Institut für Produktion und Industrielles Informationsmanagement

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

KOWIEN-Projektbericht 3/2004 – V. 1.0

Inferenzregeln des „plausiblen Schließens“ zur Explizierung von implizitem Wissen über Kompetenzen

Dipl.-Wirt.-Inf. Torben Hügens
E-Mail: Torben.Huegens@pim.uni-essen.de

Stand: 21.04.2004

KOWIEN
KOOPERATIVES WISSENSMANAGEMENT IN ENGINEERING-NETZWERKEN

Das Drittmittelprojekt KOWIEN
(„Kooperatives Wissensmanagement in Engineering-Netzwerken”)
wird mit Mitteln des
Bundesministeriums für Bildung und Forschung (BMBF) gefördert
(Förderkennzeichen Hauptband 02 PD 1060).
Die Mitglieder des Projektteams danken
für die großzügige Unterstützung ihrer Forschungs- und Transferarbeiten.

April 2004

Inhaltsverzeichnis

Abkürzungs- und Akronymverzeichnis

AIFB	Angewandte Informatik und Formale Beschreibungsverfahren
Aufl.	Auflage
d.h.	das heißt
DAML	DARPA Agent Markup Language
DARPA	Defense Advanced Research Projects Agency
DTD	Document Type Definition
F-Logic	Frame Logic
Hrsg.	Herausgeber
KI	Künstliche Intelligenz
KIF	Knowledge Interchange Format
No.	Number
NS	Namespace
o.S.	ohne Seitenangabe
o.V.	ohne Verfasser
OIL	Ontology Interchange Language
OKBC	Open Knowledge Base Connectivity
OWL	Web Ontology Language
OXML	OntoEdit's XML-based Ontology Representation Language
RDF	Resource Description Framework
RDF(S)	Resource Description Framework
RDQL	RDF Data Query Language
RQL	RDF Query Language
S.	Seite
sog.	so genannt(e)
UNIX	Open-Source-Betriebssystem
URI	uniform ressource identifier (übersetzt: eindeutige Adresse)
Vgl. / vgl.	vergleiche
W3C	World Wide Web Consortium
XML	Extensible Markup Language
XOL	XML-based Ontology Exchange Language
z.B.	zum Beispiel

Symbolverzeichnis

->	Einwertige funktionale Methode auf Instanzebene Ergebnis ist ein Objekt oder die leere Menge
->>	Mehrwertige funktionale Methode auf Instanzebene Ergebnis ist ein Objekt, mehrere Objekte oder die leere Menge
<->	Bijugat (Genau dann…, wenn...)
->	Subjugat (Wenn…, dann...)
::	Unterklassen-F-Atom
:	Is-a-F-Atom
//	Einleitung eines Kommentars in den meisten Programmiersprachen
|=	Start einer Befehlszeile in PowerLoom
=>	Satz-Operator aus dem Knowledge Interchange Format oder einwertige funktionale Methode in F-Logic auf Klassenebene
<=	Satz-Operator aus dem Knowledge Interchange Format oder einwertige funktionale Methode in die umgekehrte Richtung auf Klassenebene
=>>	Satz-Operator aus dem Knowledge Interchange Format oder mehrwertige funktionale Methode in F-Logic auf Klassenebene
>	Satz-Operator aus dem Knowledge Interchange Format, der linke Teil ist echt größer als der rechte Teil vom Operator
=	Satz-Operator aus dem Knowlegde Interchange Format, Gleichheit

Abbildungs- und Tabellenverzeichnis

Abbildungen:

Tabellen:

1 Problemstellung

Das *Wissen*, das Unternehmen zur Verfügung steht, wird in der heutigen schwierigen wirtschaftlichen Lage immer wichtiger. Wissen wird zum Überlebensfaktor und muss im Unternehmen gebunden werden, damit ein Unternehmen wirtschaftlich erfolgreich sein kann.

Wissen wird hier verstanden als die „Gesamtheit aller Kenntnisse und Fertigkeiten“[1)]. Dabei wird ein Bewusstsein vorausgesetzt, das zu einer Reflektion im Stande ist[2)]. Erst durch dieses Bewusstsein ist die Bildung, Abstraktion, Verarbeitung und Veränderung von Wissen möglich. *Kausales Wissen* ist die Beschreibung von Ursache-Wirkungs-Zusammenhängen. Das kausale Wissen ist von besonderer Relevanz, weil neues Wissen „generiert“ werden soll und Zusammenhänge bei der Generierung neuen Wissens genutzt werden sollen. Dieses neue Wissen soll genutzt werden, um z.B. neue Produkte auf den Markt zu bringen oder vorhandene so zu verbessern, dass sie erfolgreich am Markt abgesetzt werden können.

Durch den starken Druck auf Unternehmen, immer weiter zu rationalisieren, entsteht die Notwendigkeit, die Organisation eines Unternehmens an die Marktbedingungen anzupassen. Eine beispielhafte Organisation ist das Shamrock-Modell von CHARLES HANDY[3)]. Im Folgenden ist der schematische Aufbau eines Unternehmens nach diesem Modell dargestellt.

---

1) Zusammenfassung einer ausführlichen Diskussion über den Begriff Wissen: Vgl. Rehäuser/Krcmar (Wissensmanagement), S. 3-6.

2) Vgl. Specht (Wissensbasierte Systeme), S. 4-6.

3) Vgl. Handy (Age of Unreason), S. 90-115.

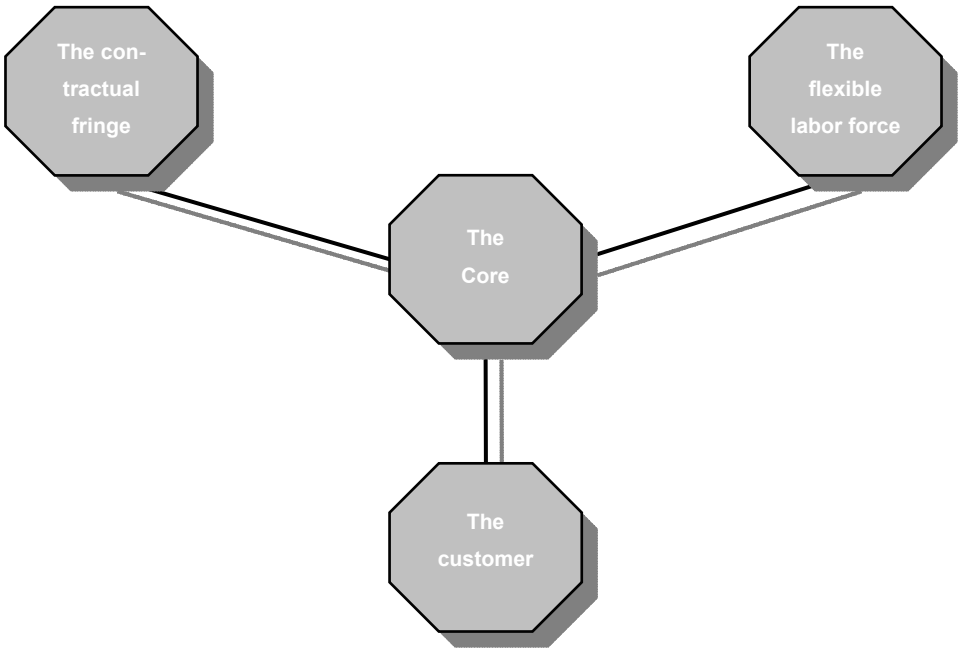

Abbildung 1: Schamrock-Organisation[4)]

In diesem Modell wird Unternehmen empfohlen, sich auf den Kern (*The Core*) zu konzentrieren. Nur die Mitarbeiter, die für das Unternehmen wichtiges Wissen enthalten und hoch-qualifiziert sind, werden fest angestellt. Diese Mitarbeiter haben aber auch eine überdurchschnittliche Leistung für das Unternehmen zu erbringen.

Daneben existieren die flexiblen Teilzeit-/Zeitarbeitmitarbeiter (*The flexible labor force*), die das Unternehmen unterstützen, wenn sie benötigt werden. Dabei müssen Unternehmen allerdings verstehen, dass auch diese Mitarbeiter als wertvoll angesehen werden müssen. Es muss auch in sie investiert werden, z.B. durch Schulungen und Trainings.

Die Zulieferer (*The contractual fringe*) sind der dritte Teil des Modells. Sie sind für die eigentliche Herstellung der Produkte zuständig und ergänzen den Kern in den Bereichen, den er nicht erfüllen kann. Die Zulieferer sind aber nur für eine bestimmte Zeit per Vertrag an den Kern gebunden, so dass der Kern auf sich ändernde Marktbedingungen reagieren kann.

---

4) Die Darstellung wird so in der Quelle nicht aufgezeigt, sie lässt sich aber durch die textuellen Beschreibungen entnehmen [Quelle: Selbst erstellt, in Anlehnung an den Text aus: Handy (Age of Unreason), S. 94-101.].

Als Ergänzung des Modells ist es möglich, den Kunden (*The customer*) mit einzubeziehen. Die Kunden können einen Teil der Arbeit übernehmen und so dem Unternehmen helfen, Kosten zu sparen. Ein Beispiel hierfür ist IKEA. Die Kunden bauen die Möbel selber auf und erhalten so einen Preisvorteil. Aber auch der umgekehrte Weg ist möglich, wie dies zum Beispiel das Kino-Unternehmen SF Bio in Schweden demonstriert. Die Buchung über das Internet ist teuerer als an der Kasse, obwohl dem Unternehmen durch die Buchung über das Internet Arbeit erspart wird.

Da sich das Unternehmen so stark auf die wenigen Mitarbeiter im Kern konzentriert, muss das Wissen möglichst im Unternehmen gehalten werden, auch wenn ein Mitarbeiter das Unternehmen verlässt. Auch das Wissen der Zulieferer und flexiblen Mitarbeiter sollte möglichst im Unternehmen gehalten werden. Wissen wird aber meist in natürlichsprachlichen Dokumenten oder „in den Köpfen" der Mitarbeiter festgehalten und ist so für andere nicht greifbar, da das Wissen meist nur implizit vorliegt.

Um Wissen aus natürlichsprachlichen Dokumenten zu extrahieren, ist die *Wissensakquisition* notwendig, die z.B. Teil der fünf Hauptkomponenten eines wissensbasierten Systems ist[5)]. *Natürlichsprachliche Dokumente* sind Dokumente, die z.B. als handschriftlicher Text oder als Word-Dateien vorliegen. Nach der Akquisition muss das Wissen in „geeigneter" Form repräsentiert werden. Zur Repräsentation gibt es verschiedene Sprachen, die im Kapitel 2 näher beschrieben werden. Dabei gibt es einen Dissens darüber, ob formale Sprachen verwendet werden müssen oder ob die natürliche Sprache zur Repräsentation ausreicht[6)].

Ist das Wissen im Unternehmen existent, so müssen im Kern des Unternehmens Mitarbeiter vorhanden sein, die dieses Wissen durch ihre Kompetenzen nutzen können. *Kompetenzen* werden hier verstanden als handlungsbefähigendes Wissen[7)].

Ein Unternehmen muss also das Management des Wissens als eines seiner Hauptziele betrachten, um am Markt erfolgreich bleiben zu können oder zu werden. Im Zusammenhang mit Wissensmanagement treten die folgenden Begriffe häufig auf: *Inferenzen*, *Deduktion* und *Schließen*, die oft als Synonyme verwendet werden[8)]. Hier wird in An-

---

5) Vgl. Bibel (Deduktion), S. 14-15.

6) Vgl. Fensel (Sinn und Unsinn), S. 1-15.

7) Vgl. Zelewski (Wissensmanagement mit Ontologien), S. 14.

8) Vgl. Bibel (Deduktion), S. 18.

lehnung an BIBEL wie folgt unterschieden: Deduktionssysteme sind das Vorliegen eines vollständigen und konsistenten Kalküls. Die Deduktion beweist durch die Anwendung eines Kalküls eine Aussage, die in einem Anwendungsfeld Gültigkeit besitzt. Dabei werden formale Schlussverfahren verwendet, die Bestandteile des Kalküls sind[9]. Schließen ist der Vorgang, den der Mensch oder der Computer vollzieht, wenn er aus vorhandenem Wissen neues Wissen generiert.[10]

Ein häufig verwendetes, deduktives Verfahren zum Schließen ist der Modus Ponens der klassischen Logik[11]:

Regel: Wenn A wahr ist, dann ist B wahr.

Faktum (Beobachtung): A ist wahr.

Schluss: B ist wahr.

Durch die Beobachtung, dass A wahr ist, und durch die vorhandene Regel, dass B wahr ist, wenn A wahr ist, folgt, dass B wahr ist.

Das *plausible Schließen* ist eine sehr verallgemeinerte Form des Modus Ponens. Die beim Modus Ponens notwendige Identität zwischen dem „Wenn"-Teil der Regel und dem Faktum wird nicht mehr unterstellt, sondern es wird von einer Ähnlichkeit ausgegangen. Der Begriff der Ähnlichkeit umfasst dabei den Bereich zwischen „Identität" und „Gegenteil". Der Modus Ponens ändert sich so zu:

Regel: Wenn x = A, dann ist y = B.

Faktum (Beobachtung): x = A'.

Schluss: y = B.

A und A' müssen ähnlich sein.

Durch die Beobachtung, dass x = A' ist, und durch die Regel, wenn x = A, dann ist y = B, folgt durch die Plausibilitätsannahme, dass y = B ist.

---

9) Vgl. Specht (Wissensbasierte Systeme), S. 52.

10) Hier wird Inferenz als Synonym von Schließen verstanden.

11) In Anlehnung an: Bibel (Deduktion), S. 106-107.

Weitere Schlüsse sind möglich durch[12]:

- die *Induktion:* leitet mutmaßliche Gesetzmäßigkeiten aus vielen Einzelbeobachtungen ab und
- die *Abduktion:* vollzieht Schlüsse von Phänomenen auf mutmaßliche Ursachen aus Kausalwissen und Beobachtungen.

Um gültige Schlüsse zu erhalten, müssen besondere formale Merkmale beachtet werden[13]. Durch die Formalisierung einer Aussage wird nicht die sprachliche Struktur verändert, sondern die Aussage präzise und eindeutig dargestellt. Die *Logik* bietet durch ihre präzisen Ausdrucksmöglichkeiten eine Möglichkeit zur Formalisierung. Daher spielt die Logik bei der Wissensrepräsentation und bei Schlussfolgerungen eine große Rolle[14]. Zum Beispiel bietet sich die *Prädikatenlogik* an, weil durch sie eine Unabhängigkeit zwischen Semantik und syntaktischer Struktur erreicht wird. Eine eingeschränkte Variante der Prädikatenlogik ist die *Horn-Logik*[15]. Sie hat bei der Berechenbarkeit bessere Eigenschaften als die Prädikatenlogik. Jedoch hat sie nur eine geringe Ausdrucksstärke als die vollständige Prädikatenlogik, da z.B. die Darstellung von Disjunktion und Negation eingeschränkt ist.

Im Folgenden wird die Betrachtung auf *Inferenzregeln* fokussiert, um aus natürlichsprachlichen Dokumenten „neues“ Wissen zu erschließen. Inferenzregeln ermöglichen den Zugriff sowohl auf das explizit[16] als auch auf das implizit[17] in natürlichsprachlichen Dokumenten[18] vorhandene Wissen.

Allerdings ist zu beachten, dass BIBEL unterschiedliche Bedeutungen der Begriffe verwenden. Durch die Inferenzregeln und die vorhandenen Fakten wird das „neue“, impli-

---

12) Vgl. Specht (Wissensbasierte Systeme), S. 52.

13) Vgl. Bibel (Deduktion), S. 17.

14) Vgl. Bibel (Wissensrepräsentation), S. 24-26.

15) Vgl. Bibel (Wissensrepräsentation), S. 97-99.

16) Explizit geht zurück auf die Unterscheidung von explizitem und implizitem Wissen durch Nonaka / Takeuchi (Wissen), S. 18-22. Explizites Wissen ist Wissen, welches klar dargelegt ist.

17) Vgl. Nonaka / Takeuchi (Wissen), S. 18-22.

18) Vgl. Bibel (Deduktion), S. 15.

zit vorhandene Wissen expliziert. Hierzu wird meist eine Logik verwendet[19].

Der Begriff der Ontologie wird in den Forschungsgebieten über Wissen und das Management von Wissen häufig verwendet. Eine oft verwendete Definition für eine Ontologie geht auf GRUBER zurück. Hiernach ist eine Ontologie: *„an explicit specification of a conceptualization“*[20].

(Stand der Beispiele und des Anhangs des Projektberichts ist 04/2003.)

19) Vgl. Specht (Wissensbasierte Systeme), S. 56.

20) Vgl. Gruber (Ontology Specifications), S. 1.
Übersetzung: Eine Ontologie ist eine explizite Spezifikation einer Konzeptualisierung.

2 Sprachen zur Darstellung von Inferenzregeln

Die Darstellung von Wissen muss für seine Verwendung in Computern so aufbereitet werden, dass das Wissen aus den verarbeiteten Dokumenten formalsprachlich repräsentiert werden kann. Dazu muss eine entsprechende Darstellungsform gefunden werden. Diese muss sowohl eine Syntax als auch eine Semantik besitzen[21]. Dies ist notwendig, damit sowohl die Struktur als auch die Bedeutung festgelegt ist. Die Syntax ist der Aufbau einer Sprache aus Zeichen und Ausdrücken. Die Semantik ist die, durch Interpretation erschließbare Bedeutung. Um das Inferenzieren von „neuem" Wissen zu ermöglichen, muss es möglich sein, Inferenzregeln darzustellen und diese auch ausführen zu können. In den nächsten Abschnitten werden daher verschiedene Formen zur Darstellung von Inferenzregeln vorgestellt.

2.1 Natürliche Sprache

Eine häufig verwendete Form zur Repräsentation von Wissen ist die natürliche Sprache. Soll sie maschinell weiterverwendet werden, so muss sie mit einer formalen Semantik versehen werden[22]. Nur so ist ihre Eindeutigkeit gegeben. Ein Satz kann aber, auch wenn er syntaktisch korrekt gebildet ist, mehrere Bedeutungen haben. Der mit einem Satz verbundene Kontext ermöglicht erst seine Eindeutigkeit.

Zum Beispiel ist der Satz: „Es ist außen später als nachts" syntaktisch korrekt, allerdings ist die Semantik nicht verständlich. Würde man den Satz: „Außen ist es kälter als innen" nehmen, so wären sowohl die Syntax als auch die Semantik korrekt.

Diese Eindeutigkeit muss für die weitere Verwendung in einem wissensbasierten System hergestellt werden. Die Verarbeitung der natürlichen Sprache ist daher ein eigenes Forschungsgebiet, auf das hier nicht näher eingegangen wird. Als Möglichkeit zur Darstellung von Inferenzregeln in maschineller Form scheidet die natürliche Sprache somit aus, allerdings wird sie zur Erläuterung der Beispiele im Kapitel 3 verwendet.

---

21) Vgl. Flores-Mendez (Reason Logically), S. 1.

22) Vgl. Bibel (Wissensrepräsentation), S. 22-23.

2.2 Formale Sprachen

2.2.1 Logik-basierte Sprachen

2.2.1.1 F-Logic

F-Logic ist eine logik-basierte Sprache, die die Repräsentation von Wissen und Inferenzregeln ermöglicht[23]. F-Logic steht dabei für Frame Logic. Die Frame Logic kann Beziehungen zwischen Objekten darstellen. So wird so eine Modellierung im objektorientierten Stil möglich. F-Logic ist eine deklarative Sprache[24]. Die Syntax ist der Prädikatenlogik erster Ordnung ähnlich. In F-Logic sind auch atomare Ausdrücke möglich, die Moleküle genannt werden[25].

Ein Beispiel für ein Molekül ist:

X

(X ist eine Klasse, die nicht weiter zerlegt werden kann.)

Hier kann nur in kurzer Form auf das Modell von F-Logic eingegangen werden, eine Einführung zu F-Logic findet sich bei KIFER[26].

Die folgende Tabelle gibt eine Darstellung syntaktischer Konstrukte und ihrer Semantik:

23) Vgl. Decker (Domain-Specific Declarative), S. 1.

24) Eine deklarative Sprache ist eine Sprache, die erklärend ist.

25) Atomare Ausdrücke lassen sich nicht weiter zerlegen.

26) Vgl. Kifer / Lausen (F-Logic).

syntaktisches Konstrukt	Semantik
X::Y	X ist eine Subklasse von Y
X:Y	X ist eine Instanz von Y
Y[Z->X]	die Instanz Y hat eine Methode Z mit dem Wert X; Z wird oftmals auch als Eigenschaft/Attribut mit der Ausprägung X verwendet
Y[Z=>X]	alle Instanzen der Klasse Y verwenden die Methode Z und die Klasse der Ergebnisse ist X

Tabelle 1: Darstellung von syntaktischen Konstrukten
(Quelle: in Anlehnung an Decker (Domain-Specific Declarative), S. 3)

Dies ist nur ein kleiner Ausschnitt der syntaktischen Darstellungsmöglichkeiten und ihrer semantischen Bedeutungen. Um komplexere Ausdrücke darstellen zu können, ist eine Kombination der syntaktischen Konstrukte durchführbar.

In F-Logic sind auch Inferenzregeln implementierbar, sodass eine spätere Inferenzierung von neuem Wissen ermöglicht wird.

Ein Beispiel wäre:

```
FORALL X, Y
        Regel-Kopf
X:mann[sohn->>Y]
        Dann-Komponente (Regel-Kopf)
<- Y:mann[vater->X].
        Wenn-Komponente (Regel-Körper)
```

Eine Regel wird folgendermaßen gelesen: Wenn der Regel-Körper erfüllt ist, so gilt auch der Regel-Kopf. In diesem Beispiel bedeutet dies: Wenn Y ein Mann ist, dessen Vater X ist, dann ist Y Sohn von X.

2.2.1.2 Knowledge Interchange Format

Das KIF (Knowledge Interchange Format) basiert auf der monotonen deduktiven[27)] Logik. Es wurde konstruiert, um Wissen zwischen verschiedenen Computern austauschen zu können. Hierbei stand die Menschenlesbarkeit nicht im Vordergrund[28)]. Ein Computer, der Wissen verarbeiten soll, muss das Wissen in ein eigenes Repräsentationsformat konvertieren. KIF soll nur als Austauschformat verwendet werden. Die wesentlichen Bestandteile von KIF sind: eine beschreibende Semantik und eine verständliche Logik. Hierdurch soll die Sprache gut zu implementieren sein. Da in dieser Sprache keine Regeln vorgesehen sind und der Fokus auf dem Austausch von Wissen zwischen Computern liegt, unterbleibt eine eingehende Betrachtung.

2.2.1.3 Ontolingua

Ontolingua ist ein Werkzeug zur Beschreibung von Ontologien, das zu verschiedenen Beschreibungssprachen kompatibel ist und die Konvertierung in diese Sprachen ermöglicht[29)]. Die Syntax und Semantik basiert auf KIF, das stark erweitert wurde. Ontolingua wird dazu verwendet, Dokumente, die in einer deklarativen Sprache geschrieben sind, in ein von Repräsentationssystemen verwendetes Format zu konvertieren. Der Vorteil liegt darin, dass unterschiedliche Benutzer die verschiedensten Sprachen verwenden können. Die Sprachen werden in Ontolingua in ein standardisiertes Format konvertiert und können so ausgetauscht werden. Die Beschreibungssprachen, die Ontolingua kennt und übersetzen kann, werden in einer sog. Frame-Ontologie festgelegt. Ontolingua soll, wie KIF, nur als ein erweitertes Austauschformat dienen[30)]. Es erfolgt keine weitere Betrachtung im Zusammenhang mit Regeln, weil auch hier Regeln nicht dargestellt werden können.

---

27) Monoton heißt, dass durch Hinzufügen neuen Wissens die Gültigkeit alten Wissens nicht beeinflusst wird. Vgl. Bibel (Wissensrepräsentation), S. 112-114.

28) o.V. (Knowledge Interchange Format), S. 2-4.

29) Vgl. Gruber (Ontology Specifications), S. 5.

30) Vgl. Gruber (Ontology Specifications), S. 5.

2.2.1.4 Loom

Loom[31] ist ein Wissensrepräsentationssystem[32]. Es besteht insgesamt aus sechs Komponenten:

1. Abfragemöglichkeit
2. „Classifier“[33]
3. Produktionsregel-System
4. Kontext-Mechanismus
5. Standard-Schlussfolgerer
6. Zeit-Schlussfolgerer

Loom wurde erstellt, um die Konstruktion von „intelligenter“ Software zu unterstützen[34]. Die Schlussfolgerungsmächtigkeit wird durch eine Inferenzmaschine hergestellt, die „Classifier“ genannt wird. Der Loom-“Classifier“ hat die Aufgabe, die Hierarchie zwischen Konzepten und Instanzen sowie die dadurch bestehenden Beziehungen zwischen Konzepten und Instanzen zu verarbeiten. Loom kennt zwei verschiedene logische Sprachen:

- Description Logic
- prädikatenlogikbasierte Sprache

Die prädikatenlogikbasierte Sprache ist KIF sehr ähnlich[35]. Wissen, das in Loom repräsentiert wird, wird in die Kategorien Definition, Implikation, Produktions-Regel und Aussage eingeteilt.

Beispiele zu den Kategorien:

(defconcept A)

Definition eines Konzepts

(implies AB C)

Festlegung einer Implikation zwischen den Konzepten

---

31) Weitere Informationen: o.V. (LOOM).

32) Vgl. MacGregor (Retrospective on Loom), S. 2-4.

33) Classifier = deutsch: Klassifizierer.

34) Vgl. MacGregor (Deductive Inference), S. 1-5.

35) Vgl. MacGregor (Retrospective on Loom), S. 5.

A und B mit dem „Namen“ C

```
(tell (A Müller) (B Müller))
```

Definition einer Aussage, A = Müller, B = Müller

```
(ask (AB Meier))
```

Produktions-Regel

In Loom ist deduktive Inferenz möglich. Dadurch kann aus zwei gegebenen Beschreibungen von Klassen durch den „Classifier“ geschlossen werden, dass die eine Subklasse der anderen ist oder dass sie äquivalent oder disjunkt sind. Auch Reifikation[36)] ist in Loom möglich.

Ein Beispiel für eine einfache Inferenz in Loom ist:

```
(implies (>= Alter 18) Volljährig)
```

In natürlichsprachlicher Form bedeutet dies, dass, wenn das Alter einer Entität größer oder gleich 18 ist, dann ist diese Entität volljährig.

2.2.1.5 PowerLoom

PowerLoom ist der Nachfolger von Loom und wird zurzeit in der Version 2.0 entwickelt[37)]. PowerLoom ist ein Wissensrepräsentations- und Schlussfolgerungssystem[38)]. Um die Wissensrepräsentation und Schlussfolgerungen zu ermöglichen, wird zwischen zwei Teilen unterschieden:

- eine logik-basierte Sprache, die eine Variante von KIF ist,
- eine deduktive Komponente, die auf Prolog basiert.

---

36) Reifikation ist eine Aussage über eine Aussage.

37) Weitere Informationen: o.V. (PowerLoom).

38) Vgl. MacGregor / Chalupsky / Melz (PowerLoom Manual), S. 1.

Zurzeit kann die deduktive Komponente Horn-Regeln, Negation, einfache Gleichheitsregeln und rekursive[39] Regeln verarbeiten. In Version 2.0 sind auch komplexere Regeln möglich. Für die Implementierung von PowerLoom wurde die Programmiersprache STELLA[40] entworfen, die Lisp sehr ähnlich ist.

Eine Regel wird in PowerLoom wie folgt repräsentiert:

```
¦= (defrule BEISPIEL
```

zunächst wird die Regel definiert und ein Name „BEISPIEL“ vergeben

```
(forall (?x JUNGE) (?y MAEDCHEN)
```

anschließend wird mit forall der Regelkopf eingeleitet (?x und ?y sind Variablen, JUNGE und MAEDCHEN sind Klassen)

```
(<= (hat_Schwester ?x ?y))
```

Wenn-Komponente

```
(hat_Bruder ?y ?x)))
```

Dann-Komponente

Natürlichsprachliche Erläuterung:

Wenn die Variable ?x, die ein Junge ist, eine Schwester hat, die die Variable ?y ist, dann wird festgelegt, dass ?y einen Bruder ?x hat.

2.2.2 XML-basierte Sprachen

Um Ontologien und das darin enthaltene Wissen einer großen Anzahl von Personen zugänglich zu machen, werden Repräsentationsformate entwickelt, die über das Internet

39) Rekursiv bedeutet, dass die Ausführung so lange immer ein gleiches Programmteil durchläuft, bis eine End-Bedingung erreicht ist.

40) Vgl. zu STELLA: o.V. (STELLA).

ausgetauscht werden können. Als Grundlage wird XML verwendet. XML[41] ist die Extensible Markup Language. XML bietet in der Grundform keine strikte Syntax, es handelt sich daher eher um eine Metasprache. Eine Abbildung von Semantik ist nicht vorgesehen[42]. Es werden nur die Regeln für ein syntaktisch korrektes Dokument im Standard des W3C beschrieben[43].

Jedes XML-Dokument besteht aus einer Abfolge von Elementen (Tags). Zu Beginn wird jeweils ein Start-Tag und ein End-Tag verwendet, zwischen denen die eigentliche Information eingefügt wird.

Ein Beispiel ist:

<Mitarbeiter> Name des Mitarbeiters </Mitarbeiter>

Im Start-Tag können zusätzlich Attribute angegeben werden. Alle Elemente zusammen bilden eine Hierarchie. Durch die fehlende Möglichkeit der Darstellung von Semantik wird XML als eigenständige Sprache nicht weiter betrachtet.

41) Informationen zu XML: o.V. (XML).

42) Vgl. Staab (Semantic Web), S.3.

43) Vgl. auch in Erdmann / Studer (XML Documents), S 1-2.

2.2.2.1 XOL

XOL wird als eine XML-basierte Ontologie-Austauschsprache beschrieben[44]. Dabei stützt sich XOL auf OKBC[45]. XOL ist eine Document Type Definition (DTD)[46]. In einer DTD wird die Syntax festgelegt, in welcher die einzelnen Tags verwendet werden dürfen. Zudem wird die Semantik der Tags definiert. Zusätzlich ist es möglich, eigene Tags in der DTD zu spezifizieren. Durch die Weitergabe der DTD an möglichst viele Personen wird eine Standardisierung der Dokumente erreicht, da alle mit dieser DTD erstellten Dokumente hinsichtlich ihrer Syntax und Semantik gleich sind. So wird ein Austausch der Ontologien untereinander möglich. Da in der DTD von XOL die Darstellung von Inferenzregeln nicht vorgesehen ist, wird keine weitere Betrachtung dieser Sprache durchgeführt.

2.2.2.2 RDF und RDF-Schema

Eine Anwendung von XML ist RDF. RDF[47], das Ressource Description Framework, hat ein einfaches Modell[48]:

- Jede Beziehung wird syntaktisch dargestellt als ein Tripel, ein Beispiel wäre: ein Autor ist Experte in Rechtschreibung.
- Jede Ressource[49] wird durch eine URI (uniform ressource identifier) eindeutig identifiziert, eine URI kann z.B. „http://www.pim.uni-essen.de/Autor“ sein[50].

RDF-Schema[51] ist eine Erweiterung zu RDF, die weitere Elemente (Tags) bietet. Ein Schema legt die verwendbaren Elemente fest. Eine Darstellung von Klassen, Klassen-Hierarchien, Eigenschaften und Eigenschafts-Hierarchien wird ermöglicht.

---

44) Vgl. Karp / Chaudhri / Thomere (XOL), S 2-3.

45) OKBC ist die Open Knowledge Base Connectivity, eine Schnittstelle, um auf Wissensbasen zugreifen zu können. Weitere Informationen unter: o.V. (OKBC) oder Chaudhri / Farquhar / Fikes / Karp / Rice (Open Knowledge).

46) DTD = Document Type Defintion. Die DTD ist ein Schema, um die in einem XML-Dokument verwendbaren Tags einzuschränken und so eine stärkere Syntaxfestlegung zu erreichen. Gegen die DTD kann eine Prüfung auf Gültigkeit des Dokuments erfolgen.

47) Weitere Informationen: Miller / Swick / Brickley (RDF).

48) Unter einem Modell versteht man ein abstraktes Abbild eines Systems.

49) Eine Ressource kann eine Webseite oder jedes beliebige reale Element sein.

50) Allerdings besteht hier die Problematik, dass sich das Format der URIs noch in der Entwicklung befindet. Weiter Informationen hierzu: Connolly (Naming).

51) Weitere Informationen: McBride (RDF Schema).

Ein Beispiel für die Syntax von RDF und RDF-Schema ist:

```
<rdfs:Class rdf:ID=„Kino“>
</rdfs:Class>
<rdfs:Class rdf:ID=„Kinosäle“>
  <rdfs:subClassOf rdf:resource=„Kino“>
        </rdfs:subClassOf>
</rdfs:Class>
```

Zunächst wird eine Klasse festgelegt, die die ID Kino erhält. Durch die Klassenfestlegung wird Kino als Ressource definiert. Anschließend wird eine weitere Klasse mit der ID Kinosäle festgelegt. In dieser Klasse wird zusätzlich festgelegt, dass sie eine Subklasse der Klasse Kino ist.

RDF und RDF-Schema verwenden die gleiche Syntax wie XML. Jedes RDF(S)-Dokument besteht aus einer Folge von Tags. Hier wird zusätzlich durch die Festlegung der verwendbaren Tags die Semantik festgelegt. Aber auch RDF(S) enthält nur Basis-Modellierungs-Elemente zur Darstellung von Ontologien[52)]. Allerdings ist die Reifikation, also eine Aussage über eine Aussage, möglich und direkt in RDF vorgesehen. Die Reifikation könnte in der Praxis von Relevanz sein, um z.B. die Güte einer Aussage zu klassifizieren.

52) Vgl. Staab / Erdmann / Maedche / Decker (Extensible Approach), S. 1-3.

Um den Anwendungsbereich von RDF(S) zu vergrößern, ist es möglich, durch die Verwendung von Namensräumen und Reifikation eigene Standards im RDF(S)-Format zu definieren. Ein Namensraum ermöglicht die Unterscheidung zwischen unterschiedlichen Modellierungsebenen[53]. Die Darstellung des gesamten Modells von RDF ist hier aus Platzgründen nicht möglich, es sei aber auf die Einführungen zum Thema verwiesen[54].

RDF ist nicht auf die Darstellung in XML beschränkt. Zusätzlich ist eine Darstellung in 3-Tupeln und azyklischen, gerichteten, beschrifteten Grafen möglich[55].

Die Inferenzmöglichkeiten in RDF(S) sind sehr eingeschränkt. Denn nur die Konstrukte subClassOf und subPropertyOf ermöglichen einfache Inferenzen[56]. RDF selbst ist als Basis für komplexe Inferenzen und Anfragen nicht geeignet, da dies im Metamodell der Sprache nicht vorgesehen ist. Daher muss eine andere Sprache zur Implementierung gefunden werden[57]. Im Modell des Semantic Web, in dem RDF eine zentrale Rolle spielt, ist vorgesehen, dass Regeln auf RDF aufgesetzt werden[58].

Es gibt allerdings erweiterte Sprachen, die die Inferenzierung ermöglichen sollen. Beispiele für solche Sprachen[59] sind[60]: RQL, Versa, RDF Squish Query Language, Triple und RDQL.

- RQL basiert auf einem formalen Grafenmodell, welches die RDF-Modellierungselemente enthält, und sie um Abfragemöglichkeiten erweitert.

---

53) Modellierungsebenen sind verschiedene Ebenen eines Modells. Z.B. wird in der Software-Entwicklung oft das 3-Ebenen-Modell verwendet, welches die Entwicklung in die Bereiche Benutzerebene, Anwendungslogik und Datenbankebene trennt.

54) Einführungen in RDF finden sich in: Bray (RDF and Metadata) und Bray (What is RDF).

55) Vgl. Decker / Brickley / Saarela / Angele (Inference Service), S. 1.

56) Vgl. Guha / Lassila / Miller / Brickley (Enabling Inferencing), S. 4.

57) Vgl. Decker / Brickley / Saarela / Angele (Inference Service), S. 5-6.

58) Vgl. Staab (Semantic Web), S. 4.

59) Zudem existieren weitere Sprachen wie Nexus [Vgl. o.V. (Nexus).].

60) Vgl. Ogbuji (Techniques for Knowledge Management), Miller (Squish query language), Sintek/Decker (Triple), Karvounarakis/Christophides/Alexaki/Plexousakis/Scholl (Declarative Query Language).

- Versa konzentriert sich auf die Betrachtung von Knoten und Kanten im RDF-Modell. Es bietet ein Datenmodell, Funktionen und Primitive für flexible Suchanfragen. Zusätzlich sind transitive[61] Operationen und Aggregationen möglich[62].
- Die RDF Squish Query Language ist eine Abfragesprache für eine einfache Grafen-Navigation. Dabei werden Teilgrafen-Übereinstimmungen überprüft.
- Triple ist eine Regel-Sprache. Sie basiert auf der Horn Logik, die syntaktisch erweitert wurde um RDF-Modellierungselemente.
- RDQL ist eine Query-Sprache[63], die von Hewlett-Packard entwickelt wurde. Sie wird durch das Jena-Toolkit unterstützt, ermöglicht allerdings nur Abfragen.

Die drei erstgenannten Sprachen ermöglichen nur die Implementierung von Anfragen.

2.2.2.3 DAML + OIL

DAML + OIL[64] ist aus den zwei separaten Entwicklungen DAML[65] und OIL[66] entstanden. DAML + OIL basiert auf XML[67]: XML ist die Syntax für RDF, und RDF ist die Syntax für DAML+OIL. Gleichzeitig ist DAML + OIL auch eine Teilsprache von RDF, da einige Ausdrücke in RDF geschrieben werden[68]. Besonders zu erwähnen ist die Erweiterung von RDF um Negation, Konjunktion und Disjunktion sowie um Typen von Eigenschaften. Für die Inferenzierung von Wissen stellt DAML + OIL zusätzliche Möglichkeiten zur Verfügung[69]. Dazu gehören:

- transitive Eigenschaften
- inverse Klassen
- eindeutige Eigenschaften

---

61) Transitiv bedeutet, dass, wenn A = B und B = C gilt, auch A = C gilt.

62) Eine detailliertere Beschreibung findet sich unter: Ogbuji (Versa).

63) Weitere Informationen: o.V. (RDQL).

64) Weitere Informationen: o.V. (Daml + OIL).

65) Weitere Informationen: o.V. (Darpa Agent).

66) Weitere Informationen: o.V. (OIL).

67) Vgl. Gil / Ratnakar (Markup Languages), S. 2.

68) Eine detaillierte Einführung findet sich unter: Ouellet / Ogbuji (Introduction to DAML).

69) Vgl. o.V. (Language Feature Comparison), S. 1-2.

- disjunkte Klassen

Die Implementierung von Inferenzregeln wird aber nicht unterstützt. Wie bei RDF besteht auch hier die Möglichkeit, die Sprache Triple für die Implementierung von Inferenzregeln zu verwenden, und eine Darstellung von Triple in DAML + OIL ist denkbar[70]. Als weitere Möglichkeit wird im Moment die DAML-Query-Language entwickelt[71].

70) Vgl. Sintek / Decker (TRIPLE), S. 1.

71) Weitere Informationen: o.V. (DAML).

2.2.2.4 OXML

OXML[72] ist die interne Repräsentationssprache, die von OntoEdit[73] verwendet wird. Sie ist eine XML-Anwendung[74]. OXML besteht aus acht Komponenten, die jeweils einen bestimmten Tag besitzen. Zusätzlich erfolgt ähnlich wie in RDF eine eindeutige Identifizierung durch einen Namespace. Die acht Komponenten sind: Ontologie, Konzept, Relation, Instanz eines Konzepts, Axiom, Prädikat, Instanz eines Prädikats und Modul. Die Relationen können Eigenschaften besitzen, wodurch Symmetrie, Reflexivität, Transitivität, Antisymmetrie, Irreflexivität und Intransitivität dargestellt werden können. Axiome zur Darstellung von Regeln werden in OXML in F-Logic dargestellt. Daher können die später zu F-Logic gemachten Aussagen bezüglich der Inferenzmächtigkeit auch auf OXML übertragen werden. Ein Nachteil von OXML ist aber, dass es kein anerkannter Standard ist. Dadurch wird ein Austausch des in diesen Ontologien vorhandenen Wissens erschwert. Außerdem ist zurzeit keine Inferenzmaschine erhältlich, die OXML direkt verarbeiten kann. Eine Inferenzierung ist möglich durch Umwandlung von OXML in F-Logic. Eine Aussage über die Erhaltung der Bedeutung ist allerdings nicht möglich. Eine weitere Betrachtung erfolgt daher nicht.

---

72) Weitere Informationen zu OXML: o.V. (Ontoprise).

73) OntoEdit wurde vom Institut AIFB der Universität Karlsruhe entwickelt und wird nun kommerziell über die Ontoprise GmbH vertrieben.

74) Vgl. Erdmann (OXML 2.0), S. 2-4.

3 Implementierung von Inferenzregeln in ausgewählten Sprachen

3.1 Beispiele in natürlicher Sprache

Bei der Akquisition von Wissen über Kompetenzen aus natürlichsprachlichen Dokumenten soll aus dem Wissen, welches schon vorhanden ist, neues Wissen generiert werden und implizit vorhandenes Wissen expliziert werden. Im Folgenden werden zunächst 21 Inferenzregeln in natürlicher Sprache konzipiert. In den nächsten Abschnitten werden die ersten fünf Regeln in unterschiedlichen formalen Sprachen implementiert. Ausnahme: In F-Logic werden alle Regeln implementiert.

- 1. Regel: Wenn ein Mitarbeiter an einer Schulung über eine bestimmte Thematik teilgenommen hat und das Niveau der Schulung professionell war, dann verfügt er über eine Kompetenz im Gegenstandsbereich der Schulung, mit der Ausprägung „Anfänger".
- 2. Regel: Wenn ein Mitarbeiter einen Chef hat und dieser Chef wiederum einen Chef hat, dann hat auch der Mitarbeiter den letztgenannten Chef.
- 3. Regel: Wenn ein Mitarbeiter mindestens zwei Projekte erfolgreich geleitet hat, dann besitzt er Führungsfähigkeiten in Bezug auf Projekte. Die Projekte müssen allerdings als „erfolgreich" bewertet worden sein.
- 4. Regel: Wenn ein Mitarbeiter in mindestens zwei Projekten gearbeitet hat und beide Projekte als „erfolgreich" bewertet wurden, dann besitzt er Teamfähigkeit.
- 5. Regel: Wenn ein Mitarbeiter einen Chef hat, dann hat der Chef auch Weisungsbefugnis über den Mitarbeiter.
- 6. Regel: Wenn ein Mitarbeiter mindestens zwei Software-Produkte programmiert hat, bei denen jeweils Java als Programmiersprache eingesetzt wurde und die Ergebnisse mindestens „zufrieden stellend" waren, dann folgt daraus, dass er Kompetenz in Java besitzt.
- 7. Regel: Wenn ein Mitarbeiter „Experte" in Mathematik ist, dann ist seine Kompetenz bei der Konstruktion „lernfähig".
- 8. Regel: Wenn ein Mitarbeiter „Experte" in Mathematik und „Experte" in Physik ist, dann ist er als Konstrukteur „lernfähig".

- 9. Regel: Wenn ein Mitarbeiter „Experte“ in logischem Denken ist, dann ist er „Erfahrener“ in technischen Abläufen.
- 10. Regel: Wenn ein Mitarbeiter „Experte“ in logischem Denken ist, dann ist er gut geeignet für Projekte.
- 11. Regel: Wenn ein Mitarbeiter im Ausland gelebt hat und in diesem Land nicht die Sprache „Deutsch“ gesprochen wird, dann ist er „Erfahrener“ in dieser Sprache.
- 12. Regel: Wenn ein Mitarbeiter früher in einem anderen Unternehmen gearbeitet hat und dieses Unternehmen in einer Branche tätig ist, dann ist er „Erfahrener“ in dieser Branche.
- 13. Regel: Wenn ein Mitarbeiter früher in einem anderen Unternehmen gearbeitet hat, dann hat er Kenntnisse als „Erfahrener“ über dieses Unternehmen.
- 14. Regel: Wenn ein Mitarbeiter mit einem Kunden gearbeitet hat, dann hat er Kenntnisse über diesen Kunden, mit der Ausprägung „Erfahrener“.
- 15. Regel: Wenn ein Bewerber ein Praktikum, in einem Unternehmen absolviert hat und das Unternehmen in einer Branche tätig ist, dann hat er Kenntnisse über die Branche, mit der Ausprägung „Anfänger“.
- 16. Regel: Wenn ein Mitarbeiter Schulungen zu einem bestimmten Gegenstandsbereich durchgeführt hat, dann hat er diese Kompetenz und die Kompetenz „Präsentationsfähigkeit“, mit den Ausprägungen „Erfahrener“.
- 17. Regel: Wenn ein Mitarbeiter in einem Projekt gearbeitet hat und im Projekt nicht deutsch gesprochen wurde, dann hat er Fremdsprachenkompetenz, mit der Ausprägung „Erfahrener“.
- 18. Regel: Wenn ein Mitarbeiter Autor einer Publikation ist, die in einer anderen Sprache als „Deutsch“ geschrieben wurde, dann hat er Fremdsprachenkompetenz, mit der Ausprägung „Erfahrener“.
- 19. Regel: Wenn ein Mitarbeiter Autor einer Publikation ist, die ein Thema oder mehrere Themen hat, dann hat er in diesen Themen Kompetenzen, mit den Ausprägungen „Erfahrener“.
- 20. Regel: Ein Mitarbeiter, der einen Vortrag auf einem Workshop zu einem Thema gehalten hat, hat Kompetenzen in diesem Thema, mit der Ausprägung „Erfahrener“.

Ein Beispiel für eine grafische Darstellung einer Regel stellt Abbildung 2 dar:

Abbildung 2: Grafische Darstellung einer Regel

3.2 Beispiele in F-Logic

Im Folgenden werden alle Regeln in F-Logic implementiert.

1. Regel:

```
FORALL Mitarbeiter1, Vorgang1, Kompetenz1, REL
Mitarbeiter1[REL->Anfaenger]<-
        Vorgang1: Schulung  [hat_Gegenstandsbereich->>Kompetenz1;
                             hat_Schulungsniveau->professionell;
                             hat_Teilnehmer->>Mitarbeiter1]
                             AND zugeordnet(Kompetenz1,REL).
```

Beispiel: zugeordnet (Java,hat_Java_Kompetenz)

2. Regel:

```
FORALL Mitarbeiter1, Mitarbeiter2, Mitarbeiter3
Mitarbeiter1 [hat_Vorgesetzten->>Mitarbeiter3]<-
        Mitarbeiter1:Mitarbeiter [hat_Vorgesetzten->>Mitarbeiter2] AND
        Mitarbeiter2:Mitarbeiter [hat_Vorgesetzten->>Mitarbeiter3].
```

3. Regel:

FORALL Mitarbeiter1, Projekt1, Projekt2

Mitarbeiter1[hat_Fuehrungsfaehigkeiten->Experte]<-

Mitarbeiter1:Mitarbeiter[leitet_Projekt->>Projekt1] AND

Mitarbeiter1:Mitarbeiter[leitet_Projekt->>Projekt2] AND

NOT EQUAL (Projekt1, Projekt2) AND

Projekt1:Projekt[hat_bewertung->erfolgreich] AND

Projekt2:Projekt[hat_bewertung->erfolgreich].

4. Regel:

FORALL Mitarbeiter1, Projekt1, Projekt2

Mitarbeiter1[hat_Teamfaehigkeit->Erfahrener]<-

Mitarbeiter1:Mitarbeiter[arbeitet_in_Projekt->>Projekt1] AND

Mitarbeiter1:Mitarbeiter[arbeitet_in_Projekt->>Projekt2] AND

NOT EQUAL (Projekt1, Projekt2) AND

Projekt1:Projekt[hat_bewertung->erfolgreich] AND

Projekt2:Projekt[hat_bewertung->erfolgreich].

5. Regel:

FORALL Mitarbeiter1, Mitarbeiter2

Mitarbeiter1:Mitarbeiter[hat_Vorgesetzten->Mitarbeiter2]<->

Mitarbeiter2:Manager [hat_Weisungsbefugnis_ueber->>Mitarbeiter1].

6. Regel:

```
FORALL Mitarbeiter1, Software1, Software2
Mitarbeiter1[hat_Javakompetenz->Erfahrener]<-
        Mitarbeiter1:Mitarbeiter[hat_programmiert_Software->>Software1] AND
        Mitarbeiter1:Mitarbeiter[hat_programmiert_Software->>Software2] AND
        Software1:Software[ist_programmiert_in_Programmiersprache->Java] AND
        Software2:Software[ist_programmiert_in_Programmiersprache->Java] AND
        NOT EQUAL (Software1,Software2) AND
        Software1:Software[hat_Softwarebewertung->zufrieden_stellend] AND
        Software2:Software[hat_Softwarebewertung->zufrieden_stellend].
```

7. Regel:

```
FORALL Mitarbeiter1
Mitarbeiter1[hat_Konstruktionskompetenz->lernfaehig]<-
        Mitarbeiter1[hat_Mathematikkompetenz->Experte].
```

8. Regel:

```
FORALL Mitarbeiter1
Mitarbeiter1[hat_Konstruktionskompetenz->lernfaehig]<-
        Mitarbeiter1[hat_Mathematikkompetenz->Experte] OR
        Mitarbeiter1[hat_Physikkompetenz->Experte].
```

9. Regel:

```
FORALL Mitarbeiter1
Mitarbeiter1[hat_technische_Ablaufkompetenz->Erfahrener]<-
        Mitarbeiter1:Mitarbeiter[hat_Logikkompetenz->Experte].
```

10. Regel:

FORALL Mitarbeiter1, Projekt1

Mitarbeiter1[geeignet_fuer->Projekt1]<-

Mitarbeiter1:Mitarbeiter[hat_Logikkompetenz->Experte] AND

Projekt1:Projekt.

11. Regel:

FORALL Mitarbeiter1, Sprache1, Land1, REL

Mitarbeiter1[REL->Erfahrener]<-

Mitarbeiter1:Mitarbeiter[lebte_in_Staat->Land1] AND

Land1:Staat[hat_Sprache->Sprache1] AND

NOT EQUAL (Sprache1, „deutsch“) AND

zugeordnet (Sprache1,REL).

Beispiel: zugeordnet (englisch, hat_Englisch_Kompetenz)

12. Regel:

FORALL Mitarbeiter1, Unternehmen1, Branchenklassifikation1, REL

Mitarbeiter1[REL->Erfahrener]<-

Mitarbeiter1:Mitarbeiter[hat_gearbeitet_fuer->>Unternehmen1] AND

Unternehmen1[taetig_in_Branche->>Branchenklassifikation1] AND

zugeordnet(Branchenklassifikation1,REL).

Beispiel: zugeordnet(Maschinenbau,hat_Maschinenbaukompetenz)

13. Regel:

FORALL Mitarbeiter1, Unternehmen1, REL

Mitarbeiter1[REL->Erfahrener]<-

Mitarbeiter1:Mitarbeiter[hat_gearbeitet_fuer->>Unternehmen1] AND

zugeordnet(Unternehmen1,REL).

Beispiel: zugeordnet(DMT,hat_DMT_Kenntnisse)

14. Regel:

FORALL Mitarbeiter1, Kunde1, REL

Mitarbeiter1[REL->Erfahrener]<-

Mitarbeiter1:Mitarbeiter[hat_gearbeitet_mit->>Kunde1] AND

zugeordnet(Kunde1,REL).

Beispiel: zugeordnet(Aldi,hat_Aldi_Kenntnisse)

15. Regel:

FORALL Mitarbeiter1, Unternehmen1, Branchenklassifikation1, REL

Bewerber[REL->>Anfaenger]<-

Person1:Bewerber[hat_Praktikum_gemacht->>Unternehmen1] AND

Unternehmen1[taetig_in_Branche->>Branchenklassifikation1] AND

zugeordnet(Branchenklassifikation1,REL).

Beispiel: zugeordnet(Maschinenbau,hat_Maschinenbau_Kenntnisse)

16. Regel:

FORALL Mitarbeiter1, Kompetenz1, Schulung1, REL

Mitarbeiter1[REL->>Erfahrener) AND

Mitarbeiter1[hat_Praesentationsfaehigkeit->Erfahrener] <-

Mitarbeiter1:Mitarbeiter[hat_durchgefuehrt->>Schulung1] AND

Schulung1:Schulung[hat_Gegenstandsbereich->>Kompetenz1) AND

zugeordnet(Kompetenz1,REL).

Beispiel: zugeordnet(Java,hat_Java_Kompetenz)

17. Regel:

FORALL Mitarbeiter1, Sprache1, Projekt1, REL

Mitarbeiter1[REL->Erfahrener]<-

Mitarbeiter1:Mitarbeiter[hat_gearbeitet_in->Projekt1] AND

Projekt1:Projekt[hat_Sprache -> Sprache1] AND

NOT EQUAL (Sprache1, deutsch) AND

zugeordnet(Sprache1,REL).

Beispiel: zugeordnet(daenisch,hat_daenisch_Kompetenz)

18. Regel:

FORALL Mitarbeiter1, Sprache1, Publikation1, REL

Mitarbeiter1[REL->Erfahrener]<-

Mitarbeiter1:Mitarbeiter[hat_Publikation->Publikation1] AND

Publikation1:Publikation[hat_Sprache->Sprache1] AND

NOT EQUAL (Sprache1,deutsch) AND

zugeordnet(Sprache1,REL).

Beispiel: zugeordnet(englisch,hat_Englisch_Kompetenz)

19. Regel:

FORALL Mitarbeiter1, Publikation1, Thema1, REL

Mitarbeiter1[REL->Erfahrener]<-

Publikation1: Publikation [hat_Thema->>Thema1;

hat_Autor->>Mitarbeiter1] AND

zugeordnet(Thema1,REL).

Beispiel: zugeordnet (SQL,hat_SQL_Kompetenz)

20. Regel:

FORALL Mitarbeiter1, Vortrag1, Thema1, REL

Person1[REL->Erfahrener]<-

Vortrag1:Vortrag [hat_Thema->>Thema1;

hat_Referenten ->> Person1] AND

zugeordnet(Thema1,REL).

Beispiel: zugeordnet (Wissensmanagement,hat_WM_Kompetenz)

3.3 Beispiele in Loom

In Loom können Regeln, wie in Abschnitt 2.2.4 vorgestellt, ausgedrückt werden. Die in Kapitel 3.1 vorgestellten, beispielhaften Regeln übersteigen allerdings die Modellierungsmächtigkeit von Loom. Es sind nur einfache Regeln möglich, die die Kategorisierung von Loom verwenden. Konzepte können überprüft werden, weitere Regeln scheinen nicht möglich zu sein. Diese Regeln können überprüfen, ob eine Hierarchie, Disjunktheit oder Äquivalenz zwischen Konzepten vorliegt. Auch das Schließen über unvollständiges Wissen ist möglich. Ein Beispiel wäre, dass aus den disjunkten Konzepten Frau und Mann und der Regel, dass nur Frauen schwanger werden können, geschlossen wird, dass eine schwangere Person eine Frau ist.

Darstellung der disjunkten Konzepte in Loom-Syntax:

```
(disjoint Male Female)
(implies Pregnant Female)
```

3.4 Beispiele in PowerLoom

PowerLoom bietet die Möglichkeit, Regeln darzustellen. Im Folgenden werden hier nur die ersten fünf Regeln zu obigen Beispielen vorgestellt:

1. Regel:

```
¦= (defrule KOMPETENZERWERB
  (forall ((?x MITARBEITER) (?y SCHULUNG) (?z KOMPETENZ)
    (<= (hat_Gegenstandsbereich ?y ?z)
    (and (hat_Schulungsniveau ?y professionell)
    (and (hat_Teilnehmer ?y ?x))))
      (hat_Kompetenz ?x ?z)
      (hat_Kompetenzauspraegung ?x Anfaenger)))
```

2. Regel:

```
¦= (defrule CHEF
  (forall ((?x MITARBEITER) (?y MITARBEITER) (?z MITARBEITER)
    (<= (hat_Chef ?x ?y)
    (and (hat_Chef ?y ?z)))
      (hat_Chef ?x ?z))))
```

3. Regel:

```
¦= (defrule FUEHRUNGSFAEHIGKEIT
  (forall ((?x MITARBEITER) (?p1 PROJEKT1) (?p2 PROJEKT2)
    (<= (hat_gearbeitet_in_Projekt ?x ?p1)
        (and (hat_gearbeitet_in_Projekt ?x ?p2)
        (and (hat_Projektmanager ?p1 ?x)
        (and (hat_Projektmanager (p2 ?x)
        (and (bewertung ?p1 erfolgreich)
        (and (bewertung ?p2 erfolgreich)))))))
                (hat_Kompetenz ?x Fuehrungsfaehigkeit)
                (and (hat_Kompetenzauspraegung ?x Experte)))))
```

Die Ungleichheit von zwei Projekten wird schon durch die Definition der Konzepte in PowerLoom festgelegt, deswegen muss sie hier nicht explizit ausgedrückt werden.

4. Regel:

```
¦= (defrule TEAMFAEHIGKEIT
  (forall (?x MITARBEITER) (?t1 TEAM) (?t2 TEAM)
    (<= (hat_gearbeitet_in_Team ?x ?t1)
        (and(hat_gearbeitet_in_Team ?x ?t2)
        (and ( (bewertung ?t1 erfolgreich)
        (and(bewertung ?t2 erfolgreich)))))
                (hat_Kompetenz ?x Teamfaehigkeit)
                (and (hat_Kompetenzauspraegung ?x Erfahrener)))))
```

Die Ungleichheit von zwei Teams wird schon durch die Definition der Konzepte in PowerLoom festgelegt, deswegen muss sie hier nicht explizit ausgedrückt werden.

5. Regel:

```
¦= (defrule WEISUNGSBEFUGNIS1
  (forall (?x MITARBEITER) (?y MITARBEITER)
    (<= (hat_Chef ?x ?y))
      (hat_Weisungsbefugnis_ueber ?y ?x)))
¦= (defrule WEISUNGSBEFUGNIS2
  (forall (?y MITARBEITER) (?x MITARBEITER)
    (<=(hat_Weisungsbefugnis_ueber ?y ?x))
      (ist_Chef ?y ?x)))
```

Die in diesem Fall vorliegende Symmetrie kann in PowerLoom nicht in einer Regel dargestellt werden, deswegen werden hier zwei Regeln modelliert.

3.5 Beispiele in RDF(S) und DAML + OIL

RDF(S) ermöglicht nicht die Darstellung von Inferenzregeln. Es gibt jedoch einfache Inferenzmechanismen, wie im Abschnitt 2.3.2 angesprochen. Die Ergänzungen und Überlegungen, um Inferenzen in RDF(S) zu ermöglichen, gehen meist in die Richtung, ein eigenes Schema zu entwickeln oder F-Logic-Ausdrücke in RDF als Queries[75)] darzustellen. Hierdurch wird aber der bestehende Standard erweitert und die Austauschbarkeit erschwert, da proprietäre[76)] Teile eingefügt werden. Nach den Standards für DAML+OIL und RDF(S) ist die Verwendung von z.B. F-Logic nicht vorgesehen, sodass eine Erweiterung des Standards durchgeführt wird, die nicht allgemein anerkannt ist.

DAML + OIL enthält durch die Erweiterung von RDF(S) alle Inferenzmechanismen von RDF(S). Zusätzlich gibt es noch die Möglichkeit, transitive, inverse, disjunkte und eindeutige Eigenschaften zu definieren[77)]. Zurzeit ist in DAML + OIL aber keine Reifikation möglich.

---

75) Eine Query ist im Datenbankbereich eine Abfrage.

76) Als proprietär wird etwas bezeichnet, das nicht standardisiert ist.

77) Vgl. o.V. (Language Feature Comparison), S. 2.

Als Tool steht OilEd[78)] zur Verfügung, das aber nur die Sprache OIL verwendet. Die Fortschritte, die durch die Verbindung von DAML+OIL gemacht wurden, können somit nicht verwendet werden. Wegen der fehlenden Möglichkeit zur Darstellung von Regeln können die Beispiele in DAML + OIL nicht dargestellt werden.

78) Weitere Informationen: o.V. (OilEd).

4 Kriterien zur Evaluation von Sprachen für Inferenzregeln

4.1 Kriterien der klassischen Logik

Die Kriterien der klassischen Logik sind wichtig für die Güte des aus der Wissensbasis generierten „neuen“ Wissens. Kriterien, die hier von Relevanz sind, sind „soundness“ und „completeness“[79)] sowie Monotonie[)] und Ausdrucksmächtigkeit. „Soundness“ bedeutet, dass durch die Inferenzmechanismen aus gültigem Wissen nur gültiges Wissen generiert wird. „Completeness“ bedeutet, dass es für jede semantisch gültige Formel mindestens eine syntaktische Inferenz gibt, mit der sich die Gültigkeit der Formel herleiten lässt. Monotonie heißt, dass Formeln, deren Gültigkeit mittels zulässiger Inferenzen hergleitet wurde, ihre Gültigkeit nicht verlieren können. Durch neu hinzugewonnenes Wissen kann sich die Gültigkeit von Formeln also niemals ändern. Dabei ist allerdings zu überlegen, ob die Nicht-Monotonie ein Gütekriterium oder eher nicht sinnvoll ist. Die Unbeständigkeit der Ergebnisse kann auch zum Problem werden, wenn man konstante Ergebnisse benötigt. Die Ausdrucksmächtigkeit begrenzt die Menge denkmöglicher Sachverhalte, die sich mit den Ausdrucksmitteln einer Sprache darstellen lassen. Da meist ganze Unternehmen abgebildet werden sollen, muss die Ausdrucksmächtigkeit einer Sprache dies ermöglichen.

4.2 Kriterien der Betriebswirtschaftslehre

Neben den Kriterien der klassischen Logik muss auch eine Betrachtung von betriebswirtschaftlichen Kriterien erfolgen, da diese Kriterien für den späteren Einsatz von Inferenzregeln und Tools zur Implementierung der Regeln in einem Unternehmen ausschlaggebend sind. Die Kriterien werden in Tabelle 2 mit ihren Erläuterungen angegeben.

79) Vgl. Flores-Mendez (Reason Logically), S. 2-3.

Kriterium:	**Erläuterung:**
Integrationsfähigkeit	Integration der Explizierung und der Generierung von neuem impliziten Wissen in die Abläufe des Unternehmens
Effizienz	Verhältnis von Ressourceneinsatz zum erzielbaren Nutzen
Schutz der Rechte Dritter	Bei der Akquisition von Wissen aus Dokumenten ist darauf zu achten, dass keine Rechte Dritter verletzt werden. So ist z.B. die Akquisition von Wissen aus PDF-Dokumenten rechtlich fraglich, da das PDF-Format von Adobe rechtlich geschützt wurde.
Schutz der Daten	Da in einer Wissensbasis Kompetenzen einzelner Mitarbeiter erfasst werden, ist der Datenschutz zu beachten. Die detaillierten Daten von Mitarbeitern unterliegen diesem Schutz und dürfen nur vertraulich behandelt werden. Dazu ist ein Berechtigungskonzept notwendig, dass die einem Nutzer zur Verfügung stehenden Daten einschränkt.
Nachvollziehbarkeit	Der Nutzer muss in der Lage sein, die Inferenzen des Systems nachzuvollziehen. Dieses Kriterium wird durch eine einfache Syntax einer Sprache unterstützt.
Natürlichkeit bei der Modellierung	Orientierung der Modellierung am menschlichen Denken.
Leichte Bedienbarkeit	Die Benutzung eines Tools sollte leicht möglich sein.

Tabelle 2: Kriterienkatalog der Betriebswirtschaftslehre
(Quelle: selbst erstellt)

5 Evaluation

5.1 Evaluation von F-Logic

Die Inferenzmaschine von F-Logic ist „sound“ und „complete“[80], es wird nur gültiges Wissen inferenziert und alle hergeleiteten Formeln sind gültig. Daher sind die zwei wichtigsten Kriterien der klassischen Logik erfüllt. Auch ist F-Logic monoton. Die Ausdrucksmächtigkeit von F-Logic ist hoch, da die Sprache durch die Verwendung der Frame Logic viele Möglichkeiten zur Darstellung bietet.

Die betriebswirtschaftlichen Kriterien sind nur schwer einzuschätzen, da zurzeit nur das Tool OntoEdit als Wissenseditor und das Tool OntoBroker[81] als Inferenzmaschine zur Verfügung stehen. Eigentlich verwendet OntoEdit OXML[82] als interne Repräsentationssprache. Da der OntoBroker aber nur F-Logic und Prädikatenlogik verarbeiten kann und Inferenzregeln in OXML in F-Logic übersetzt werden, erfolgt hier nur die Betrachtung von F-Logic. Die Integrationsfähigkeit ist zurzeit nur sehr eingeschränkt möglich, da durch die Komplexität der Syntax von F-Logic zunächst ein erhöhter Aufwand für das Einarbeiten in die Sprache notwendig ist, der zeit- und somit auch kostenaufwendig ist. Jedoch wird der Nutzer durch die grafische Oberfläche von OntoEdit unterstützt. Da F-Logic noch nicht vollständig in OntoEdit implementiert ist, ist das Tool noch nicht für den Endanwender geeignet. Es fehl die Darstellbarkeit von n-stelligen Relationen mit $n \geq 3$. Außer den Regeln erfolgt die interne Darstellung in OXML, teilweise sind die Sprachelemente von OXML nicht in F-Logic konvertierbar. Der erzielbare Nutzen ist nicht bezifferbar. Die Effektivität der Ergebnisse hängt davon ab, wie präzise die Anfrage gestellt wurde. Wenn Inferenzregeln angewendet werden, sind die natürlichsprachlichen Dokumente meist schon in eine formale Sprache umgewandelt, sodass das Problem des Schutzes der Rechte Dritter bereits bei der Wissensakquisition beachtet werden muss. Der Datenschutz muss durch Zugriffsbeschränkungen für die Benutzer realisiert werden, d.h., dass unterschiedliche Benutzer verschiedene Rechte besitzen, die Daten der Wissensbasis einzusehen und Inferenzregeln auszuführen. Dies könnte aber dazu führen, dass die Ergebnisse von Inferenzregelanwendungen nicht vollständig sind. Beispielhaft könnte ein Benutzer die Aufgabe haben, alle Mitarbeiter zu suchen, die ein

80) Vgl. Corcho / Gómez-Pérez (Evaluating Knowledge), S. 8.

81) Weitere Informationen zu OntoEdit und OntoBroker: o.V. (Ontoprise).

82) Vgl. für weitere Informationen zu OXML: Erdmann (OXML 2.0).

Gehalt über einem bestimmten Betrag haben, und deren Gehalt um 3% zu erhöhen. Der Benutzer hat, aufgrund von Einschränkungen seiner Zugriffsrechte, nur das Recht, die Daten bestimmter Abteilungen des Unternehmens zu verwenden. Dadurch würden die Mitarbeiter in den nicht zugreifbaren Abteilungen keine Lohnerhöhung erhalten. Eine andere Möglichkeit wäre die Anonymisierung der Daten für bestimmte Nutzer, wodurch zwar eine Inferenz über die gesamte Wissensbasis möglich wäre, aber die Daten der einzelnen Mitarbeiter geschützt würden. Dies bedeutet aber nach der Inferenzregelanwendung zusätzlichen Aufwand, da der Mitarbeiter sich an einen Mitarbeiter mit mehr Rechten wenden muss, der ihm die Ergebnisse unter Umständen zugänglich macht.

Mit F-Logic ist es dem Nutzer nach der Einarbeitung in die Syntax und die Semantik leicht möglich, die Schlussfolgerungen nachzuvollziehen. Durch die erfüllten Kriterien der klassischen Logik ist auch die semantische und syntaktische Gültigkeit des generierten Wissens gewährleistet. Die Aktualität des Wissens in der Wissensbasis ist abhängig von der Update-Häufigkeit des Datenbestandes. Die derzeit vorhandenen Tools, die zur Erstellung und Wartung der Wissensbasis und zur Ausführung von Inferenzregeln genutzt werden, sind von ihrer Bedienbarkeit her komplex (diese Einschätzung beruht auf einem subjektiven Urteil). Schon die Installation eines solchen Tools überfordert die meisten Anwender. Von der Bedienung her entsprechen die Programme auch nicht dem gewohnten „Look and Feel“ der Windows-Oberfläche, sodass wenige Anreize bestehen, eine häufige Aktualisierung durchzuführen. Die Bedienbarkeit ist aber ein entscheidender Faktor für den Erfolg von Wissensbasen und Inferenzregeln in einem Unternehmen. Ist das eingesetzte Softwaretool leicht zu installieren und anzuwenden, so wird ein Nutzer motiviert, sich mit dem System zu beschäftigen.

Die Natürlichkeit bei der Modellierung ist in F-Logic gegeben, allerdings muss sich der Nutzer erst stark auf die Syntax von F-Logic einstellen.

5.2 Evaluation von Loom

Der in Loom verwendete „Classifier“ generiert nach MACGREGOR auf Anfragen Antworten, die „sound“, aber nicht „complete“ sind[83)]. Außerdem ist er auf Subklassen-, Äquivalenz- und Disjunktheits-Anfragen beschränkt, sodass nicht alle für die Inferenzierung von Wissen notwendigen Anfragen gestellt werden können. Die Kriterien der klassischen Logik werden vollständig erfüllt. Die Kriterien der Betriebswirtschaftslehre sind nur schwer zu beurteilen, weil hierzu die Verwendung eines Tools von Vorteil gewesen wäre.

Die Sprachen von Loom sind komplex, und durch die Unterscheidung in zwei Sprachen besteht die Notwendigkeit, beide Sprachen zu beherrschen, was den Aufwand für das Erlernen deutlich erhöht. Die Natürlichkeit der Modellierung ist gegeben, da Loom durch die Verwendung von einfachen Beschreibungen eine Lesbarkeit ermöglicht. Die Bedienbarkeit war nicht beurteilbar, da sich das Tool nur unter UNIX-Betriebssystemen installieren lässt, diese Betriebssysteme standen aber nicht zur Verfügung. Über alle anderen Kriterien sind keine Aussagen möglich, weil keine Informationen zur Verfügung standen.

5.3 Evaluation von PowerLoom

PowerLoom liegt derzeit in der zweiten Version (1.1) vor, allerdings sind über diese Sprache und das zugehörige Tool nur sehr wenige Informationen verfügbar. Eine Evaluation wird so erschwert. Ob die Kriterien der klassischen Logik erfüllt sind, kann nicht festgestellt werden. Die betriebswirtschaftlichen Kriterien können ebenfalls nur schwer eingeschätzt werden, weil das derzeitige Tool nur ein Kommandozeilenprogramm ist. Eine Integration in bestehende Abläufe wird dadurch erschwert, da die Mitarbeiter heute an grafische Oberflächen gewöhnt sind. Die Effizienz und Effektivität scheinen gut zu sein, da die Sprache sehr klar strukturiert ist. Auch die Möglichkeiten zur Darstellung von Inferenzregeln durch die Sprache sind ausgeprägt. Der Schutz der Rechte Dritter kann nicht überprüft werden, da der Nutzer des Tools das Wissen in das System selbst eingeben muss, andere Dokumente können nicht durch eine Akquisitionstechnik eingefügt werden. Die Korrektheit der Daten wird zum Beispiel durch die Mög-

83) Vgl. MacGregor (Deductive Inference), S. 3.

lichkeit zur Definition von Wertebereichen unterstützt. Die Bedienbarkeit ist durch das Kommandozeilentool nur sehr eingeschränkt gegeben, da der heutige Benutzer an eine grafische Oberfläche gewöhnt ist, die er mit der Maus bedienen kann. Bei diesem Tool muss er aber „kryptische“ Befehle eingeben, um das Programm zu bedienen. Über alle anderen Kriterien können keine Aussagen gemacht werden, weil weitere Informationen fehlen.

5.4 Evaluation von RDF(S) und DAML + OIL

RDF(S) und DAML + OIL sollen zwar einfache Inferenzmechanismen besitzen[84)]. Doch sowohl die Publikationen zu diesem Thema als auch die Softwaretools bleiben diese Möglichkeiten in der praktischen Anwendung schuldig. Daher war keine Möglichkeit vorhanden, die Inferenzregeln darzustellen. Triple erscheint als eine Erweiterung von RDF(S) und DAML + OIL, die eine Implementierung von Inferenzregeln ermöglicht. Bei genauerer Betrachtung stellte sich allerdings heraus, dass die Syntax für die Implementierung der Inferenzregeln der von F-Logic entspricht[85)]. In den Regeln werden zwar die Tags von RDF(S) und DAML + OIL als Variablen verwendet, dadurch erfolgt allerdings keine grundlegende Änderung der Syntax. Daher können die zu F-Logic gemachten Aussagen auch auf Triple übertragen werden. Zu den betriebswirtschaftlichen Kriterien sind keine Aussagen möglich, da die zur Verfügung stehenden Tools keine Darstellung und somit auch Anwendung von Inferenzregeln ermöglichen. Zusätzlich wird durch die Tools die Erstellung von Dokumenten in RDF oder DAML + OIL nur sehr einfach unterstützt, es erfolgt anscheinend nur eine Überprüfung der Korrektheit der Syntax.

84) Vgl. Corcho / Gómez-Pérez (Evaluating Knowledge), S. 8.

85) Vgl. Sintek / Decker (TRIPLE), S. 3-4.

5.5 Zusammenfassende Darstellung der Ergebnisse

Im Folgenden wird eine tabellarische Zusammenfassung der Ergebnisse vorgenommen. Hierzu wird ein dreiwertiges Bewertungsschema verwendet:

- \+ Kriterium erfüllt
- \- Kriterium nicht erfüllt
- o Ausprägung des Kriteriums nicht beurteilbar

Kriterium vs. Sprache	F-Logic	Loom	PowerLoom	RDF(S) / DAML+OIL
Soundness	+	+	o	o
Completeness	+	-	o	o
Monotonie	+	o	o	o
Ausdrucksmächtigkeit	+	-	o	o
Integrationsfähigkeit	o	o	o	o
Effizienz	+	o	+	o
Effektivität	+	o	+	o
Schutz der Rechte Dritter	-	o	o	o
Schutz der Daten	-	o	o	o
Nachvollziehbarkeit	-	o	-	o
Natürlichkeit bei der Modellierung	-	+	o	o
leichte Bedienbarkeit	-	o	-	o

Tabelle 3: Tabellarische Darstellung der Ergebnisse
(Quelle: selbst erstellt)

Zwischen den Kriterien Effizienz / Effektivität und Nachvollziehbarkeit scheint eine Wechselwirkung zu bestehen. Eine Beurteilung des einen Kriteriums als erfüllt hat eine Bewertung des anderen Kriteriums als nicht erfüllt zur Folge. Allerdings konnte kein näherer Zusammenhang festgestellt werden.

6 Fazit und zukünftige Entwicklungen

Bei der Akquisition von Wissen und der damit verbundenen Konvertierung in ein formales Repräsentationsformat muss eine Sprache ausgewählt werden. Es gibt eine große Menge von Sprachen, die zur Repräsentation von Wissen geeignet sind. Möchte man Inferenzen durchführen, so verkleinert sich die Zahl der zur Verfügung stehenden Sprachen. Möchte man die Inferenzregeln auch in der Sprache implementieren, so gibt es nur drei Sprachen, die dies ermöglichen: F-Logic, Loom und PowerLoom.

Loom ermöglicht dies allerdings nur in eingeschränktem Umfang, da nur einfache Regeln möglich sind, wie in Kapitel 3.3 erläutert. PowerLoom befindet sich noch in einem frühen Entwicklungsstadium und wird nur durch ein Kommandozeilen-Tool unterstützt. Zu PowerLoom gibt es nur sehr wenige Publikationen, sodass über die derzeitigen Einsatzmöglichkeiten keine Aussage möglich ist.

F-Logic liegt in einer Version vor, die die meisten Kriterien erfüllt. Zusätzlich ist sie die einzige Sprache, die in Unternehmen angewandt werden kann, da sie durch kommerziell verfügbare Tools unterstützt wird. Allerdings ist die Einarbeitung in die Syntax mit Aufwand verbunden, den Unternehmen zunächst scheuen werden. Durch die Tools OntoEdit und OntoBroker ist zwar eine Vereinfachung der Bedienung gegeben, aber durch die nicht vollständige Implementierung aller Möglichkeiten von F-Logic ist eine zusätzliche Bearbeitung der erstellten Ontologien und Regeln „von Hand“ notwendig. Dies setzt die Kenntnis von F-Logic voraus. So sind die von OntoEdit exportierten Ontologien im Format F-Logic nicht sofort weiter einsetzbar, weil zunächst OntoEdit-spezifische Erweiterungen entfernt werden müssen.

Alle anderen Sprachen sind ungeeignet, mit ihrer eigenen Syntax Inferenzregeln darzustellen. Viele Verbesserungsansätze, wie z.B. Triple, gehen dahin, F-Logic-Ausdrücke in die Sprachen aufzunehmen. Da aber keine Tools zur Verfügung stehen, die den Einbezug von F-Logic-Ausdrücken unterstützen, und diese Erweiterungen bestehende Standards ergänzen und somit proprietäre Lösungen sind, wurde eine weitere Betrachtung nicht durchgeführt.

Im Rahmen der Forschung im Bereich des Semantic Web[86)] wird zurzeit die OWL (Web Ontology Language) entwickelt[87)]. Sie basiert auf DAML+OIL und RDF(S). Da

diese Sprache aber erst in der Entstehung ist, sind noch keine Aussagen über die Inferenzierungs- und Darstellungsmöglichkeiten möglich. Ein erster Working Draft[88)] mit einer abstrakten Syntax-Beschreibung und der dahinter stehenden Semantik steht allerdings schon jetzt zur Verfügung. Es besteht somit die Möglichkeit, dass es in näherer Zukunft eine Alternative zu F-Logic geben wird, da OWL alle Schwächen ihrer Vorgänger in Stärken umwandeln soll[89)].

86) Weitere Informationen zum Thema Semantic Web: Decker / Sintek (SemanticWeb).

87) Weitere Informationen: o.V. (WebOnt) und o.V. (OWL).

88) Die abstrakte Syntax-Notation ist zu finden unter: o.V. (OWL).

89) Allgemeine Beschreibung der Ziele von OWL: o.V. (WebOnt).

Literaturverzeichnis

Bibel (Deduktion):

Bibel, W.: Deduktion. München, Wien 1992.

Bibel (Wissensrepräsentation):

Bibel, W.; Hölldobler, St.; Schaub, T.: Wissensrepräsentation und Inferenz: Eine grundlegende Einführung. Braunschweig, Wiesbaden 1993.

Bray (RDF and Metadata):

Bray, T.: RDF and Metadata. [Im Internet unter der URL: http://www.xml.com/pub/a/98/06/rdf.html, Datum des Zugriffs: 12.03.2004.]

Bray (What is RDF):

Bray, T.: What is RDF?. [Im Internet unter der URL: http://www.xml.com/pub/a/2001/01/24/rdf.html, Datum des Zugriffs: 12.03.2004.]

Chaudhri / Farquhar / Fikes / Karp / Rice (Open Knowledge):

Chaudhri, V.; Farquhar, A.; Fikes, R.; Karp, P.; Rice, J.: Open Knowledge Base Connectivity 2.0.3. [Im Internet unter der URL: http://www.ai.sri.com/~okbc/spec.html, Datum des Zugriffs: 12.03.2004.]

Connolly (Naming):

Connolly, D.: Naming and Addressing: URIs, URLs, ... [Im Internet unter der URL: http://www.w3.org/Addressing, Datum des Zugriffs: 12.03.2004.]

Corcho / Gómez-Pérez (Evaluating Knowledge):

Corcho, O.; Gómez-Pérez, A.: Evaluating Knowledge Representation and Reasoning Capabilities of Ontology Specification Languages. In: o.V. (Hrsg.): ECAI'00 Workshop on Application of Ontologies and Problem Solving Methods, Berlin 2000. [Im Internet unter der URL: http://delicias.dia.fi.upm.es/articulos/ocorcho/corchoetal_ecai00_ws .pdf, Datum des Zugriffs: 12.03.2004.] (eigene Paginierung).

Decker (Domain-Specific Declarative):

Decker, St.: On Domain-Specific Declarative Knowledge Representation Languages. In: Borgida, A.; Chaudri, V.; Staudt, M. (Hrsg.): Proceedings of the 5th Knowledge Representation meets Databases Workshop (KRDB98), Seattle 1998, S. 9.1-9.7. [Im Internet unter der URL: http://citeseer.nj.nec.com/article/decker98domainspecific.html, Datum des Zugriffs: 12.03.2004.]

Decker / Brickley / Saarela / Angele (Inference Service):

Decker, St.; Brickley, D.; Saarela, J.; Angele, J.: A Query and Inference Service for RDF. In: o.V. (Hrsg.): QL'98 - The Query Languages Workshop, W3C Workshop, 1998. [Im Internet unter der URL: http://www.w3.org/TandS/QL/QL98/pp/query service.html, Datum des Zugriffs: 12.03.2004] (eigene Paginierung).

Decker / Sintek (SemanticWeb):

Decker, St.; Sintek, M: SemanticWeb.org. [Im Internet unter der URL: http://www. semanticweb.org, Datum des Zugriffs: 12.03.2004.]

Erdmann (OXML 2.0):

Erdmann, M.: OXML 2.0 – Reference manual for users and developers of OntoEdit's XML-based Ontology Representation language. [Im Internet unter der URL: http://www.ontoprise.de/download/OXML2_0_manual.zip, Datum des Zugriffs: 12.03.2004.]

Erdmann / Studer (XML Documents):

Erdmann, M.; Studer, R.: Ontologies as Conceptual Models for XML Documents. In: o.V. (Hrsg.): Proceedings of the 12th Workshop for Knowledge Acquisition, Modeling and management (KAW'99), Banff (Canada) 1999. [Im Internet unter der URL: http://sern.ucalgary.ca/KSI/KAW/KAW99/papers/Erdmann1/erdmann.pdf, Datum des Zugriffs: 12.03.2004.]

Fensel (Sinn und Unsinn):

Fensel, D.: Sinn und Unsinn formaler Spezifikationssprachen für wissensbasierte Systeme. In: Künstliche Intelligenz (KI), Nr. 4, 1994, S. 26-34. [Im Internet unter der URL: http://www.ubka.uni-karlsruhe.de/cgi-bin/psview?document=/1994/wiwi/9, Datum des Zugriffs: 12.03.2004.]

Flores-Mendez (Reason Logically):

Flores-Mendez, R.: Agents that Reason Logically. In: Russel, St.; Norvig, P. (Hrsg.): Artificial Intelligence: A Modern Approach, 1995, S. 151-184. [Im Internet unter der URL: http://sern.ucalgary.ca/courses/CPSC/533/W99/reasoning/, Datum des Zugriffs: 12.03.2004.] (eigene Paginierung).

Gil / Ratnakar (Markup Languages):

Gil, Y.; Ratnakar, V.: A Comparison of (Semantic) Markup Languages. In: o.V. (Hrsg.): Proceedings of the 15th International FLAIRS Conference, Special Track on Semantic Web, Pensacola (USA) 2002. [Im Internet unter der URL: http://trellis.semanticweb.org/expect/web/semanticweb/paper.pdf, Datum des Zugriffs: 12.03.2004.] (eigene Paginierung).

Gruber (Ontology Specifications):

Gruber, Th.: A Translation Approach to Portable Ontology Specifications. Knowledge Systems Laboratory Technical Report KSL 92–71, Computer Science Department, Stanford University, Stanford 1992. [Im Internet unter der URL: http://ksl-web.stanford .edu/KSL_Abstracts/KSL-92-71.html, Datum des Zugriffs: 12.03.2004.]

Guha / Lassila / Miller / Brickley (Enabling Inferencing):

Guha, R.; Lassila, O.; Miller, E.; Brickley, D.: Enabling Inferencing. [Im Internet unter der URL: http://www.w3.org/TandS/QL/QL98/pp/enabling.html, Datum des Zugriffs: 12.03.2004.] (eigene Paginierung).

Karp / Chaudhri / Thomere (XOL):

Karp, P.; Chaudhri, V.; Thomere, J.: XOL: An XML-Based Ontology Exchange Language. [Im Internet unter der URL: http://www.ai.sri.com/pkarp/xol/, Datum des Zugriffs: 12.03.2004.] (eigene Paginierung).

Karvounarakis / Christophides / Alexaki / Plexousakis / Scholl (Declarative Query Language):

Karvounarakis, G.; Christophides, V.; Alexaki, S.; Plexousakis, D.; Scholl, M.: RQL: A Declarative Query Language for RDF. In: WWW02 Proceedings, Honolulu 2002. [Im Internet unter der URL: http://139.91.183.30:9090/RDF/publications/www2002/ www2002.pdf, Datum des Zugriffs: 12.03.2004.] (eigene Paginierung).

Kifer / Lausen (F-Logic):

Kifer, M.; Lausen, G.: F-Logic: A Higher-Order Language for Reasoning about Objects, Inheritance and Scheme. Sigmod Record, Vol. 18 (1990), No. 6, pp. 134-146. [Im Internet unter der URL: http://citeseer.ist.psu.edu/kifer90flogic.html, Datum des Zugriffs: 12.03.2004.]

MacGregor (Deductive Inference):

MacGregor, R.: Using a Description Classifier to Enhance Deductive Inference. In: o.V. (Hrsg.): Proceedings Seventh IEEE Conference on AI Applications, 1991, S. 141-147. [Im Internet unter der URL: http://www.isi.edu/isd/LOOM/papers/macgregor/florida.pdf, Datum des Zugriffs: 12.03.2004.]

MacGregor (Retrospective on Loom):

MacGregor, R.: Retrospective on Loom. [Im Internet unter der URL: http://www.isi.edu/isd/LOOM/papers/macgregor/Loom_Retrospective.html, Datum des Zugriffs: 12.03.2004.]

MacGregor / Chalupsky / Melz (PowerLoom Manual):

MacGregor, R.; Chalupksy, H.; Melz, E.: PowerLoom Manual. [Im Internet unter der URL: http://www.isi.edu/isd/LOOM/PowerLoom/documentation/documentation.html, Datum des Zugriffs: 12.03.2004.]

McBride (RDF Schema):

McBride, B.: RDF Vocabulary Description Language 1.0: RDF Schema. [Im Internet unter der URL: http://www.w3.org/TR/rdf-schema/, Datum des Zugriffs: 12.03.2004.]

Miller (Squish query language):

Miller, L.: RDF Squish query language and Java implementation. [Im Internet unter der URL: http://ilrt.org/discovery/2001/02/squish, Datum des Zugriffs: 12.03.2004.] (eigene Paginierung).

Miller / Swick / Brickley (RDF):

Miller, E.; Swick, R.; Brickley, D.: Ressource Description Framework (RDF). [Im Internet unter der URL: http://www.w3.org/RDF/, Datum des Zugriffs: 12.03.2004.]

Nonaka/Takeuchi (Wissen):

Nonaka, I.; Takeuchi, H.: Die Organisation des Wissens. Frankfurt am Main 1997.

o.V. (DAML):

o.V.: DAML Query Language (August 2002). [Im Internet unter der URL: http://www.daml.org/2002/08/dql/, Datum des Zugriffs: 12.03.2004.]

o.V. (Daml + OIL):

o.V.: Daml + OIL Reference. [Im Internet unter der URL: http://www.w3.org/TR/daml+oil-reference/, Datum des Zugriffs: 12.03.2004.]

o.V. (Darpa Agent):

o.V.: The Darpa Agent Markup Language Homepage. [Im Internet unter der URL: http://www.daml.org, Datum des Zugriffs: 12.03.2004.]

o.V. (Knowledge Interchange Format):

o.V.: Knowledge Interchange Format – draft proposed American National Standard (dpANS). [Im Internet unter der URL: http://logic.stanford.edu/kif/dpans.html, Datum des Zugriffs: 12.03.2004.] (eigene Paginierung).

o.V. (Language Feature Comparison):

o.V.: Language Feature Comparison. [Im Internet unter der URL: http://www.daml.org/language/features.html, Datum des Zugriffs: 12.03.2004.] (eigene Paginierung).

o.V. (LOOM):

o.V.: LOOM Project Home page. [Im Internet unter der URL: http://www.isi.edu/isd/LOOM/LOOM-HOME.html, Datum des Zugriffs: 12.03.2004.]

o.V. (Nexus):

o.V.: Nexus Query Language. [Im Internet unter der URL: http://www.langdale.com.au/RDF/NexusQueryLanguage.pdf, Datum des Zugriffs: 12.03.2004.]

o.V. (OIL):

o.V.: Welcome to OIL. [Im Internet unter der URL: http://www.ontoknowledge.org/oil/, Datum des Zugriffs: 12.03.2004.]

o.V. (OilEd):

o.V.: OilEd. [Im Internet unter der URL: http://oiled.man.ac.uk/, Datum des Zugriffs: 12.03.2004.]

o.V. (OKBC):

o.V.: Open Knowledge Base Connectivity Home Page. [Im Internet unter der URL: http://www.ai.sri.com/~okbc, Datum des Zugriffs: 12.03.2004.]

o.V. (Ontoprise):

o.V.: Ontoprise GmbH Homepage. [Im Internet unter der URL: http://www.ontoprise.de, Datum des Zugriffs: 12.03.2004.]

o.V. (OWL):

o.V.: OWL Web Ontology Language Abstract Syntax and Semantics. [Im Internet unter der URL: http://www.w3.org/TR/2002/WD-owl-semantics-20021108/, Datum des Zugriffs: 12.03.2004.]

o.V. (PowerLoom):

o.V.: PowerLoom Knowledge Representation System. [Im Internet unter der URL: http://www.isi.edu/isd/LOOM/PowerLoom/, Datum des Zugriffs: 12.03.2004.]

o.V. (RDQL):

o.V.: RDQL: RDF Data Query Language. [Im Internet unter der URL: http://www.hpl. hp.com/semweb/rdql.htm, Datum des Zugriffs: 12.03.2004.]

o.V. (STELLA):

o.V.: STELLA – Lisp-style Symbolic Programming with Delivery in Common-Lisp, C++ and Java. [Im Internet unter der URL: http:// www.isi.edu/isd/LOOM/stella/index. html, Datum des Zugriffs: 01.04.2004.]

o.V. (WebOnt):

o.V.: Web-Ontology (WebOnt) Working Group. [Im Internet unter der URL: http:// www.w3.org/2001/sw/WebOnt/, Datum des Zugriffs: 12.03.2004.]

o.V. (XML):

o.V.: Extensible Markup Language (XML). [Im Internet unter der URL: http:// www.w3.org/XML, Datum des Zugriffs: 12.03.2004.]

Ogbuji (Techniques for Knowledge Management):

Ogbuji, U.: Thinking XML: Basic XML and RDF Techniques for Knowledge Management, Part 6 – RDF Query using Versa. [Im Internet unter der URL: http://www-106.ibm.com/developerworks/xml/library/x-think10/index.html, Datum des Zugriffs: 12.03.2004.] (eigene Paginierung).

Ogbuji (Versa):

Ogbuji, U.: RDF Query using Versa. [Im Internet unter der URL: http://www-106.ibm.com/developerworks/xml/library/x-think10/index.html, Datum des Zugriffs: 12.03.2004.]

Ouellet / Ogbuji (Introduction to DAML):

Ouellet, R.; Ogbuji, U.: Introduction to DAML: Part I & Part II. [Im Internet unter der URL: http://www.xml.com/lpt/a/2002/01/30/daml1.html und daml2.html, Datum des Zugriffs: 12.03.2004.]

Rehäuser / Krcmar (Wissensmanagement):

Rehäuser, J.; Krcmar, H.: Wissensmanagement im Unternehmen. In: Schreyögg, G.; Conrad, P. (Hrsg.): Managementforschung 6 – Wissensmanagement. Berlin, New York 1996, S. 1-40.

Richter (Prinzipien der künstlichen Intelligenz):

Richter, M.: Prinzipien der Künstlichen Intelligenz – Wissensrepräsentation, Inferenz und Expertensysteme. 2. Aufl., Stuttgart 1992.

Sintek / Decker (TRIPLE):

Sintek, M.; Decker, St.: TRIPLE – An RDF Query, Inference, and Transformation Language. In: o.V. (Hrsg.): Proceedings DDLP'2001. [Im Internet unter der URL: http://triple.semanticweb.org/iswc2002/TripleReport.pdf, Datum des Zugriffs: 12. 03.2004.] (eigene Paginierung).

Specht (Wissensbasierte Systeme):

Specht, D.: Wissensbasierte Systeme im Produktionsbetrieb. München, Wien 1989.

Staab (Semantic Web):

Staab, St.: Semantic Web – Das Web der nächsten Generation. Karlsruher Transfer, Nr. 15, Karlsruhe 2001, S. 18-23. [Im Internet unter der URL: http://www.aifb.uni-karlsruhe.de/WBS/sst/Research/Publications/ka-transfer.pdf, Datum des Zugriffs: 12.03.2004.] (eigene Paginierung).

Staab / Erdmann / Maedche / Decker (Extensible Approach):

Staab, St.; Erdmann, M.; Maedche, A.; Decker, St.: An Extensible Approach for Modelling Ontologies in RDF(S). In: Grütter, R. (Hrsg.): Knowledge Media in Healthcare: Opportunities and Challenges, Hershey USA / London UK 2001, o.S. [Im Internet unter der URL: http://www.aifb.uni-karlsruhe.de/WBS/sst/Research/Publications/onto-rdfs.pdf, Datum des Zugriffs: 12.03.2004.]

Zelewski (Wissensmanagement mit Ontologien):

Zelewski, St.: Wissensmanagement mit Ontologien – eine einführende Darstellung, Arbeitsbericht Nr. 15, Institut für Produktion und Industrielles Informationsmanagement, Universität Essen, Essen 2002. [Im Internet unter der URL: http://www.pim.uni-essen.de/publikationen/zelewski/arbeitsberichte/essen/Arbeitsbericht_Wissensmanagement_mit_Ontologien.pdf, Datum des Zugriffs: 12.03.2004.]

Anhang

Im Folgenden wird eine erweiterte KOWIEN-Ontologie dargestellt, in die die vorgestellten Regeln integriert wurden:

// Konzepte ----------

// allgemein
Selbstkompetenz::Kompetenz.
--- Kompetenz::subjektives_Attribut.
--- subjektives_Attribut::Attribut.
--- Attribut::abstraktes_Denkobjekt.
Fachkompetenz::Kompetenz.
Methodenkompetenz::Kompetenz.
Sozialkompetenz::Kompetenz.

// Fachkompetenzen
Branchenkompetenz::Fachkompetenz.
Fremdsprachkompetenz::Fachkompetenz.
IT_Kompetenz::Fachkompetenz.
Juristische_Kompetenz::Fachkompetenz.
Produktkompetenz::Fachkompetenz.
Mandantenkompetenz::Fachkompetenz.

// Methodenkompetenzen
Analysemethoden_Kompetenz::Methodenkompetenz.
Unternehmensbewertungsmethoden_Kompetenz::Methodenkompetenz.
Wirtschaftspruefungsmethoden_Kompetenz::Methodenkompetenz.
Praesentationsmethoden_Kompetenz::Methodenkompetenz.
Kreativitaetsmethoden_Kompetenz::Methodenkompetenz.
Planungsmethoden_Kompetenz::Methodenkompetenz.
Investitionsmethoden_Kompetenz::Methodenkompetenz.

// IT-Kompetenzen
Programmiersprachen_Kompetenz::IT_Kompetenz.
Betriebssysteme_Kompetenz::IT_Kompetenz.
Anwendungsprogramme_Kompetenz::IT_Kompetenz.
Case_Tool_Kompetenz::IT_Kompetenz.
Datenbanken_Kompetenz::IT_Kompetenz.
Entwicklungstool_Kompetenz::IT_Kompetenz.
Konfigurationsmanagement_Kompetenz::IT_Kompetenz.
Middleware_Kompetenz::IT_Kompetenz.
Netzwerktechnologie_Kompetenz::IT_Kompetenz.
IT_Architektur_Kompetenz::IT_Kompetenz.
Auszeichnungssprachen_Kompetenz::IT_Kompetenz.

// Kreativititätsmethoden_Kompetenzen
Analytisch_systematische_Methoden_Kompetenz::Kreativitaetsmethoden_Kompetenz.
Intuitiv_kreative_Methoden_Kompetenz::Kreativitaetsmethoden_Kompetenz.

// Unternehmensbewertungsmethoden_Kompetenz
Gesamtbewertungsverfahren_Kompetenz::Unternehmensbewertungsmethoden_Kompetenz.
Einzelbewertungsverfahren_Kompetenz::Unternehmensbewertungsmethoden_Kompetenz.

// Objekte
abstraktes_Denkobjekt::Denkobjekt.
Denkobjekt::objektsprachliche_Entitaet.

objektsprachliche_Entitaet::Entitaet.
konkretes_Denkobjekt::Denkobjekt.
Quantitaet::abstraktes_Denkobjekt.
Aussage::abstraktes_Denkobjekt.
metasprachliche_Entitaet::Entitaet.
Zustand::situatives_Denkobjekt.
situatives_Denkobjekt::abstraktes_Denkobjekt.
Aktivitaet::situatives_Denkobjekt.
zeitartiges_Erfahrungsobjekt::Erfahrungsobjekt.
Erfahrungsobjekt::objektsprachliche_Entitaet.
handlungsunfaehiges_Denkobjekt::konkretes_Denkobjekt.
Akteur::konkretes_Denkobjekt.
Gebiet::handlungsunfaehiges_Denkobjekt.
Gegenstand::handlungsunfaehiges_Denkobjekt.
objektives_Attribut::Attribut.
Charaktereigenschaft::subjektives_Attribut.
Intensitaet::subjektives_Attribut.
raumartiges_Erfahrungsobjekt::Erfahrungsobjekt.
Punkt::raumartiges_Erfahrungsobjekt.
Strecke::raumartiges_Erfahrungsobjekt.
Flaeche::raumartiges_Erfahrungsobjekt.
Raum::raumartiges_Erfahrungsobjekt.
Kompetenzprofil::subjektives_Attribut.
Relation::metasprachliche_Entitaet.
Konzept::metasprachliche_Entitaet.
Regel::metasprachliche_Entitaet.
Definition::metasprachliche_Entitaet.
Sprache::objektives_Attribut.
Stelle::abstraktes_Denkobjekt.

// Intensitäten
ordinal_skalierte_Intensitaet::Intensitaet.
kardinal_skalierte_Intensitaet::Intensitaet.
Kompetenzauspraegung::ordinal_skalierte_Intensitaet.

// Betriebssystem-Kompetenzen
IBM_Betriebssystem_Kompetenz::Betriebssysteme_Kompetenz.
Unix_Betriebssystem_Kompetenz::Betriebssysteme_Kompetenz.
Windows_Betriebssystem_Kompetenz::Betriebssysteme_Kompetenz.

// Anwendungsprogramm-Kompetenzen
MS_Office_Kompetenz::Anwendungsprogramme_Kompetenz.
Helpline_Kompetenz::Anwendungsprogramme_Kompetenz.
infonea_Kompetenz::Anwendungsprogramme_Kompetenz.
SAP_Kompetenz::Anwendungsprogramme_Kompetenz.
MS_Visio_Kompetenz::Anwendungsprogramme_Kompetenz.

// MS-Office-Kompetenzen
MS_Excel_Kompetenz::MS_Office_Kompetenz.
MS_Word_Kompetenz::MS_Office_Kompetenz.
MS_Powerpoint_Kompetenz::MS_Office_Kompetenz.
MS_Access_Kompetenz::MS_Office_Kompetenz.

// Datenbanken-Kompetenzen
MS_Access_Kompetenz::Datenbanken_Kompetenz.

// Auszeichnungssprachen_Kompetenzen
HTML_Kompetenz::Auszeichnungssprachen_Kompetenz.
XML_Kompetenz::Auszeichnungssprachen_Kompetenz.
XHTML_Kompetenz::Auszeichnungssprachen_Kompetenz.

// Netzwerktechnologie_Kompetenz
Netzwerkprotokoll_Kompetenz::Netzwerktechnologie_Kompetenz.
Netztopologien_Kompetenz::Netzwerktechnologie_Kompetenz.
Netzwerkarchitekturen_Kompetenz::Netzwerktechnologie_Kompetenz.
Netzwerkdienste_Kompetenz::Netzwerktechnologie_Kompetenz.

// Akteure
Individualakteur::Akteur.
menschliches_Kollektiv::Kollektivakteur.
Kollektivakteur::Akteur.
Organisation::menschliches_Kollektiv.
machinelles_Kollektiv::Kollektivakteur.
Mitarbeiter::Individualakteur
Projektmanager::Individualakteur.

// Organisationstypen
Team::Organisation.
Unternehmensnetzwerk::Organisation.
virtuelles_Unternehmen::Unternehmensnetzwerk.
strategisches_Unternehmensnetzwerk::Unternehmensnetzwerk.
regionales_Unternehmensnetzwerk::Unternehmensnetzwerk.
Unternehmen::Organisation.
Abteilung::Organisation.

// Unternehmenstypen
Fischerei_und_Fischzucht::Unternehmen.
Bergbau::Unternehmen.
Kohlenbergbau::Bergbau.
Erzbergbau::Bergbau.
Verarbeitendes_Gewerbe::Unternehmen.
Textil_und_Bekleidungsgewerbe::Verarbeitendes_Gewerbe.
Spinnstoffaufbereitung_und_Spinnerei::Textil_und_Bekleidungsgewerbe.
Weberei::Textil_und_Bekleidungsgewerbe.
Textilveredlung::Textil_und_Bekleidungsgewerbe.
Herstellung_von_konfektionierten_Textilwaren_ohne_Bekleidung::
Textil_und_Bekleidungsgewerbe.
Bekleidungsgewerbe::Textil_und_Bekleidungsgewerbe.
Ledergewerbe::Verarbeitendes_Gewerbe.
Ledererzeugung::Ledergewerbe.
Lederverarbeitung::Ledergewerbe.
Herstellung_von_Schuhen::Ledergewerbe.
Holzgewerbe_ohne_Moebel::Verarbeitendes_Gewerbe.
Papier_Verlags_und_Druckgewerbe::Verarbeitendes_Gewerbe.
Energie_und_Wasserversorgung::Unternehmen.
Baugewerbe::Unternehmen.
Handel_Instandhaltung_u_Reperatur_v_Kraftfahrzeugen_u_Gebrauchsguetern::Unternehmen.
Gastgewerbe::Unternehmen.
Verkehr_und_Nachrichtenuebermittlung::Unternehmen.
Kredit_und_Versicherungsgewerbe::Unternehmen.
Grundstuecks_und_Wohnungswesen_Vermietung_bewegl_Sachen_::Unternehmen.
oeffentliche_Verwaltung_Verteidigung_Sozialversicherung::Unternehmen.
Erziehung_und_Unterricht::Unternehmen.
Gesundheits__Veterinaer_und_Sozialwesen::Unternehmen.
Kokerei_Mineraloelv_H_und_V_von_Spalt_und_Brutstoffen::Verarbeitendes_Gewerbe.
Chemische_Industrie::Verarbeitendes_Gewerbe.
Glasgewerbe_Keramik_V_von_Steinen_und_Erden::Verarbeitendes_Gewerbe.
Metallerzeugung_und_bearbeitung::
Herstellung_Erzeugung_u_Bearbeitung_v_Metallerzeugnissen.
Herstellung_Erzeugung_u_Bearbeitung_v_Metallerzeugnissen::Verarbeitendes_Gewerbe.
Maschinenbau::Verarbeitendes_Gewerbe.
Fahrzeugbau::Verarbeitendes_Gewerbe.

H_von_Moebeln_Schmuck_Musikinstrumenten_Sportgeraeten_Spielwaren::
Verarbeitendes_Gewerbe.
Land_und_Forstwirtschaft::Unternehmen.
Landwirtschaft_und_Jagd::Land_und_Forstwirtschaft.
Forstwirtschaft::Land_und_Forstwirtschaft.
Steinkohlenbergbau::Kohlenbergbau.
Braunkohlenbergbau::Kohlenbergbau.
Ernaehrungsgewerbe::Verarbeitendes_Gewerbe.
H_von_Maschinen_zur_Erzeugung_u_Nutzung_mechanischer_Energie::Maschinenbau.
H_von_nicht_wirtschaftszweigspezifischen_Maschinen::Maschinenbau.
H_von_land_und_forstwirtschaftlichen_Maschinen::Maschinenbau.
H_von_Werkzeugmaschinen::Maschinenbau.
H_v_Verbrennungsmotoren_und_Turbinen::
H_von_Maschinen_zur_Erzeugung_u_Nutzung_mechanischer_Energie.
H_v_Pumpen_und_Kompressoren::
H_von_Maschinen_zur_Erzeugung_u_Nutzung_mechanischer_Energie.
H_v_Armaturen::H_von_Maschinen_zur_Erzeugung_u_Nutzung_mechanischer_Energie.
H_v_Lagern_Getrieben_Zahnraedern_und_Antriebselementen::
H_von_Maschinen_zur_Erzeugung_u_Nutzung_mechanischer_Energie.
H_v_Oefen_und_Brennern::H_von_nicht_wirtschaftszweigspezifischen_Maschinen.
H_v_Hebezeugen_und_Fordermitteln::H_von_nicht_wirtschaftszweigspezifischen_Maschinen.
H_v_kaelte_und_lufttechnischen_Erzeugnissen::
H_von_nicht_wirtschaftszweigspezifischen_Maschinen.
H_v_land_und_fortswirtschaftlichen_Zugmaschinen::
H_von_land_und_forstwirtschaftlichen_Maschinen.
H_v_handgefuehrten_kraftbetriebenen_Werkzeugen::H_von_Werkzeugmaschinen.
H_v_Werkzeugmaschinen_fuer_die_Metallbearbeitung::H_von_Werkzeugmaschinen.
Erzeugung_v_Roheisen_Stahl_u_Ferrolegierungen::Metallerzeugung_und_bearbeitung.
H_v_Rohren::Metallerzeugung_und_bearbeitung.
Erzeugung_u_erste_Bearbeitung_v_NE_Metallen::Metallerzeugung_und_bearbeitung.
Giesserei::Metallerzeugung_und_bearbeitung.
Herstellung_v_Metallerzeugnissen::
Herstellung_Erzeugung_u_Bearbeitung_v_Metallerzeugnissen.
H_v_Kraftwagen_u_Kraftwagenteilen::Fahrzeugbau.
H_v_Kraftwagen_u_Kraftwagenmotoren::H_v_Kraftwagen_u_Kraftwagenteilen.
H_v_Karosserien_Aufbauten_u_Anhaengern::H_v_Kraftwagen_u_Kraftwagenteilen.
H_v_Teilen_u_Zubehoer_f_Kraftwagen_u_Kraftwagenmotoren::
H_v_Kraftwagen_u_Kraftwagenteilen.
Sonstiger_Fahrzeugbau::Fahrzeugbau.
Schiff_u_Bootsbau::Sonstiger_Fahrzeugbau.
Bahnindustrie::Sonstiger_Fahrzeugbau.
Schienefahrzeugbau::Bahnindustrie.
Herstellung_v_Eisenbahninfrastruktur::Bahnindustrie.
Luft_u_Raumfahrzeugbau::Sonstiger_Fahrzeugbau.
H_v_Kraftraedern_Fahrraedern_u_Behindertenfahrzeugen::Sonstiger_Fahrzeugbau.
H_v_Kraftraedern::H_v_Kraftraedern_Fahrraedern_u_Behindertenfahrzeugen.
H_v_Fahrraedern::H_v_Kraftraedern_Fahrraedern_u_Behindertenfahrzeugen.
H_v_Behindertenfahrzeugen::H_v_Kraftraedern_Fahrraedern_u_Behindertenfahrzeugen.
Recycling::Verarbeitendes_Gewerbe.
Recycling_v_metallischen_Altmaterialien_u_Reststoffen::Recycling.
Recycling_v_nichtmetallischen_Altmaterialien_u_Reststoffen::Recycling.
H_v_Moebeln::H_von_Moebeln_Schmuck_Musikinstrumenten_Sportgeraeten_Spielwaren.
H_v_Schmuck::H_von_Moebeln_Schmuck_Musikinstrumenten_Sportgeraeten_Spielwaren.
H_v_Musikinstrumenten::
H_von_Moebeln_Schmuck_Musikinstrumenten_Sportgeraeten_Spielwaren.
H_v_Sportgeraeten::
H_von_Moebeln_Schmuck_Musikinstrumenten_Sportgeraeten_Spielwaren.
H_v_Spielwaren::H_von_Moebeln_Schmuck_Musikinstrumenten_Sportgeraeten_Spielwaren.
Energieversorgung::Energie_und_Wasserversorgung.
Elektrizitaetsversorgung::Energieversorgung.
Gasversorgung::Energieversorgung.

Waermeversorgung::Energieversorgung.
Wasserversorgung::Energie_und_Wasserversorgung.
Vorbereitende_Baustellenarbeiten::Baugewerbe.
Hoch_u_Tiefbau::Baugewerbe.
Bauinstallation::Baugewerbe.
Sonstiges_Ausbaugewerbe::Baugewerbe.
Vermietung_v_Baumaschinen_und_geraeten::Baugewerbe.
Handel_mit_Kraftwagen::
Kraftfahrzeughandel_Instandhaltung_u_Reparatur_v_Kraftfahrzeugen_Tankstellen.
Kraftfahrzeughandel_Instandhaltung_u_Reparatur_v_Kraftfahrzeugen_Tankstellen::
Handel_Instandhaltung_u_Raperatur_v_Kraftfahrzeugen_u_Gebrauchsguetern.
Instandhaltung_und_Reparatur_v_Kraftwagen::
Kraftfahrzeughandel_Instandhaltung_u_Reparatur_v_Kraftfahrzeugen_Tankstellen.
Handel_m_Kraftwagenteilen_u_Zubehoer::
Kraftfahrzeughandel_Instandhaltung_u_Reparatur_v_Kraftfahrzeugen_Tankstellen.

// Zeit
Zeitpunkt::zeitartiges_Erfahrungsobjekt.
Zeitspanne::zeitartiges_Erfahrungsobjekt.
Ereignis::Aktivitaet.
Prozess::Aktivitaet.
Vorgang::Prozess.
natuerlicher_Vorgang::Vorgang.
nicht_natuerlicher_Vorgang::Vorgang.
Handlung::Prozess.
Vor_Angebotsphase::Handlung.
Angebotsphase::Handlung.
Auftrag_durchfuehren::Handlung.
Projekt_nachbearbeiten::Handlung.
Projekte_unterstuetzen::Handlung.

// Phasen
Markt_beobachten::Vor_Angebotsphase.
Projekt_identifizieren::Vor_Angebotsphase.
Machbarkeit_pruefen::Vor_Angebotsphase.
Interessenbekundung::Vor_Angebotsphase.
Follow_up_des_PQ_LOI::Vor_Angebotsphase.
Kundenproblem_analysieren::Angebotsphase.
Loesungsansaetze_finden::Angebotsphase.
Angebot_kalkulieren::Angebotsphase.
Risiken_abschaetzen::Angebotsphase.
Verhandlung_und_Angebotsabschluss::Angebotsphase.
Start_und_Inception_Phase::Auftrag_durchfuehren.
Einzelauftrag_bearbeiten::Auftrag_durchfuehren.
Projekt_managen_und_controllen::Auftrag_durchfuehren.
Auftrag_abschliessen::Auftrag_durchfuehren.
Projektbeurteilung_durchfuehren::Projekt_nachbearbeiten.
Projekt_abschliessen::Projekt_nachbearbeiten.
Kontaktpflege_zum_Kunden_einleiten::Projekt_nachbearbeiten.
Wartung_und_Service_durchfuehren::Projekt_nachbearbeiten.
Strategie_entwickeln_ausrichten::Projekte_unterstuetzen.
Marketingprogramm_entwickeln::Projekte_unterstuetzen.
Vertrieb::Projekte_unterstuetzen.
Rechtsberatung::Projekte_unterstuetzen.
Personal_managen::Projekte_unterstuetzen.
Infrastruktur_bereitstellen::Projekte_unterstuetzen.
Rechnungswesen::Projekte_unterstuetzen.

// Gebiete
geographisches_Gebiet::Gebiet.
politisches_Gebiet::Gebiet.

Staat::politisches_Gebiet.
Bundesland::politisches_Gebiet.
Strasse::politisches_Gebiet.
Ort::politisches_Gebiet.

// Aussagen
Kompetenzaussage::dreistellige_Aussage.
dreistellige_Aussage::Aussage.
vierstellige_Aussage::Aussage.

// Zahlen
reelle_Zahl::Quantitaet.
imaginaere_Zahl::Quantitaet.
rationale_Zahl::reelle_Zahl.
irrationale_Zahl::reelle_Zahl.

// Aussagen
einstellige_Aussage::Aussage.
zweistellige_Aussage::Aussage.
Potenzialaussage::zweistellige_Aussage.

// Kompetenzausprägungen
Neuling::Kompetenzauspraegung.
Anfaenger::Kompetenzauspraegung.
Problemloeser::Kompetenzauspraegung.
Erfahrener::Kompetenzauspraegung.
Experte::Kompetenzauspraegung.

// Profile
Akteurs_Profil::Kompetenzprofil.
Stellen_Profil::Kompetenzprofil.

// Projekt
Projekt::Handlung.
Projekterfolg::ordinal_skalierte_Intensitaet.
Projektmitarbeiter::Individualakteur.

// Schulung
Schulung::Handlung.
Schulungsniveau::ordinal_skalierte_Intensitaet.

// Publikation
Publikation::Gegenstand.

// Vortrag
Vortrag::Handlung.
Abteilung::Organisation.

// Software
Software::Gegenstand.
Softwarebewertung::ordinal_skalierte_Intensitaet.

// Lokale Relationen ----------

Kompetenz[enthalten_in_Kompetenzaussage=>>Kompetenzaussage].
objektsprachliche_Entitaet[wird_definiert_durch=>>Definition].
Akteur[hat_Name=>>xsdSTRING;

--- ist_betroffen_von_Kompetenzaussage=>>Kompetenzaussage;
--- hat_Kompetenzprofil=>Kompetenzprofil;
--- hat_Geburtstag=>Zeitpunkt;
--- hat_Adresse=>>Punkt;
--- lebte_in_Staat=>>Staat;
--- ist_Autor=>>Publikation;
--- hat_programmiert_Software=>>Software].
Kompetenzauspraegung[hat_numerischen_Wert=>xsdINTEGER].
Individualakteur[arbeitet_fuer=>>Unternehmen;
--- gehoert_zu_Team=>>Team;
--- hat_Vorgesetzten=>Individualakteur;
--- hat_gearbeitet_fuer=>>Kollektivakteur;
--- hat_gearbeitet_in_Abteilung=>>Abteilung;
--- Arbeitsdauer_in_Abteilung=>>xsdINTEGER;
--- hat_durchgefuehrt_Schulung=>>Schulung;
--- hat_Vortrag_gehalten=>>Vortrag].
Unternehmensnetzwerk[hat_Mitgliedsunternehmen=>>Unternehmen].
Unternehmen[ist_betroffen_von_Kompetenzaussage=>>Kompetenzaussage;
--- hat_Mitarbeiter=>>Individualakteur;
--- ist_Mitglied_von_Unternehmensnetzwerk=>>Unternehmensnetzwerk].
Zeitpunkt[liegt_in_Zeitspanne=>>Zeitspanne].
Zeitspanne[hat_Beginn=>>Zeitpunkt;
--- hat_Ende=>>Zeitpunkt;
--- beinhaltet_Zeitspanne=>>Zeitspanne;
--- vor_Zeitspanne=>>Zeitspanne;
--- endet_am_Beginn_von_Zeitspanne=>>Zeitspanne;
--- nach_Zeitspanne=>>Zeitspanne;
--- waehrend_Zeitspanne=>>Zeitspanne;
--- beginnt_am_Ende_von_Zeitspanne=>>Zeitspanne;
--- ueberlappt_Zeitspanne=>>Zeitspanne;
--- gleiche_Zeitspanne_wie=>>Zeitspanne;
--- endet_mit_Ende_von_Zeitspanne=>>Zeitspanne;
--- beginnt_am_Anfang_von_Zeitspanne=>>Zeitspanne].
Staat[hat_Sprache=>>Sprache].
Kompetenzaussage[betrifft_Entitaet=>Entitaet;
--- beinhaltet_Kompetenz=>Kompetenz;
--- beinhaltet_Kompetenzauspraegung=>Kompetenzauspraegung;
--- enthalten_in_Kompetenzprofil=>Kompetenzprofil].
Kompetenzprofil[beinhaltet_Kompetenzaussage=>>Kompetenzaussage].
Relation[hat_Definitionsbereich=>>Konzept;
--- hat_Wertebereich=>>Konzept].
Konzept[vebunden_mit_Relation=>>Relation].
Definition[definiert=>>objektsprachliche_Entitaet].
Stelle[ist_betroffen_von_Kompetenzaussage=>>Kompetenzaussage;
--- hat_Stellenbeschreibung=>>xsdSTRING].
Projektmanager[leitet_Projekt=>>Projekt;
--- hat_weisungsbefugnis_ueber=>>Projektmitarbeiter].
Projekt[hat_Projektmanager=>Projektmanager;
--- hat_Bewertung=>Projekterfolg;
--- hat_Titel=>>xsdSTRING;
--- hat_Team=>Team;
--- hat_Projektmitarbeiter=>>Projektmitarbeiter;
--- arbeitet_fuer_Unternehmen=>>Unternehmen].
Projektmitarbeiter[arbeitet_in_Projekt=>>Projekt;
--- „gehört_zu_Team“=>>Team;
--- hat_Praktikum_gemacht=>>Unternehmen].
Schulung[hat_schulungsniveau=>Schulungsniveau;
--- hat_Gegenstandsbereich=>>Kompetenz;
--- hat_Teilnehmer=>>Individualakteur].
Potenzialaussage[betrifft_Entitaet=>Akteur;
--- beinhaltet_Kompetenz=>Kompetenz].

```
Sprache[wird_gesprochen_in=>>Staat].
Publikation[hat_Autor=>>Akteur;
--- ist_geschrieben_in_Sprache=>Sprache;
--- hat_Thematik=>>Kompetenz].
Vortrag[hat_Thema=>>Kompetenz].
Software[ist_programmiert_von=>>Akteur;
--- ist_programmiert_in_Programmiersprache=>>Programmiersprachen_Kompetenz;
--- hat_Softwarebewertung=>Softwarebewertung].
```

// Instanzen ----------

```
// Selbstkompetenzen
Eigenstaendigkeit:Selbstkompetenz.
Strukturierungsfaehigkeit:Selbstkompetenz.
Lernbereitschaft:Selbstkompetenz.
Einsatzbereitschaft:Selbstkompetenz.
Verantwortungsbereitschaft:Selbstkompetenz.
Reflexionsfaehigkeit:Selbstkompetenz.
Flexibilitaet:Selbstkompetenz.
Zuverlaessigkeit:Selbstkompetenz.
Risikobereitschaft:Selbstkompetenz.
Belastbarkeit:Selbstkompetenz.
Entscheidungsfaehigkeit:Selbstkompetenz.
Systemisches_Denken:Selbstkompetenz.
Kreativitaet:Selbstkompetenz.

// Sozialkompetenz
Kommunikationsfaehigkeit:Sozialkompetenz.
Kontaktfreudigkeit:Sozialkompetenz.
Durchsetzungsfaehigkeit:Sozialkompetenz.
Didaktische_Faehigkeit:Sozialkompetenz.
Delegationsfaehigkeit:Sozialkompetenz.
Motivationsfaehigkeit:Sozialkompetenz.
Kritikfaehigkeit:Sozialkompetenz.
Kooperationsfaehigkeit:Sozialkompetenz.
Konfliktloseungsfaehigkeit:Sozialkompetenz.
Koordinationsfaehigkeit:Sozialkompetenz.

// Branchenkompetenz
Maschinen_und_Anlagenbau_Kompetenz:Branchenkompetenz.
Automobilindustrie_Kompetenz:Branchenkompetenz.

// Juristische Kompetenz
HGB_Kompetenz:Juristische_Kompetenz.
InsO_Kompetenz:Juristische_Kompetenz.
EStR_Kompetenz:Juristische_Kompetenz.

// Methoden-Kompetenz
Morphologischer_Matrix_Kompetenz:Analytisch_systematische_Methoden_Kompetenz.
Problemloesungsbaum_Kompetenz:Analytisch_systematische_Methoden_Kompetenz.
Brainstorming_Kompetenz:Intuitiv_kreative_Methoden_Kompetenz.
Methode_635_Kompetenz:Intuitiv_kreative_Methoden_Kompetenz.
Synektik_Kompetenz:Intuitiv_kreative_Methoden_Kompetenz.

// Bewertungsverfahren-Kompetenz
Ertragswertverfahhren_Kompetenz:Gesamtbewertungsverfahren_Kompetenz.
DCF_Verfahren_Kompetenz:Gesamtbewertungsverfahren_Kompetenz.
Vergleichswertverfahren_Kompetenz:Gesamtbewertungsverfahren_Kompetenz.
```

Substanzwertverfahren_Kompetenz:Einzelbewertungsverfahren_Kompetenz.
Liquidationswertverfahren_Kompetenz:Einzelbewertungsverfahren_Kompetenz.
EVA_Verfahren_Kompetenz:Gesamtbewertungsverfahren_Kompetenz.

// Anwendungsprogramm-Kompetenz
BAAN_Kompetenz:Anwendungsprogramme_Kompetenz.
Bea_Logic_Kompetenz:Anwendungsprogramme_Kompetenz.
Comet_Kompetenz:Anwendungsprogramme_Kompetenz.
Exchange_Kompetenz:Anwendungsprogramme_Kompetenz.
IBM_Websphere_Kompetenz:Anwendungsprogramme_Kompetenz.
MS_Backoffice_Kompetenz:Anwendungsprogramme_Kompetenz.
MS_Excel_97_Kompetenz:MS_Excel_Kompetenz.
MS_Word_2000_Kompetenz:MS_Word_Kompetenz.
MS_Powerpoint_2000_Kompetenz:MS_Powerpoint_Kompetenz.
MS_Powerpoint_XP_Kompetenz:MS_Powerpoint_Kompetenz.
MS_Access_97_Kompetenz:Datenbanken_Kompetenz.
MS_Powerpoint_97_Kompetenz:MS_Powerpoint_Kompetenz.
MS_Word_97_Kompetenz:MS_Word_Kompetenz.
MS_Word_XP_Kompetenz:MS_Word_Kompetenz.
MS_Excel_2000_Kompetenz:MS_Excel_Kompetenz.
MS_Excel_XP_Kompetenz:MS_Excel_Kompetenz.
MS_Visio_2002_Kompetenz:MS_Visio_Kompetenz.
HTML_2_Kompetenz:HTML_Kompetenz.
HTML_3_2_Kompetenz:HTML_Kompetenz.
HTML_4_Kompetenz:HTML_Kompetenz.
XHTML_1_Kompetenz:XHTML_Kompetenz.
XML_1_Kompetenz:XML_Kompetenz.
XML_1_1_Kompetenz:XML_Kompetenz.
infonea_2_x_Kompetenz:infonea_Kompetenz.
infonea_3_0_Kompetenz:infonea_Kompetenz.
Helpline_2_x_Kompetenz:Helpline_Kompetenz.
Helpline_3_0_Kompetenz:Helpline_Kompetenz.
SAP_R_3_Kompetenz:SAP_Kompetenz.
SAP_R_2_Kompetenz:SAP_Kompetenz.

// Betriebssystem-Kompetenz
IBM_OS2_Kompetenz:IBM_Betriebssystem_Kompetenz.
Linux_Kompetenz:Unix_Betriebssystem_Kompetenz.
Windows_2000_Kompetenz:Windows_Betriebssystem_Kompetenz.
Windows_95_Kompetenz:Windows_Betriebssystem_Kompetenz.
Windows_98_Kompetenz:Windows_Betriebssystem_Kompetenz.
Windows_XP_Kompetenz:Windows_Betriebssystem_Kompetenz.

// Netz-Kompetenz
IP_Family_Kompetenz:Netzwerkprotokoll_Kompetenz.
OSI_Layer_Kompetenz:Netzwerkprotokoll_Kompetenz.
TCP_IP_Kompetenz:Netzwerkprotokoll_Kompetenz.
Ringtopologie_Kompetenz:Netztopologien_Kompetenz.
Sterntopologie_Kompetenz:Netztopologien_Kompetenz.
ATM_Kompetenz:Netzwerkarchitekturen_Kompetenz.
Ethernet_Netze_Kompetenz:Netzwerkarchitekturen_Kompetenz.
FDDI_Kompetenz:Netzwerkarchitekturen_Kompetenz.
ServerClient_Architektur_Kompetenz:Netzwerkarchitekturen_Kompetenz.
TokenRing_Netze_Kompetenz:Netzwerkarchitekturen_Kompetenz.

// Datenbank-Kompetenz
MS_Acess_2000_Kompetenz:Datenbanken_Kompetenz.

// Definitionen
Java_Kompetenz:Programmiersprachen_Kompetenz.
Stelle_1:Stelle.

„1“:Neuling.
„1“[hat_numerischen_Wert->>1.0;
--- wird_definiert_durch->>
„_Zerlegt_die_Aufgabe_in_kontextfreie_Teile._Dadurch__dass_er_wenige_allgemeine_Regeln_beherrscht__ist_er_in_seinem_Handeln_sehr_langsam.__“].
„2“:Anfaenger.
„2“[hat_numerischen_Wert->>2.0;
--- wird_definiert_durch->>
„_Weist_die_F_higkeit_auf__situative_Aspekte_in_sein_Handeln_einzubinden._Er_lernt_aus_realen_Situationen_und_kann_seine_Kompetenz_ausbauen.__“].
„3“:Problemloeser.
„3“[hat_numerischen_Wert->>3.0;
--- wird_definiert_durch->>
„_Ist_flexibel_in_seinem_Handeln__da_er_die_Gesamtsituation__berblickt_und_Strategien_entwickeln_kann.__“].
„4“:Erfahrener.
„4“[hat_numerischen_Wert->>4.0;
--- wird_definiert_durch->>
„_Kann_auf_-_f_r_die_Probleml_sung_relevantes_-_Wissen_zur_ckgreifen_und_seine_Entscheidungen_dementsprechend_konsolidieren.__“].
„5“:Experte.
„5“[hat_numerischen_Wert->>5.0;
--- wird_definiert_durch->>
„_Handelt_zielbewusst__ohne_einen_rationalen_Entscheidungsprozess_zu_durchlaufen._Sein_Handeln_erfolgt_instinktiv.__“].
KA_1_Stelle_1:Kompetenzaussage.
KA_1_Stelle_1[beinhaltet_Kompetenz->>Java_Kompetenz;
--- beinhaltet_Kompetenzauspraegung->>„4“;
--- betrifft_Entitaet->>Stelle_1].
KA_1_IA_1:Kompetenzaussage.
KA_1_IA_1[beinhaltet_Kompetenz->>Java_Kompetenz;
--- beinhaltet_Kompetenzauspraegung->>„2“;
--- betrifft_Entitaet->>IA_1].
erfolgreich:Projekterfolg.
durchschnittlich:Projekterfolg.
„nicht-erfolgreich“:Projekterfolg.
„Führungsfähigkeit“:Sozialkompetenz.

Projektmanager1:Projektmanager.
Projektmanager1[hat_Name->>„Torben“].
Projekt1:Projekt.
Projekt1[hat_Projektmanager->>Projektmanager1;
--- hat_Bewertung->>erfolgreich;
--- hat_Titel->>„KOWIEN“].
Projekt2:Projekt.
Projekt2[hat_Projektmanager->>Projektmanager1;
--- hat_Bewertung->>erfolgreich;
--- hat_Titel->>„MOTIWIDI“].

Beginner:Schulungsniveau.
Fortgeschritten:Schulungsniveau.
Professionell:Schulungsniveau.

// Axiome ----------

FORALL X,Y (X[enthalten_in_Kompetenzaussage->>Y]) <-> (Y[gehoert_zu_Akteur->>X]).
FORALL X,Y (X[enthalten_in_Kompetenzaussage->>Y]) <-> (Y[gehoert_zu_Akteur->>X]).
FORALL X,Y (X[arbeitet_fuer->>Y]) <-> (Y[hat_Mitarbeiter->>X]).

```
FORALL X,Y ( X[arbeitet_fuer->>Y] ) <-> ( Y[hat_Mitarbeiter->>X] ).
FORALL X,Y ( X[hat_Mitgliedsunternehmen->>Y] ) <-> (
Y[ist_Mitglied_von_Unternehmensnetzwerk->>X] ).
FORALL X,Y ( X[hat_Mitgliedsunternehmen->>Y] ) <-> (
Y[ist_Mitglied_von_Unternehmensnetzwerk->>X] ).
FORALL X,Y ( X[liegt_in_Zeitspanne->>Y] ) <-> ( Y[waehrend_Zeitspanne->>X] ).
FORALL X,Y ( X[liegt_in_Zeitspanne->>Y] ) <-> ( Y[waehrend_Zeitspanne->>X] ).
FORALL X,Y ( X[nach_Zeitspanne->>Y] ) <-> ( Y[vor_Zeitspanne->>X] ).
FORALL X,Y ( X[nach_Zeitspanne->>Y] ) <-> ( Y[vor_Zeitspanne->>X] ).
FORALL X,Y,Z ( X[nach_Zeitspanne->>Z] ) <- ( (X[nach_Zeitspanne->>Y] and
Y[nach_Zeitspanne->>Z]) ).
FORALL X,Y,Z ( X[waehrend_Zeitspanne->>Z] ) <- ( (X[waehrend_Zeitspanne->>Y] and
Y[waehrend_Zeitspanne->>Z]) ).
FORALL X,Y ( X[gleiche_Zeitspanne_wie->>Y] ) <- ( Y[gleiche_Zeitspanne_wie->>X] ).
FORALL X,Y,Z ( X[gleiche_Zeitspanne_wie->>Z] ) <- ( (X[gleiche_Zeitspanne_wie->>Y] and
Y[gleiche_Zeitspanne_wie->>Z]) ).
FORALL X,Y ( X[betrifft_Entitaet->>Y] ) <-> ( Y[ist_betroffen_von_Kompetenzaussage->>X] ).
FORALL X,Y ( X[betrifft_Entitaet->>Y] ) <-> ( Y[ist_betroffen_von_Kompetenzaussage->>X] ).
FORALL X,Y ( X[beinhaltet_Kompetenz->>Y] ) <-> ( Y[enthalten_in_Kompetenzaussage->>X] ).
FORALL X,Y ( X[beinhaltet_Kompetenz->>Y] ) <-> ( Y[enthalten_in_Kompetenzaussage->>X] ).
FORALL X,Y ( X:Konzept ) <- ( X::Y ).
FORALL X,Y,R ( (R:Relation and (R[hat_Definitionsbereich->>X] and R[hat_Wertebereich-
>>Y])) ) <- ( (X[R=>>Y] or X[R=>Y]) ).
FORALL X,Y,R ( (R:Relation and (R[hat_Definitionsbereich->>X] and R[hat_Wertebereich-
>>Y])) ) <- ( (X[R=>>Y] or X[R=>Y]) ).
FORALL R,W,X,Y,Z ( Y:W ) <- ( (X:Z[R->>Y] and Z[R=>W]) ).
FORALL IA,St,Kom,KAuss_IA,KAuss_St,KAusp_IA,KAusp_St,KAusp_IA_Nr,KAusp_St_Nr,Diff (
eignungswert(IA,St,Kom,Diff) ) <- ( (IA:Individualakteur[ist_betroffen_von_Kompetenzaussage-
>>KAuss_IA] and (KAuss_IA:Kompetenzaussage[betrifft_Entitaet->>IA;beinhaltet_Kompetenz-
>>Kom;beinhaltet_Kompetenzauspraegung->>KAusp_IA] and
(KAusp_IA:Kompetenzauspraegung[hat_numerischen_Wert->>KAusp_IA_Nr] and
(St:Stelle[ist_betroffen_von_Kompetenzaussage->>KAuss_St] and
(KAuss_St:Kompetenzaussage[betrifft_Entitaet->>St;beinhaltet_Kompetenz-
>>Kom;beinhaltet_Kompetenzauspraegung->>KAusp_St] and
(KAusp_St:Kompetenzauspraegung[hat_numerischen_Wert->>KAusp_St_Nr] and evalu-
able_(Diff,-(KAusp_IA_Nr,KAusp_St_Nr)))))))) ).
FORALL Kompetenzaussage1 ( Kompetenzaussa-
ge1:Kompetenzaussage[beinhaltet_Kompetenz->„Führungsfähigkeit“;betrifft_Entitaet-
>Mitarbeiter1;beinhaltet_Kompetenzauspraegung->5] ) <- ( (Mitarbei-
ter1:Projektmanager[leitet_Projekt->>ProjektA] and (Mitarbeiter1:Projektmanager[leitet_Projekt-
>>ProjektB] and (ProjektA:Projekt[hat_Bewertung->>„erfolgreich“] and (Pro-
jektB:Projekt[hat_Bewertung->>„erfolgreich“] and not (equal(ProjektA,ProjektB)))))) ).
FORALL Individualakteur1 <- Individualakteur1:Individualakteur.

FORALL Mitarbeiter1, Vorgang1, Kompetenz1, REL
Mitarbeiter1[REL->Anfaenger]<-
--- Vorgang1: Schulung-- [hat_Gegenstandsbereich->>Kompetenz1;
--- ---- --- --- hat_Schulungsniveau->professionell;
--- ---- --- --- hat_Teilnehmer->>Mitarbeiter1]
--- ---- --- --- AND zugeordnet(Kompetenz1,REL)

FORALL Mitarbeiter1, Mitarbeiter2, Mitarbeiter3
Mitarbeiter1 [hat_Vorgesetzten->>Mitarbeiter3]<-
--- Mitarbeiter1:Mitarbeiter [hat_Vorgesetzten->>Mitarbeiter2] AND
--- Mitarbeiter2:Mitarbeiter [hat_Vorgesetzten->>Mitarbeiter3].

FORALL Mitarbeiter1, Projekt1, Projekt2
Mitarbeiter1[hat_Fuehrungsfaehigkeiten->Experte]<-
--- Mitarbeiter1:Mitarbeiter[leitet_Projekt->>Projekt1] AND
--- Mitarbeiter1:Mitarbeiter[leitet_Projekt->>Projekt2] AND
--- NOT EQUAL (Projekt1, Projekt2) AND
```

```
--- Projekt1:Projekt[hat_bewertung->erfolgreich] AND
--- Projekt2:Projekt[hat_bewertung->erfolgreich].

FORALL Mitarbeiter1, Projekt1, Projekt2
Mitarbeiter1[hat_Teamfaehigkeit->Erfahrener]<-
--- Mitarbeiter1:Mitarbeiter[arbeitet_in_Projekt->>Projekt1] AND
--- Mitarbeiter1:Mitarbeiter[arbeitet_in_Projekt->>Projekt2] AND
--- NOT EQUAL (Projekt1, Projekt2) AND
--- Projekt1:Projekt[hat_bewertung->erfolgreich] AND
--- Projekt2:Projekt[hat_bewertung->erfolgreich].

FORALL Mitarbeiter1, Mitarbeiter2
Mitarbeiter1:Mitarbeiter[hat_Vorgesetzten->Mitarbeiter2]<->
--- Mitarbeiter2:Manager [hat_Weisungsbefugnis_ueber->>Mitarbeiter1].

FORALL Mitarbeiter1, Software1, Software2
Mitarbeiter1[hat_Javakompetenz->Erfahrener]<-
--- Mitarbeiter1:Mitarbeiter[hat_programmiert_Software->>Software1] AND
--- Mitarbeiter1:Mitarbeiter[hat_programmiert_Software->>Software2] AND
--- Software1:Software[ist_programmiert_in_Programmiersprache->Java] AND
--- Software2:Software[ist_programmiert_in_Programmiersprache->Java] AND
--- NOT EQUAL (Software1,Software2) AND
--- Software1:Software[hat_Softwarebewertung->zufrieden_stellend] AND
--- Software2:Software[hat_Softwarebewertung->zufrieden_stellend].

FORALL Mitarbeiter1
Mitarbeiter1[hat_Konstruktionskompetenz->lernfaehig]<-
--- Mitarbeiter1[hat_Mathematikkompetenz->Experte].

FORALL Mitarbeiter1
Mitarbeiter1[hat_Konstruktionskompetenz->lernfaehig]<-
--- Mitarbeiter1[hat_Mathematikkompetenz->Experte] OR
--- Mitarbeiter1[hat_Physikkompetenz->Experte].

FORALL Mitarbeiter1
Mitarbeiter1[hat_technische_Ablaufkompetenz->Erfahrener]<-
--- Mitarbeiter1:Mitarbeiter[hat_Logikkompetenz->Experte].

FORALL Mitarbeiter1, Projekt1
Mitarbeiter1[geeignet_fuer->Projekt1]<-
--- Mitarbeiter1:Mitarbeiter[hat_Logikkompetenz->Experte]
--- AND Projekt1:Projekt.

FORALL Mitarbeiter1, Sprache1, Land1, REL
Mitarbeiter[REL->Erfahrener]<-
--- Mitarbeiter1:Mitarbeiter[lebte_in_Staat->Land1] AND
--- Land1:Staat[hat_Sprache->Sprache1] AND
--- NOT EQUAL (Sprache1, „deutsch“) AND
--- zugeordnet (Sprache1,REL).

FORALL Mitarbeiter1, Unternehmen1, Branchenklassifikation1, REL
Mitarbeiter1[REL->Erfahrener]<-
--- Mitarbeiter1:Mitarbeiter[hat_gearbeitet_fuer->>Unternehmen1] AND
--- Unternehmen1[taetig_in_Branche->>Branchenklassifikation1] AND
--- zugeordnet(Branchenklassifikation1,REL).

FORALL Mitarbeiter1, Unternehmen1, REL
Mitarbeiter1[REL->Erfahrener]<-
--- Mitarbeiter1:Mitarbeiter[hat_gearbeitet_fuer->>Unternehmen1] AND
--- zugeordnet(Unternehmen1,REL).
```

FORALL Mitarbeiter1, Kunde1, REL
Mitarbeiter1[REL->Erfahrener]<-
--- Mitarbeiter1:Mitarbeiter[hat_gearbeitet_mit->>Kunde1] AND
--- AND zugeordnet(Kunde1,REL).

FORALL Mitarbeiter1, Unternehmen1, Branchenklassifikation1, REL
Bewerber[REL->>Anfaenger<-
--- Person1:Bewerber[hat_Praktikum_gemacht->>Unternehmen1] AND
--- Unternehmen1[taetig_in_Branche->>Branchenklassifikation1] AND
--- zugeordnet(Branchenklassifikation1,REL).

FORALL Mitarbeiter1, Kompetenz1, Schulung1, REL
Mitarbeiter1[REL->>Erfahrener) AND
Mitarbeiter1[hat_Praesentationsfaehigkeit->Erfahrener] <-
--- Mitarbeiter1:Mitarbeiter[hat_durchgefuehrt->>Schulung1] AND
--- Schulung1:Schulung[hat_Gegenstandsbereich->>Kompetenz1) AND
--- zugeordnet(Kompetenz1,REL).

FORALL Mitarbeiter1, Sprache1, Projekt1, REL
Mitarbeiter1[REL->Erfahrener]<-
--- Mitarbeiter1:Mitarbeiter[hat_gearbeitet_in->Projekt1] AND
--- Projekt1:Projekt[hat_Sprache -> Sprache1] AND
--- NOT EQUAL (Sprache1, deutsch) AND
--- zugeordnet(Sprache1,REL).

FORALL Mitarbeiter1, Sprache1, Publikation1, REL
Mitarbeiter1[REL->Erfahrener]<-
--- Mitarbeiter1:Mitarbeiter[hat_Publikation->Publikation1] AND
--- Publikation1:Publikation[hat_Sprache->Sprache1] AND
--- NOT EQUAL (Sprache1,deutsch) AND
--- zugeordnet(Sprache1,REL).

FORALL Mitarbeiter1, Publikation1, Thema1, REL
Mitarbeiter1[REL->Erfahrener]<-
--- Publikation1: Publikation [hat_Thema->>Thema1;
--- ---- --- --- --- hat_Autor->>Mitarbeiter1] AND
--- zugeordnet(Thema1,REL).

FORALL Mitarbeiter1, Vortrag1, Thema1, REL
Person1[REL->Erfahrener]<-
--- Vortrag1:Vortrag---[hat_Thema->>Thema1;
--- ---- --- --- hat_Referenten ->> Person1] AND