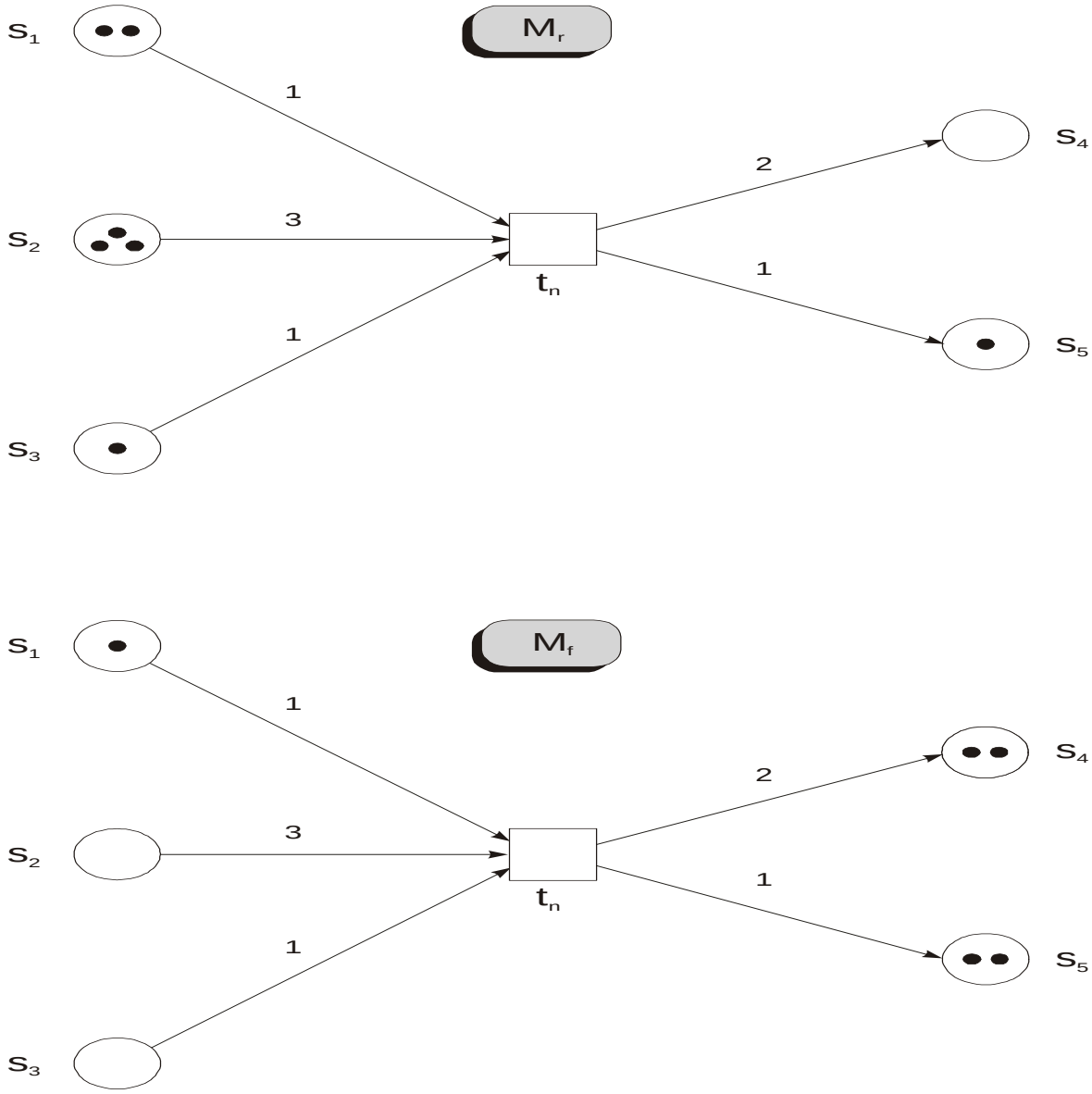

Abbildung 12: Wirkung des Schaltens einer Transition[42]

Stellen, Gewichte und Kapazitäten entsprechen hier deklarativem Wissen hingegen Transitionen und die spezifische Anordnung (Flussrelation) aller Elemente prozeduralem Wissen.

4.3.4. Semantisches Objektmodell (SOM)

Gegenstand der objektorientierten Methoden sind allgemein Objekte, die Nachrichten austauschen. Die Objekte werden vollständig beschrieben durch die Definition ihrer Daten und Methoden (Objektschema), wobei Methoden bei Eingang einer Nachricht ausgeführt werden können oder müssen und dabei auf die objekteigenen Daten zugreifen. Einzelne Objekte werden anschließend zu Objektklassen zusammengefasst. Der Prozessablauf wird durch eine bestimmte Reihenfolge des Nachrichtenaustauschs zwischen den modellierten Objekten und der jeweiligen Ausführung der objekteigenen Methoden dargestellt (Interaktionsdiagramme).[83]

Das Semantische Objektmodell (SOM) wurde 1990 erstmals von FERSTL und SINZ vorgestellt. Es unterstützt die Modellierung von Geschäftsprozessen in Form von betrieblichen Transaktionen und Objekten als Ausgangspunkt für die weitere Analyse und Gestaltung des betrieblichen Informationssystems[84]. Dabei sind insbesondere eine durchgängige Orientierung der Transaktionen und Objekte an der Zielerfüllung der Unternehmung und ihre hierarchische Zerlegung sowie eine explizite Modellierung der Koordination von Objekten bei Durchführung von Transaktionen wichtig.

Das SOM beinhaltet einen Modellierungsansatz, eine Unternehmensarchitektur und ein Vorgehensmodell. Repräsentiert wird das gesamte betriebliche Objektsystem, wobei zwei Sichten zur Abgrenzung dienen. Nach der Außensicht wird der Modellierungsgegenstand als offenes, zielgerichtetes sozio-technisches System behandelt, nach der Innensicht als ein verteiltes System lose gekoppelter Objekte, die bei gemeinsamen Zielen kooperieren. Das betrieblich Objekt wird als eine Menge von Aufgaben, die gemeinsame Sach- und Formalziele verfolgen, betrachtet, die ihre Transaktionen koppeln.

Die besonderen Merkmale des ganzheitliche Modellierungsansatzes im SOM sind nach FERSTL/SINZ[85]: *Modellierungsreichweite*, *Modellumfang/-integration* und die *formalen Modelleigenschaften.* Die Modellierung eines betrieblichen Systems[86], als ein System interagierender Geschäftsprozesse, wird sowohl aus der Perspektive der Leistungserstellung als auch aus der ihrer Lenkung vorgenommen. Ergänzend wer-

---

83) Vgl. LANG (1997) S.19.

84) Vgl. FERSTLSINZ (1993)

85) Vgl. FERSTLSINZ (1994) S.4f.

86) FERSTL/SINZ verstehen unter einem betrieblichem System einen Oberbegriff für Unternehmen, Unternehmensverbünde oder Geschäftsbereiche von Unternehmen (Vgl. FERSTLSINZ (1994), S.3).

den zur Abbildung der Ressourcen die Anwendungssysteme zum Modellierungsobjekt. Der SOM-Ansatz unterscheidet, bezüglich des *Modellumfangs*, die Teilmodelle Unternehmensplan, Geschäftsprozessmodelle und Anwendungsspezifikationen. Der Unternehmensplan berücksichtigt hierbei die Außenansicht, die Geschäftsprozessmodelle die Innenansicht eines Systems. Ein Ressourcenmodell wird von den Anwendungsspezifikationen abgebildet. Die *Modellintegration* wird durch ein integriertes Meta-Modell hergestellt, das für Geschäftsprozesse und Anwendungssysteme ein einheitliches Modellierungskonzept bereithält und somit die Beziehungen zwischen den Teilmodellen und den möglichen Modellsichten, die alle auf einer semi-formalen und vorwiegend diagrammgestützten Darstellungsweise beruhen, festlegt. Die weiteren formalen Modelleigenschaften werden durch die konsequente Ausrichtung auf eine objektorientierte Sichtweise determiniert. Die exemplarische Repräsentation von Wissen wird in Abbildung 13 dargestellt.

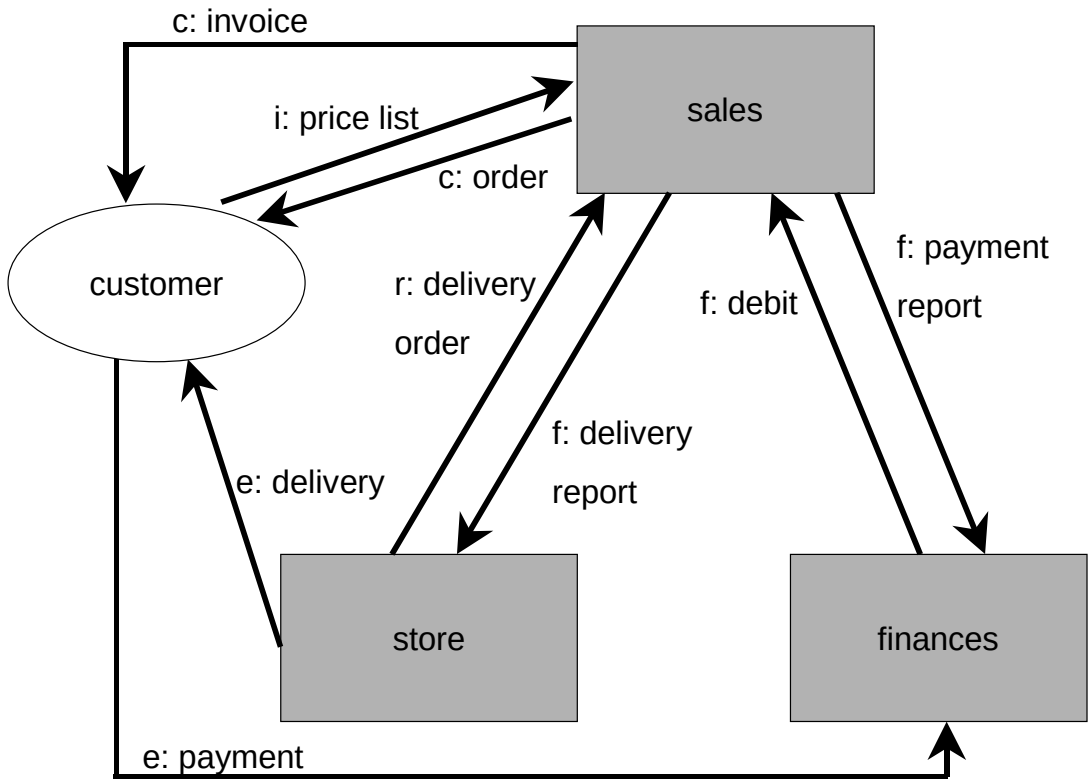

Abbildung 13: Interaction schema of business process distribution (4th level)[87]

87) Abbildung entnommen aus FERSTLSINZ (1998), S. 356.

Sie zeigt exemplarisch die finale Dekomposition des Geschäftsprozesses *Distribution* im interaction schema, das zusammen mit dem task-event schema die 4. Ebene darstellt (für weitere Informationen siehe FERSTL/SINZ).[83]

4.4. Kritische Betrachtung der Repräsentationen

Bei den ereignisgesteuerte Prozessketten erreicht man lediglich eine geringe Modellkomplexität, durch Abbildung nur eines Modelltyps, vorteilhaft hingegen ist die intuitiv leichte Verständlichkeit der zeitlich-logischen Prozessdarstellung.

Bei den Petri-Netzen stößt man hingegen schnell auf Akzeptanzbarrieren, weil kompliziertere Sachverhalte eine ungleich größere Verkomplizierung der Darstellung nach sich ziehen.

Die EPK haben gerade auch deshalb weite Verbreitung bei der Modellierung von Geschäftsprozessen gefunden, obwohl sie als semi-formales Modell über eine unscharfe Syntax und Semantik verfügen. Außerdem existiert eine Standard-Lösung (ARIS).

Gegenüber den objektorientierten Sichten entspricht die ereignisorientierte Sicht der natürlichen Modellbildung eher, die objektorientierte Sicht weist jedoch konzeptionelle Vorteile auf, z.B. hinsichtlich Modellierungsaufwand, Konsistenzsicherung und Wiederverwendbarkeit.

5. Repräsentationssprachen in speziellen Verwendungszusammenhängen

Aufgrund der ausgeprägten Prozesssicht bei den beteiligten Projektpartnern und den eigenen Kenntnissen über Prozessmodellierung und die verbesserte Nachvollziehbarkeit durch Dritte wird im weiteren Verlauf des Projekts KOWIEN auf die vorliegenden Geschäftsprozesse der Fokus der Tätigkeiten gelegt. An späterer Stelle des Projektablaufs werden Prozesse hinsichtlich ihrer Wissensquellen und –ziele bezüglich der Füllung von Kompetenzprofilen untersucht, deshalb wird im folgenden Abschnitt noch einmal in besonderer Form auf Geschäftsprozesse und die damit verbundenen Modelle eingegangen, nicht zuletzt auch um eine eigene einheitliche Sprache diesbezüglich zu installieren.

Einer ersten Systematisierung von (Geschäftsprozess-)modellen folgen die Anforderungen an Referenzmodelle aus betriebswirtschaftlicher Sicht. Anschließend wird die Anwendbarkeit auf das Projekt KOWIEN untersucht.

5.1. Systematisierung von (Geschäftsprozess-)Modellen

Aufgrund ihrer generalisierten Reichweite[88] intendierter Anwendungen bei gleichzeitigem normativem Repräsentationsanspruch, wird sich bei der weiteren Arbeit auf Referenzmodelle konzentriert.

5.2. Referenzmodelle aus betriebswirtschaftlicher Sicht

Im betrieblichen Umfeld kann man Vorgehensmodelle, Branchenreferenzmodelle und Softwarereferenzmodelle beobachten:[89]

Vorgehensmodelle beschreiben Projektabläufe wie beispielsweise die Durchführung eines Projekts zum Business Process Reengineering, die Einführung einer Standardsoftware oder die Durchführung einer ISO-Zertifizierung.

Branchenmodelle dokumentieren sowohl Prozesse als auch Daten- und Funktionsstrukturen, die typisch für eine bestimmte Wirtschaftsbranche sind. Branchenmodelle beziehen sich z.B. auf operative Geschäftsprozesse wie Logistik, Produktentwicklung und Rechnungswesen. In ARIS (Architektur Integrierter Informationssysteme), das die methodische Grundlage für die Branchenreferenzmodelle von Scheer dar-

88) Auch unter Berücksichtigung der Verpflichtung aus dem Projektrahmenplan zur Erstellung eines generischen Vorgehensmodells.

89) Vgl. LANG (1997) S. 21ff.

stellt[90], stehen z.B. Modelle für die Branchen Papierindustrie, Chemische Industrie, Maschinenbau, Anlagenbau und Energie zur Verfügung.

Softwaremodelle bilden die typischen Prozesse, Funktions- und Datenstrukturen von Standardsoftwaresystemen ab. Softwarespezifische Referenzmodelle können sehr gut bei der Auswahl und Einführung einer Software zum Modellvergleich zwischen Anforderungen und Angebot der Software sowie zum Customizing von Standardsoftware eingesetzt werden.[91]

6. Anforderungen an Repräsentationssprachen aus Sicht von KOWIEN

Die Anforderungen präsentieren sich in einer ersten überblicksartigen Struktur folgendermaßen:

- *Möglichkeit der Abbildung von Ontologien*, aus dem Projektvorhaben ergibt sich die Notwendigkeit Ontologien erstellen können zu müssen, das bedeutet mindestens eine Sprache muss, bei der Erstellung des wissensbasierten prototypischen Systems die Repräsentation von Ontologien und ihre maschinelle Verarbeitbarkeit ermögliche.
- *Computerverarbeitbarkeit,* sämtliche zu verwendende Sprachen müssen mit dem Computer (besser: formal) verarbeitbar sein um neues Wissen maschinell generieren zu können.
- *Informationsgehalt intersubjektiv nachprüfbar,* bedeutet, dass das repräsentierte Wissen innerhalb einer Sprache auch Dritten gegenüber kommunizierbar und kontrollierbar sein muss. Insbesondere der sensible Bereich der sozialen und Selbst-Kompetenzen wird beispielsweise vom Datenschutzbeauftragten strengstens verfolgt und kritisiert werden.
- *Rückschlüsse auf Kompetenz,* nicht zuletzt muss das innerhalb einer Sprache repräsentierte Wissen Rückschlüsse auf Kompetenzen und die Bildung von dazugehörigen Profilen erlauben.

90) Vgl. REITER (1997) S. 34.

91) Vgl. SCHEER (1997) S. 9.

7. Praktische Empfehlungen an Partnerunternehmen

Die verwendeten Repräsentationssprachen der Partner werden in dem Arbeitsbericht zu den Zwecken und Sprachen des Wissensmanagements dargelegt und eingehender hinsichtlich ihrer Unterstützungsfunktion bei der ontologiegestützten Kompetenzprofilerstellung untersucht. Es zeigt sich, dass für das Projekt KOWIEN F-Logic, RDF(S) bzw. XML und Ontologien die geeigneten Werkzeuge für eine Repräsentation des Wissens über Kompetenzprofile sind. Die anderen genannten Sprachen sollten in einer Checkliste bereitgehalten werden um sie bei den Partner unternehmen identifizieren zu können und somit vorhandenes (also bereits repräsentiertes) Wissen nutzen zu können. Derzeit wird mit den Projektpartnern eine allgemeingültige Geschäftsprozesse-Darstellung erarbeitet, innerhalb der besondere Stellen verortet werden, an denen sich Wissen bildet oder sammelt oder aber Wissen gebraucht oder verloren geht. Hierzu erscheint es notwendig die genannten Sprachen und ihre Rolle näher zu untersuchen und für das zu entwickelnde wissensbasierte System nutzbar zu machen.

8. Literaturverzeichnis

[LOOM GUIDE 91] Loom User's Guide for Loom version 1.4. ISX Corporation, August 1991.
URL: http://www.isi.edu/isd/LOOM/documentation/usersguide1.4.ps,
Zugriff am 12.5.2002.

BAUMGARTEN, BERND: Petri-Netze: Grundlagen und Anwendungen, Mannheim 1990.

BECKERMANN, ANSGAR: Einführung in die Logik. Berlin – New York 1997.

BIBEL, WOLFGANG: Wissensrepräsentation und Inferenz, Wiesbaden 1993.

BUCHER, T.G.: Einführung in die angewandte Logik. Berlin – New York 1987.

CHEN, PETER P.: ACM Transactions on Database Systems, Vol. 1, No.1, März 1976.

CORCHO ET AL.: OntoWeb Technical Roadmap 1.0.
URL: http://www.ontoweb.org/download/deliverables/D11_v1_0.pdf,
Zugriff am 9.4.2002.

CZAYKA, L.: Formale Logik und Wissenschaftsphilosophie. 2. Aufl., München – Wien 2000.

DECKER, S.: On Domain-Specific Declarative Knowledge Representation and Database Languages. In: Borgida, A.; Chaudri, V.; Staudt M. [Hrsg.]: Proceedings of the 5th Knowledge Representation meets Databases Workshop (KRDB98), Seattle 1998.

FENSEL, DIETER; VAN HARMELEN, FRANK: A Comparison of Languages which Operationalize and Formalise KADS Models of Expertise. In: The Knowledge Engineering Review, Vol. 9, 1994.

FERSTL, O.; SINZ, E.: SOM Modeling of Business Systems. In: Bernus, Mertins, Schmidt (Hrsg.): Handbook on Architectures of Information Systems, Berlin 1998, S. 339-358.

FERSTL, O.; SINZ, E.: Der Ansatz des semantischen Objektmodells zur Modellierung von Geschäftsprozessen. In: Bamberger Beiträge 1994 , Nr. 23.

FERSTL, O.; SINZ, E.: Geschäftsprozessmodellierung. In: Wirtschaftsinformatik, 35 (1993) 6, S. 589-592.

GÖRTZ, GÜNTHER: Wissensrepräsentation, 1991.

HAUN, M.: Wissensbasierte Systeme, Renningen 2000.

HEINSOHN, J.; SOCHER-AMBROSIUS, R.: Wissensverarbeitung. Heidelberg – Berlin 1999.

HENNINGS, R.D.: Informations- und Wissensverarbeitung – Theoretische Grundlagen Wissensbasierter Systeme Berlin 1991.

KIFER, M; LAUSEN, G.; WU, J.: Logical Foundations of Object-Oriented and Frame-Based Languages. In: Journal of the ACM. o.Jg. (1995).

KLAUS, GEORG; MANFRED BUHR (Hrsg.): Philosophisches Wörterbuch. Berlin 1971.

LANG, KLAUS: Gestaltung von Geschäftsprozessen mit RPB, Wiesbaden 1997.

MATES, B.: Elementare Logik. 2. Aufl., Göttingen 1978.

MINSKY, M.A.: A Framework for Representing Knowledge, in: Winston, P.H. (Hrsg.): The Psychology of Computer Vision, New York 1975.
URL: http://web.media.mit.edu/~minsky/papers/Frames/frames.html,
Zugriff am 6.4.2002.

MÜLLER, PETER: Spezifikation und Implementierung einer Annotationssprache für eine objektorientierte Programmiersprache, TU München 1995.
URL: http://www.informatik.fernuni-hagen.de/import/pi5/veroeffentlichung/salsa/salsa.pdf ,
Zugriff am 3.4.2002.

MÜLLER-CLOSTERMANN, B.: Modelle der Informatik, Kap. 2: Petri-Netze, S. 2-4, Skript WS 01/02. URL: http://www.informatik.uni-essen.de/SysMod/lehre/ModelleInfDaten/SkriptWS2000/Kap2PetriNetze2000.pdf,
Zugriff am 10.4.2002.

o.V.: How to write F-Logic Programs – A Tutorial for the Language F.Logic.
URL: http://www.ontoprise.de/download/f_logik_tutorial.zip ,
Zugriff am 10.4.2002.

OSTERMAYER, RAINER: Pragmatisch-situative Wissensrepräsentation – ein Baustein für das Wissensmanagement, Aachen 2001.

PARTSCH ET AL.: Requirements Engineering, München 1991.

POCSAI, ZSOLT: Ontologiebasiertes Wissensmanagement in der Produktentwicklung, Aachen 2000.

PUPPE, FRANK [Hrsg.]: Expertensysteme 93 / 2. Deutsche Tagung Expertensysteme (XPS-93), Hamburg, 17. - 19. Februar 1993.

RAUTENBERG, W.: Einführung in die mathematische Logik. Braunschweig – Wiesbaden 1996.

REITER, CHRISTIAN: Toolbasierte Referenzmodellierung in: Entwicklungsstand und Entwicklungsperspektiven der Referenzmodellierung, Münster 1997.
URL: http://www.wi.uni-muenster.de/inst/arbber/ab52.pdf,
Zugriff am 14.2.2002.

ROSENSTENGEL, B.; WINAND, U.: Petri-Netze – Eine anwendungsorientierte Einführung, 4. Aufl., Braunschweig 1991.

ROSS, D.T.: Structured Analysis (SA): A Language for Communicating Ideas. In: IEEE Transactions on Software Engineering, Vol. SE-3, No. 1, 1977, S. 16-34.

RUPPEN, P.: Einstieg in die formale Logik, Bern 1997.

RYLE, G.: The Concept of Mind, London 1949.

SCHEER, AUGUST-WILHELM: Toolbasierte Referenzmodellierung in: Entwicklungsstand und Entwicklungsperspektiven der Referenzmodellierung, Münster 1997, http://www.wi.unimuenster.de/inst/arbber/ab52.pdf, Zugriff am 14.2.2002.

SCHREIBER ET AL.: Knowledge engineering and management: the CommonKADS methodology, Massachusetts 2001, 2. Aufl..

SCHÜTTE, REINHARD.: Informations(fluss)modelle, unveröffentlichtes Manuskript 2001.

SCHÜTTE, REINHARD: Grundsätze ordnungsgemäßer Referenzmodellierung: Konstruktion konfigurations- und anpassungsorientierter Modelle, Wiesbaden 1998.

SCHWEIZER, H.: Bedeutung – Grundzüge einer internalistischen Semantik. Bern - Stuttgart – Haupt 1991.

SILBERBUSCH, PETER: Methoden der Wissensrepräsentation, in: BEHRENDT, R. (HRSG.): Angewandte Wissensverarbeitung, München 1990, S. 167.

WACHSMUTH, I.: Methoden der Künstlichen Intelligenz. Skriptum WS 2000/2001 Bielefeld.

ZELEWSKI, S.: Eignung von Petrinetzen für die Modellierung komplexer Realsysteme – Beurteilungskriterien, Wirtschaftsinformatik 38 (1996) 4.

ZELEWSKI, S.: Petrinetzbasierte Modellierung komplexer Produktionssysteme, Band 3, Arbeitsbericht Nr.7 (1995a) Universität Leipzig.

ZELEWSKI, S.: Petrinetzbasierte Modellierung komplexer Produktionssysteme, Band 1, Arbeitsbericht Nr. 5 (1995b) Universität Leipzig.

ZELEWSKI, S.: Petrinetzbasierte Modellierung komplexer Produktionssysteme, Band 9, Arbeitsbericht Nr. 14 (1995c) Universität Leipzig.

9. Anhang

Der Anhang enthält eine Liste der gängigen Formate aus Internet/Intranet, sie sollen einen ersten Überblick als Hilfestellung bei der Erfassung der Dokumente der KOWIEN-Projektpartner geben, weil die IT-Umgebung als die zentrale Wissensquelle für das Projekt angesehen wird.

Die Abkürzung MIME steht für "**M**ultipurpose **I**nternet **M**ail **E**xtensions". MIME-Typen sind ein Internet-Standard, um Dateitypen anzugeben. Im Zusammenhang mit multimedialen Elementen auf WWW-Seiten werden diese Angaben in Zukunft immer wichtiger. MIME-Typen werden bei der Kommunikation zwischen WWW-Server und WWW-Browser eingesetzt. Sowohl der WWW-Server als auch der WWW-Browser unterhält eine Liste mit ihm bekannten Dateitypen. In vielen WWW-Browsern (z.B. bei Netscape) ist das die Liste der sogenannten "Helper Applications". Beim Übertragen vom Server zum Browser wird über das HTTP-Protokoll der MIME-Type mit übertragen. Aufgrund seiner Liste mit MIME-Typen weiß der WWW-Browser, wie er die Datei zu behandeln hat.[92]

Mime-Type	Dateinamenerweiterung(en)	Beschreibung
`application/acad` (NCSA)	`dwg`	AutoCAD-Dateien
`application/dxf` (CERN)	`dxf`	AutoCAD-Dateien
`application/mif`	`mif`	Maker Interchange Format (Adobe FrameMaker)
`application/msword`	`doc dot`	MS-Word-Dateien
`application/mspowerpoint`	`ppt ppz pps pot`	MS-Powerpoint-Dateien
`application/msexcel`	`xls xla`	MS-Excel-Dateien
`application/mshelp`	`hlp chm`	MS-Windows-Hilfe-Dateien
`application/octet-stream`	`com exe bin dll class`	Ausführbare Dateien bzw. Programmcode-Dateien
`application/pdf`	`pdf`	PDF-Dateien (Adobe Acrobat Exchange/Reader)
`application/postscript`	`ai eps ps`	Postscript-Dateien (Adobe)

[92]) vgl. http://www.bbso.njit.edu/Documentations/htmldoc/tcjj.htm, 8.4.2002.

application/rtf	rtf	RTF-Dateien (Microsoft)
application/x-sh	sh	Bourne Shell Script (Unix)
application/x-csh	csh	C Shell Script (Unix)
application/x-latex	latex	LaTeX-Quelldatei (Unix)
application/x-mif	mif	Maker Interchange Format (Adobe FrameMaker Unix)
application/x-tar	tar	tar-Archivdatei (Unix)
application/x-cpio	bcpio	CPIO-Datei alt binär (Unix)
application/x-bcpio	cpio	CPIO-Datei (Posix)
application/x-sv4cpio	sv4cpio	CPIO-Datei (SVR4)
application/x-sv4crc	sv4crc	CPIO-Datei (SVR4 mit CRC)
application/x-sv4crc	sv4crc	CPIO-Datei (SVR4 mit CRC)
application/x-hdf	hdf	NCSA HDF Data File
application/x-ustar	ustar	tar-Archivdatei (Posix)
application/x-shar	shar	Shell-Archiv-Datei (Unix)
application/x-tcl	tcl	TCL-Script (Unix)
application/x-dvi	dvi	TeX dvi (Unix)
application/x-texinfo	texinfo texi	Emacs Texinfo (Unix)
application/x-troff	t tr roff	troff-Dateien (Unix)
application/x-troff-man	man	troff mit MAN-Makros (Unix)
application/x-troff-me	me	troff mit ME-Makros (Unix)
application/x-troff-ms	ms	troff mit MS-Makros (Unix)
application/x-netcdf	nc cdf	Unidata netCDF (Unix)
application/x-wais-source	src	WAIS-Quelldatei (Unix)
application/x-www-form-urlencoded		HTML-Formulardaten an CGI
audio/basic	au snd	AU- und SND-Sound-

		Dateien
audio/x-aiff	aif aiff aifc	AIFF-Sound-Dateien
audio/x-aiff	aif aiff aifc	AIFF-Sound-Dateien
audio/x-dspeeh	dus cht	Sprach-Dateien
audio/x-midi	midi mid	MIDI-Dateien
audio/x-pn-realaudio	ram ra	RealAudio-Dateien
audio/x-pn-realaudio-plugin	rpm	RealAudio-Plugin-Dateien
image/cmu-raster	ras	CMU-Raster
image/x-freehand	fh4 fh5 fhc	Freehand-Grafik
image/gif	gif	GIF-Grafik
image/ief	ief	Image Exchange Format
image/jpeg	jpeg jpg jpe	JPEG-Grafik
image/x-portable-anymap	pnm	PBM Anymap-Datei
image/x-portable-bitmap	pbm	PBM Bitmap-Datei
image/x-portable-graymap	pgm	PBM Graymap-Datei
image/x-portable-pixmap	ppm	PBM Pixmap-Datei
image/x-rgb	rgb	RBG-Grafik
image/x-windowdump	xwd	X-Windows Dump
image/tiff	tiff tif	TIFF-Grafik
text/css	css	CSS-Style-Sheet-Datei
text/html	html htm	HTML-Datei
text/javascript	js	JavaScript-Datei
text/plain	txt c cc g h hh m f90	reine Text-Datei
text/richtext	rtx	MIME Richtext
text/tab-separated-values	tsv	Datentextdatei mit Tabulatoren als
text/x-setext	etx	Struct.erw. Text
text/x-sgml	sgm sgml	SGML-Datei
video/mpeg	mpeg mpg mpe	MPEG Video
video/quicktime	qt mov	Quicktime-Video

video/x-msvideo	avi	Microsoft AVI-Video
video/x-sgi-movie	movie	Microsoft SGI-Video
video/x-sgi-movie	movie	Microsoft SGI-Video
x-world/x-vrml	wrl	VRML-Dateien

Institut für Produktion und Industrielles Informationsmanagement Universität Duisburg-Essen / Campus Essen

---

Verzeichnis der KOWIEN-Projektberichte

Nr. 1: ALPARSLAN, A.: Ablauforganisation des Wissensmanagements. Projektbericht 1/2002, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Essen, Essen 2002.

Nr. 2: ALAN, Y.: Methoden zur Akquisition von Wissen über Kompetenzen. Projektbericht 2/2002, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Essen, Essen 2002.

Nr. 3: DITTMANN, L.: Sprachen zur Repräsentation von Wissen - eine untersuchende Darstellung. Projektbericht 3/2002, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Essen, Essen 2002.

Nr. 4: DITTMANN, L.: Zwecke und Sprachen des Wissensmanagements zum Managen von Kompetenzen. Projektbericht 4/2002, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Essen, Essen 2002.

Nr. 5: ALAN, Y.; BÄUMGEN, C.: Anforderungen an den KOWIEN-Prototypen. Projektbericht 5/2002, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Essen, Essen 2002.

Nr. 6: ALPARSLAN, A.: Wissensanalyse und Wissensstrukturierung. Projektbericht 6/2002, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Essen, Essen 2002.

Nr. 7: ALAN, Y.: Evaluation der KOWIEN-Zwischenergebnisse. Projektbericht 7/2002, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Essen, Essen 2002.

Nr. 8: ZUG, S.; KLUMPP, M.; KROL, B.: Wissensmanagement im Gesundheitswesen, Arbeitsbericht Nr. 16, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2003.

Nr. 9: APKE, S.; DITTMANN, L.: Analyse von Vorgehensmodellen aus dem Software, Knowledge und Ontologies Engineering. Projektbericht 1/2003, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2003.

Nr. 10: ALAN, Y.: Konstruktion der KOWIEN-Ontologie. Projektbericht 2/2003, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2003.

Nr. 11: ALAN, Y.: Ontologiebasierte Wissensräume. Projektbericht 3/2003, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2003.

Nr. 12: APKE, S.; DITTMANN, L.: Generisches Vorgehensmodell KOWIEN Version 1.0. Projektbericht 4/2003, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2003.

Nr. 13: ALAN, Y.: Modifikation der KOWIEN-Ontologie. Projektbericht 5/2003, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2003.

Nr. 14: ALAN, Y.; ALPARSLAN, A.; DITTMANN, L.: Werkzeuge zur Sicherstellung der Adaptibilität des KOWIEN-Vorgehensmodells. Projektbericht 6/2003, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2003.

Nr. 15: ENGELMANN, K.; ALAN, Y.: KOWIEN Fallstudie - Gebert GmbH. Projektbericht 7/2003, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2003.

Nr. 16: DITTMANN, L.: Towards Ontology-based Skills Management. Projektbericht 8/2003, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2003.

Nr. 17: ALPARSLAN, A.: Evaluation des KOWIEN-Vorgehensmodells, Projektbericht 1/2004, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2004.

Nr. 18: APKE, S.; BÄUMGEN, C.; BREMER, A.; DITTMANN, L.: Anforderungsspezifikation für die Entwicklung einer Kompetenz-Ontologie für die Deutsche Montan Technologie GmbH. Projektbericht 2/2004, Projekt KOWIEN, Universität Duisburg-Essen (Campus Essen), Essen 2004.

Nr. 19: HÜGENS, T.: Inferenzregeln des „plausiblen Schließens“ zur Explizierung von implizitem Wissen über Kompetenzen. Projektbericht 3/2004, Projekt KOWIEN, Universität Duisburg-Essen (Campus Essen), Essen 2004.

Nr. 20: ALAN, Y.: Erweiterung von Ontologien um dynamische Aspekte. Projektbericht 4/2004, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2004.

Nr. 21: WEICHELT, T.: Entwicklung einer E-Learning-Anwendung zum kompetenzprofil- und ontologiebasierten Wissensmanagement – Modul 1: Grundlagen. Projektbericht 5/2004, Projekt KOWIEN, Universität Duisburg-Essen (Campus Essen), Essen 2004.