

Semantische Informationsintegration – Konzeption eines auf Beschreibungslogiken basierenden Integrationssystems für die Produktentwicklung

Zur Erlangung des akademischen Grades eines

DOKTORS DER INGENIEURWISSENSCHAFTEN

von der Fakultät für Maschinenbau des
Karlsruher Instituts für Technologie (KIT)
genehmigte

DISSERTATION

von

Dipl.-Inform. Vitalis Bittel

Tag der mündlichen Prüfung:

11. Februar 2014

Hauptreferent:

Prof. Dr. Dr.-Ing. Dr. h. c. Jivka Ovtcharova

Korreferent:

Prof. Dipl.-Ing. Dr.-Ing. Detlef Gerhard

Vorwort

Die vorliegende Dissertation entstand neben meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Informationsmanagement im Ingenieurwesen (IMI) des Karlsruher Instituts für Technologie (KIT).

Mein besonderer Dank gilt Frau Prof. Dr. Dr.-Ing. Dr. h. c. Jivka Ovtcharova, Leiterin des Instituts, für die Übernahme des Hauptreferats und das mir entgegengebrachte Vertrauen sowie die wissenschaftliche Betreuung bei der Entstehung der Arbeit.

Herrn Prof. Dipl.-Ing. Dr.-Ing. Detlef Gerhard danke ich für die eingehende Durchsicht der Dissertation und die sich daraus ergebenden wertvollen Hinweise sowie die Übernahme des Korreferates.

Darüber hinaus danke ich allen Mitarbeitern des Instituts, die mich durch anregende Kritik und stete Einsatzbereitschaft unterstützt haben. Dieser Dank gilt besonders den ehemaligen Kollegen meiner Arbeitsgruppe *Process Engineering* Dr.-Ing. Stilian Stanev, Dipl.-Wi.-Ing. Alexander Burger und Dipl.-Wi.-Ing. Ramez Awad. Dank gebührt weiterhin meinen studentischen Hilfskräften Herrn cand.-mach. Franz Jost und Herrn cand.-inform. Sebastian Imm, die mir mit Ihrem überdurchschnittlichen Engagement tatkräftig zur Seite standen.

Ganz besonders möchte ich mich bei meinen Eltern bedanken, die mir durch ihre Unterstützung während meiner gesamten Ausbildungszeit erst die Möglichkeit gegeben haben, diese Arbeit zu erstellen. Ebenso danke ich Nikolaus und Katharina, die immer an die Realisierung meiner Ziele geglaubt und mich stets auf meinem Lebensweg begleitet und unterstützt haben.

Der größte Dank gilt meiner Frau Shanna, die mich motiviert und angespornt hat, und unserer gemeinsamen Tochter Karina. Sie mussten beide auf viele gemeinsame Stunden mit ihrem Mann bzw. Vater verzichten. Ihnen möchte ich diese Arbeit widmen.

Karlsruhe,
im Februar 2014

Vitalis Bittel
Karlsruher Institut für Technologie (KIT)

Inhaltsverzeichnis

Vorwort	3
1 Einleitung	1
1.1 Ausgangssituation und Motivation	1
1.2 Zielsetzung der Arbeit	5
1.3 Vorgehensweise und Struktur der Arbeit	6
2 Grundlagen und Begriffsabgrenzung	9
2.1 Begriffsbestimmung	9
2.1.1 Daten, Information und Wissen	9
2.1.2 Wissensarten	11
2.1.3 Produktinformation/Produktwissen	11
2.2 Informationsintegration	12
2.2.1 Grundlagen	13
2.2.2 Herausforderungen der Informationsintegration	16
2.2.3 Eigenschaften des Integrationsformalismus	20
2.3 Formale Wissensrepräsentation und Inferenz	22
2.3.1 Wissensbasierte Systeme	22
2.3.2 Logikbasierte Wissensrepräsentation	24
2.4 Ontologien und Beschreibungslogiken	32
2.4.1 Ontologie	32
2.4.2 Beschreibungslogiken und Ontologiesprachen	34
2.5 Inferenzprobleme für Beschreibungslogiken	38
2.5.1 Konjunktive Anfragen für Beschreibungslogiken	39
2.5.2 Umschreiben konjunktiver Anfragen für \mathcal{DL}	42
2.6 Zusammenfassung	42
3 Stand der Forschung und der Technik	45
3.1 Allgemeine IT-Integrationsansätze	45
3.1.1 Punkt-zu-Punkt Verbindungen	45
3.1.2 Middleware-basierte Integration	46
3.1.3 Enterprise Application Integration (EAI)	51

3.1.4	Serviceorientierte Architektur (SOA)	56
3.1.5	Semantische Technologien	58
3.2	Produktdatenintegration	60
3.2.1	Standard-Datenmodelle	61
3.2.2	PDM-basierte Integration	66
3.2.3	ERP-basierte Integration	68
3.3	Relevante Forschungsaktivitäten	71
3.3.1	iViP	71
3.3.2	PDTnet	72
3.3.3	SEVENPRO	74
3.3.4	The Semaril	75
3.4	Zusammenfassende Bewertung bestehender Ansätze	77
4	Anforderungen an ein System zur semantischen Informationsintegration	81
4.1	Allgemeine Anforderungen an ein System zur ontologiebasierten Informationsintegration	81
4.2	Anforderungen an die produktbeschreibenden Datenbestände	82
4.3	Anforderungen an die Wissensrepräsentation	83
4.4	Anforderungen an die Inferenzmechanismen	85
4.5	Anforderungen an die Bereitstellung der Anfrageergebnisse	85
4.6	Sicherheitsrelevante Anforderungen	86
4.7	Anforderungen an die Architektur des Systems	86
5	Konzept zur semantischen Informationsintegration in der Produktentwicklung	89
5.1	Datenanalyse und -aufbereitung	94
5.1.1	Datenstrukturen und Sichtenbildung	95
5.1.2	Relationale Sicht auf hierarchische Datenstrukturen	97
5.1.3	Relationale Sicht auf schwach strukturierte Datenstrukturen	98
5.2	Konzept- und Rollenermittlung	99
5.2.1	Ansatz zur automatischen Konzept- und Rollenermittlung	100
5.2.2	Der Wissensrepräsentationsformalismus \mathcal{L}_{tripod}	104
5.2.3	Normalisierung der TBox	110
5.3	Aufbau eines Integrationssystems	111
5.3.1	Verfahren zum Mapping	112
5.3.2	Mathematisches Modell zur semantischen Informationsintegration	118
5.4	Anwendung eines Integrationssystems	122

5.4.1	Umschreiben einer Anfrage bezüglich des Informationsintegrationssystem	122
5.4.2	Algorithmus zum Umschreiben von Anfragen für ontologiebasierte Informationsintegration	126
5.4.3	Bereitstellung und Nutzung der Integrationsergebnisse	134
5.4.4	Systemarchitektur	136
6	Realisierung und Validierung des Ansatzes	139
6.1	Beschreibung des Anwendungsszenarios	139
6.2	Entwicklungsumgebung und beteiligte Systeme	142
6.3	Vorgehensschritte bei einer semantischen Informationsintegration	144
6.4	Vorgehensschritte beim Ontologieentwurf	145
6.4.1	Die Benutzungsoberfläche des Ontologie-Entwicklungswerkzeugs	145
6.4.2	Erstellung der Ontologie	147
6.5	Spezifikation datenquellenspezifischer Korrespondenzen	159
6.5.1	Wrapper für relationale Datenbanken	160
6.5.2	Implementierung der Korrespondenzen	162
6.6	Vorgehensweise zur ontologiebasierten Informationssuche	165
6.7	Zusammenfassung	170
7	Schlussbemerkung	171
7.1	Inhalt der Arbeit	171
7.2	Nutzen der Arbeit	172
7.3	Ausblick	174
A	Resource Description Framework	177
B	The OWL Web Ontology Language	181
C	Index	207

Was wir wissen, ist ein Tropfen; was wir nicht wissen, ist ein Ozean.

ISAAC NEWTON (1642–1727)

1. Einleitung

1.1. Ausgangssituation und Motivation

Die Produktentwicklung im Internetzeitalter stellt Produktionsunternehmen vor neue Herausforderungen. Ursache hierfür sind anhaltende rasante gesellschaftliche, politische, gesetzliche und technologische Veränderungen in einem globalen Markt [AVR08]. Daraus resultieren steigende Anforderungen an die Verfügbarkeit von Informationen bezüglich zu entwickelnden und herzustellenden Produkte sowie die hierfür notwendige starke Vernetzung verschiedener Unternehmenseinheiten im Rahmen eines Zuliefererverbundes oder einer Kunde/Zulieferer-Beziehung [ES09].

Aufgrund der Entwicklung zunehmend komplexerer Produkte in immer kürzeren Produktentwicklungszeiten sowie aufgrund des massiven Einsatzes rechnergestützter Verfahren steigt die Menge verfügbarer Informationen sehr stark an [GLW02], [Haa07]. Die Generierung immer neuer Informationen hat zur Folge, dass die jeweils relevanten Informationen in der ständig anwachsenden Datenflut mit vertretbarem Aufwand nicht mehr auffindbar sind [KLA05]. EIGNER und STELZER beschreiben diese Situation wie folgt:

Wesentlich ist in jedem Fall, dass die teamorientierte Kommunikation und die zielgerichtete Bereitstellung von Informationen für den Ingenieur wesentlich für die Entscheidungsfindung sind und damit zukünftig den Geschäftserfolg mitbestimmen. Nicht die Masse der Informationen ist entscheidend, sondern es kommt darauf an, die richtigen Informationen dahin zu leiten, sinnvoll aufzubereiten und zu präsentieren, wo sie gebraucht werden. [ES09]

Die gespeicherten Dokumente und Daten sind demnach prinzipiell wertlos, wenn sie nach der Erfassung nicht in kürzester Zeit wiederzufinden sind und für aktuell durchzuführende Aufgaben- und Problemstellungen nicht verwendet werden können.

Gemessen an diesen Zuständen ergibt sich für die Unternehmenspraxis folgendes Bild: Die zur Verfügung stehenden Daten werden häufig nur akkumuliert und mithilfe von Dokumentenmanagementsystemen verwaltet. Dies hat zur Folge, dass die nicht in Informationen und Wissen umgewandelten Dokumente in großen Datenarchiven beiseitegelegt werden und deren möglicher Nutzen somit verloren geht. Das Resultat sind große Datenbestände, deren Inhalt häufig nur oberflächlich wahrgenommen oder gar ignoriert wird und somit kein nutzbringender Einsatz erfolgen kann, um beispielsweise die Mitarbeiterproduktivität zu steigern. Folglich lautet

die Frage heute nicht mehr, ob Informationen unternehmensweit gesammelt und verfügbar gemacht werden, sondern vielmehr, wie effizient dies geschehen kann.

Aufgrund der unaufhaltsamen Zunahme von Informationsbeständen und informationstechnischen Anwendungen sowie vor allem aufgrund des verstärkten Verlangens der Anwender, aus vielen Datenquellen gleichzeitig relevante Informationen zu beziehen, ist es wesentlich schwieriger, gezielte Informationsrecherche zu realisieren. Die gezielte Suche in einer großen Informationsmenge wird demnach für Unternehmen zunehmend unerlässlich. Eine neue Studie mit dem Titel „ECM-Studie 2012: Welche Trends Unternehmen bewegen“ zeigt, dass die Mehrheit der Unternehmen eine leicht bedienbare und leistungsfähige Suche fordert (Abbildung 1.1). Darüber hinaus sind die Unternehmen nicht ausschließlich an einer möglichst schnellen Suche interessiert, sondern auch an benutzerfreundlichen Features wie einer einheitlichen Ergebnisliste oder der Integration in ECM-Systeme [Nie12].

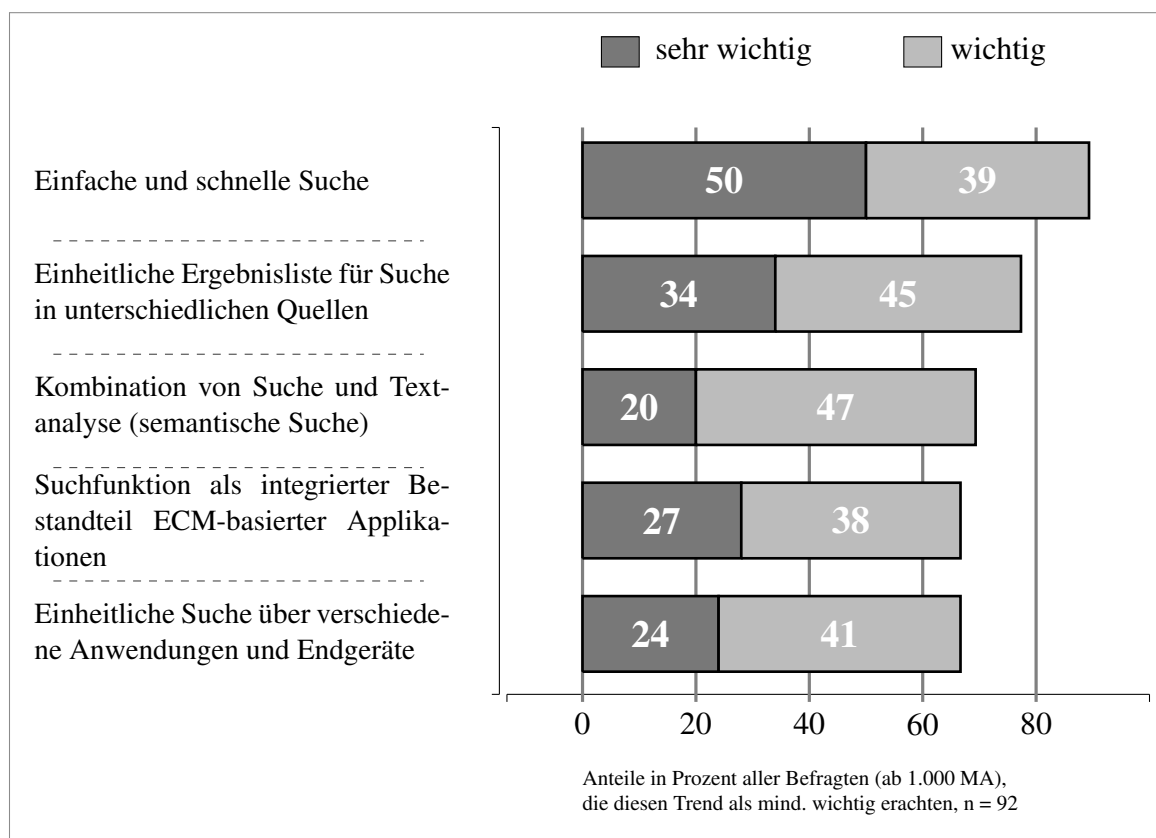


Abbildung 1.1.: Relevanz „Effizienter Suche“ für Unternehmen [Nie12]

Die Umfrage des Lehrstuhls für Produktentwicklung der Technischen Universität München in Zusammenarbeit mit dem Verband Deutscher Maschinen- und Anlagenbau (VDMA) im Jahr 2008 ergab, dass Informationsbeschaffung heutzutage überwiegend auf Basis der Volltextrecherche erfolgt und folglich die Anforderungen der Anwender nur zum Teil oder gar nicht

erfüllt [GPL08]. Damit sich also ein Anwender in einem Suchraum zurechtfinden kann, muss a priori eine klare Vorstellung vom Ergebnis gegeben sein, um danach in Form von Suchbegriffen suchen zu können. Folglich setzt eine gezielte Informationsrecherche voraus, dass der Anwender eigenes persönliches Wissen in die Suchanfrage investiert, weil die gegenwärtig in der Produktentwicklung eingesetzten Systeme kein Wissen vermitteln, sondern nur dessen Rohstoff in Form von Informationen bereitstellen.

Aus diesen Erkenntnissen ergibt sich der Bedarf, semantische Technologien einzusetzen, die Informationsstrukturen und Zugriffsmechanismen bereitstellen, mittels derer aus heterogenen Datenquellen gezielt relevante Informationen ausgewählt und dem Anwender zum richtigen Zeitpunkt am richtigen Ort zur Verfügung gestellt werden. Die Aufgabe der Wissensrepräsentation ist es dabei, den Zugriff auf eine Reihe bestehender Informationssysteme durch die genaue Spezifikation eines Diskursbereichs mittels eines formalen Modells sicherzustellen. Das dazu notwendige formale Modell eines Anwendungsbereichs stellt einerseits eine einheitliche Sicht auf die Datenquellen, andererseits wird durch logische Inferenz die semantische Heterogenität überwunden [LN07]. Hierfür müssen Systeme aufgebaut werden, die eine solche einheitliche Sicht herstellen und als Basis für alle weiteren, über den gesamten Produktentwicklungsprozess verteilten Aktivitäten dienen. Dabei geht es nicht nur um die Entwicklung von Software, was ohnehin ein extrem kostspieliges Unterfangen ist, sondern um die Sicherstellung, dass diese neuen Systeme mit den bereits vorhandenen zusammenarbeiten. Genau das erfordert die Informationsintegration. Es existieren zahlreiche Schätzungen, die besagen, dass heute große Unternehmen viel Zeit und finanzielle Mittel für Informationsintegration investieren. BERNSTEIN und HAAS legen dies wie folgt aus:

Large enterprises spend a great deal of time and money on „information integration“ - combining information from different sources into a unified format. Frequently cited as the biggest and most expensive challenge that information-technology shops face, information integration is thought to consume about 40 % of their budget. [BH08]

OLOFSON ist ähnlicher Meinung. In seiner IDC-Studie zeigt er (Abbildung 1.2), dass der Markt für Software zur Informationsintegration von 2,5 im Jahr 2007 auf 3,8 Milliarden US-Dollar in 2012, also um 8,7 % p. a. gewachsen ist [Olo08].

Zwei Ansatzpunkte bieten sich an, um diesen Zustand zu verbessern: Einerseits kann durch den Entwurf standardisierter Datenmodelle der Zugriff auf gemeinsam genutzte Informationen gewährleistet und vereinfacht werden; andererseits kann die Spezifikation der Programmschnittstellen zur einheitlichen Informationsbeschaffung beitragen. Beide Ansätze haben jedoch aufgrund der eingesetzten Systeme den entscheidenden Nachteil, dass die zu integrierenden Sys-

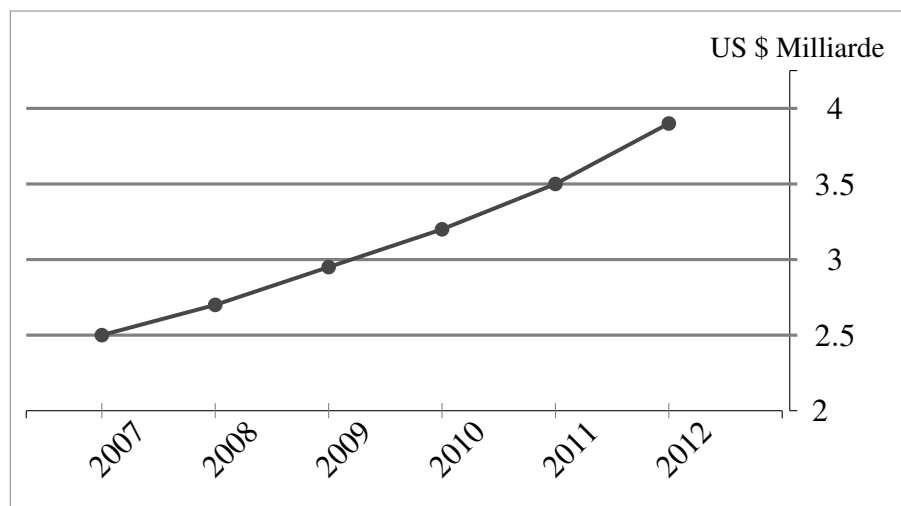


Abbildung 1.2.: Marktzuwach für Software zur Informationsintegration [Olo08]

teme mit einer Vielzahl unterschiedlicher Datenmodelle und Zugriffsschichten bereits existieren und darüber hinaus nicht mehr verändert werden sollen. Außerdem werden fortlaufend neue proprietäre Datenmodelle in neuen Applikationen aufgebaut [OBB10].

Genau hier setzt die vorliegende Arbeit an. Es wird ein Konzept vorgestellt, das Semantic-Web-Technologien zur Wissensmodellierung basierend auf der Nutzung von Ontologien einsetzt. Eine Ontologie legt dabei eine gemeinsame Sprache und einen gemeinsamen Wortschatz für die Interaktion zwischen den eingesetzten heterogenen verteilten Systemen fest. Darüber hinaus ist es möglich, logische Schlüsse mithilfe von Inferenzmechanismen zu ziehen. Die Zielsetzung der Ontologie ist die Unterstützung einer Anwendung, die eine Suche nach Produktinformationen für den Unternehmensbereich „Produktentwicklung“ erleichtern und damit die interaktive Informationsbeschaffung in diesem Anwendungsgebiet vereinfachen soll. Folglich müssen Datenmodellbeschreibungen für existierende Systeme entwickelt, in einer Ontologie verfügbar gemacht, durch zwischengeschachtelte Formalismen auf ein formales konzeptuelles Modell abgebildet und dadurch gekapselt werden. Mit diesem Ansatz können die operationellen Datenbanken der verwendeten Systeme weiterhin unverändert eingesetzt werden.

Im Mittelpunkt dieser Arbeit stehen daher Ansätze, die Vision semantischer Technologien in Bezug auf die Informationsintegration Realität werden zu lassen. Konkret werden ein Konzept zur semantischen Integration und ein Rahmenwerk zu dessen Realisierung vorgestellt. Das Ziel dieser Integration ist es, die Interoperabilität heterogener Informationssysteme mit großen Datenbeständen zu gewährleisten und die Informationsbeschaffung durch Formulierung kontextbezogener Aufgaben- oder Problemstellungen zu ermöglichen. Übergeordnetes und daher visionäres Ziel ist es, Entscheidungsprozesse wissensbasiert auszuführen.

1.2. Zielsetzung der Arbeit

Gegenstand dieser Arbeit ist die Entwicklung eines Konzepts für semantische Informationsintegration zur Gewährleistung der Interoperabilität von heterogenen, verteilten Informationssystemen im Anwendungsumfeld unternehmensübergreifender Produktentwicklung. Bei der Entwicklung des Konzepts sollen sowohl der Aspekt *Formalisierung konzeptueller Strukturen* als auch der Aspekt *Wissensvermittlung* mittels Informationsbereitstellung berücksichtigt werden. Der Fokus liegt erstens auf der Umsetzung notwendiger Systemumgebung mittels Softwareprototypen, zweitens auf der Veranschaulichung des praktischen Nutzens am Beispiel eines industriellen Anwendungsszenarios im Bereich Maschinenbau.

Im einzelnen verfolgt diese Arbeit folgende Ziele:

- Zunächst sollen die Grundprinzipien, Eigenschaften und Probleme der auf Beschreibungslogiken basierenden Wissensrepräsentation beschrieben werden, um diese auf das Anwendungsumfeld *Produktentwicklung* anwenden zu können.
- Nachfolgend soll die Bedeutung von Informationsintegration im Allgemeinen und in der Produktentwicklung veranschaulicht werden.
- Ausgehend von den zuvor gewonnenen Erkenntnissen soll ein Konzept entwickelt werden, mit dem semantische Informationsintegration in der Produktentwicklung erzielt werden kann.
- Um eine Realisierung des Konzepts zu ermöglichen, sollen ein Wissensrepräsentationsformalismus entwickelt, ein Mappingverfahren erarbeitet und ein mathematisches Modell aufgestellt werden.
- Die Tragfähigkeit des entwickelten Konzepts und dessen Nutzenpotenzial sollen in einer prototypischen Implementierung gezeigt werden.

Durch das entwickelte Konzept bzw. darauf basierender Softwarelösung sollen folgende Vorteile erzielt werden:

- Abbildung konzeptueller Strukturen mittels globaler Ontologie. Hierbei werden unterschiedliche Begriffswelten der zu integrierenden Datenquellen und deren korrespondierenden Bereiche als gleichwertig beschrieben und miteinander semantisch verknüpft.
- Verringerter Aufwand, um weitere Datenquellen zu integrieren. Dies soll insbesondere durch die modularisierte Architektur und Wiederverwendung von konzeptuellen Strukturen erreicht werden.

- Wissensrepräsentation auf Basis beschreibungslogischer Formalismen ermöglicht eine benutzerzentrische Sicht auf die Inhalte der Wissensbasis. Dadurch werden sämtliche Kontextinformationen aus einer Vielzahl von Informationssystemen durch ein gemeinsames und dem Anwender vertrautes Vokabular beschrieben.
- Die semantische Suche nach Wissensressourcen verbessert deutlich die Qualität der Suchergebnisse als herkömmliche Systeme. Dies führt letztendlich zu einer Steigerung der Mitarbeiterproduktivität.

Diese Arbeit schildert demnach die theoretischen und praktischen Erfahrungen des Autors bei der Entwicklung und Realisierung der semantischen Informationsintegration, einschließlich seiner eigenen Implementierungen.

Hauptsächlich wird aber von diesen Erfahrungen abstrahiert und ein allgemeiner Lösungsansatz vorgestellt, der mit seinen Algorithmen und Werkzeugen ein generelles Konzept zur semantischen Informationsintegration in Produktentwicklung bereitstellt.

1.3. Vorgehensweise und Struktur der Arbeit

Ausgehend von der in Abschnitt 1.2 beschriebenen Zielsetzung werden in Kapitel 2 zunächst die wichtigsten Begriffe und Grundlagen zur semantischen Informationsintegration erläutert. Hierfür wird in das Gebiet der Informationsintegration sowie der formalen Wissensrepräsentation eingeführt. Den Ausführungen vorgreifend soll an dieser Stelle festgehalten werden, dass die deklarative Wissensrepräsentation auf Basis von Beschreibungslogiken einen guten Kompromiß bezüglich der Ausdrucksstärke der Sprache und der Komplexität von Inferenzmechanismen darstellt und somit einen Erfolg versprechenden Formalismus für die Informationsintegration in der Produktentwicklung bildet.

Daran anschließend wird im Kapitel 3 der Stand der Technik und Forschung, insbesondere Formen und Lösungen zur informationstechnischen Integration von Informationssystemen betrachtet. Bestehende Ansätze zur Informationsintegration werden nachfolgend vor dem Hintergrund der oben beschriebenen Ausgangssituation und Zielsetzung untersucht und bewertet. Der Abschnitt 3.4 fasst die Defizite heutiger Lösungen zusammen.

Die hieraus gewonnenen Ergebnisse bilden die Grundlage für die Ableitung der Anforderungen an ein System zur ontologiebasierten Informationsintegration. Darauf basierend werden in Kapitel 4 die Ziele und Anforderungen für das zu entwickelnde Konzept definiert.

In Kapitel 5 wird ein Konzept für ein System zur semantischen Informationsintegration erarbeitet. Dazu wird ein Formalismus zur vollständigen Beschreibung von Informationen der zu

integrierenden Datenquellen vorgestellt. Des Weiteren werden Verfahren zum Mapping von Ontologiemodellen und heterogenen verteilten Datenquellen entwickelt. Im nächsten Schritt wird ein mathematisches Modell als Gegenstand des Integrationsformalismus erarbeitet. Basierend auf dem Wissensrepräsentationsformalismus und auf dem mathematischen Modell wird die Anfragesprache sowie die Algorithmen entwickelt, um logische Schlüsse ziehen zu können.

Ausgehend von dem in Kapitel 5 entwickelten Konzept erfolgt in Kapitel 6 auf Basis eines Anwendungsszenarios die Realisierung und Validierung des Ansatzes zur semantischen Informationsintegration in Produktentwicklung.

In Kapitel 7 werden die Ergebnisse dieser Arbeit zusammengefasst. Der Ausblick zeigt Anknüpfungspunkte für zukünftige Arbeiten. Die Abbildung 1.3 zeigt die vorliegende Arbeit in ihrer Struktur.

Allgemeine Gestaltungsrichtlinien

In der Arbeit werden folgende Textformatierungen zur besseren Übersicht verwendet:

- *kursive Schrift* zur Einführung neuer Begriffe oder zur Hervorhebung bestimmter Sachverhalte,
- **Fettschrift** zur besseren Gliederung von Textteilen,
- serifenlose Schrift zur Beschreibung von Konzept- und Rollennamen der Ontologie, sowie von Quellcodes,
- Schreibmaschinenschrift zur Darstellung von Dateinamen und primitiven Datentypen,
- KAPITALSCHRIFT zur Kennzeichnung von Autorennamen,
- Anführungszeichen zur Markierung von besonderen Textstellen,
- Zitate sind zur Ergänzung der Darlegung als eigener, vom linken und rechten Seitenrand zusätzlich eingerückter Absatz in kursiver Schrift eingefügt.

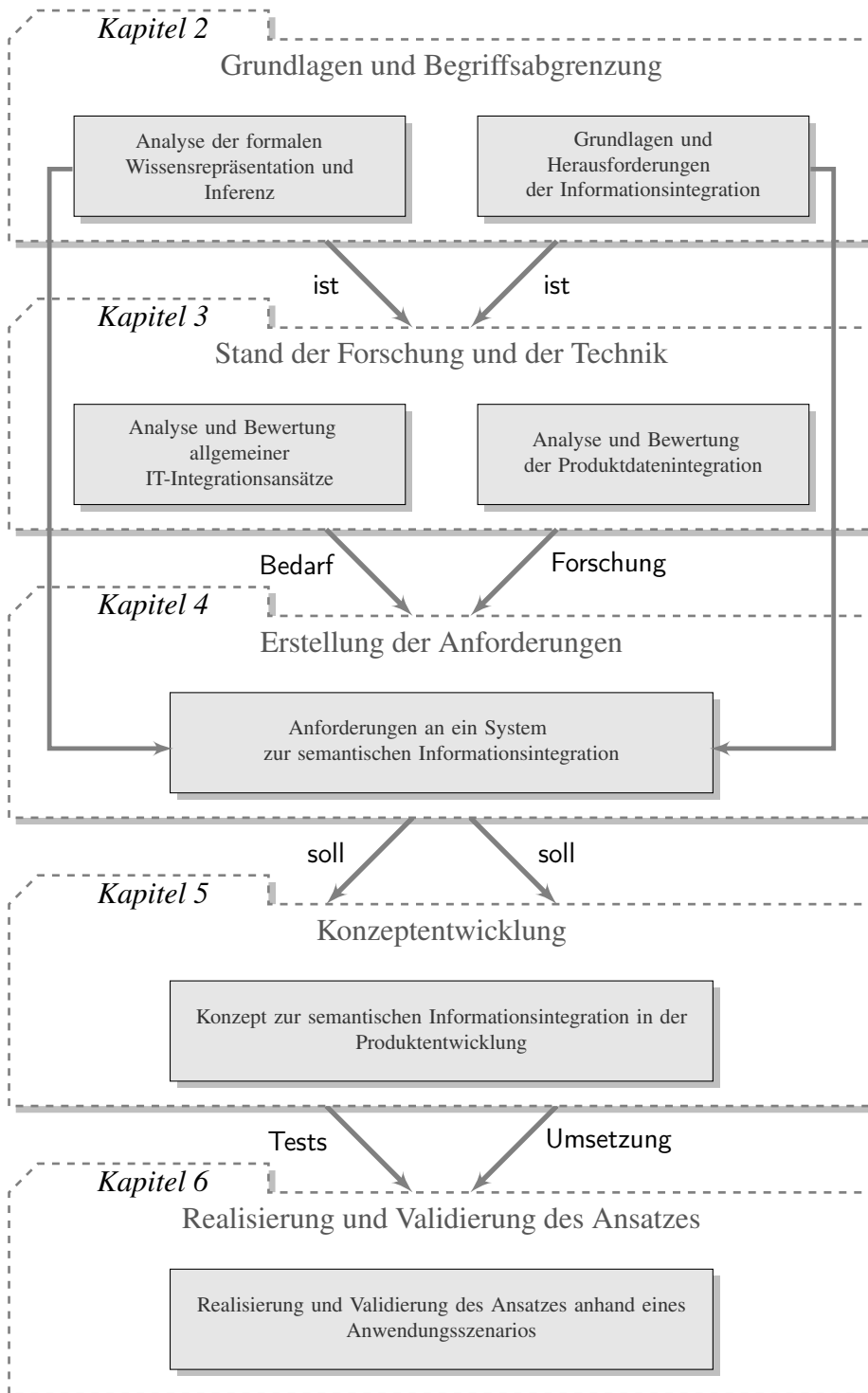


Abbildung 1.3.: Struktur der vorliegenden Arbeit

Man hat nicht die Weisheit im Sack, man muß sie sich mühsam erarbeiten.

HANNS EISLER (1898–1962)

2. Grundlagen und Begriffsabgrenzung

Für das Verständnis der vorliegenden Arbeit ist die Erläuterung der wichtigsten Begriffe und Grundlagen zur semantischen Informationsintegration für die Produktentwicklung notwendig. Die zu erläuternden Grundlagen dienen dazu, das Untersuchungsfeld der Arbeit zu analysieren und zu beschreiben sowie eine Eingrenzung der betrachteten Aspekte vorzunehmen. Ziel ist es, den Einflussfaktor von Beschreibungslogiken auf eine semantische Informationsintegration darzustellen. Die Basis dieses Abschnitts sind Aspekte zur Sicherstellung der semantischen Interoperabilität durch das Integrationswerkzeug und Instrumente des Wissensmanagements. Dieser Abschnitt fokussiert deshalb die Integrationsansätze, die Informationen in einem Kontext sehen und durch formale Wissensrepräsentation, die auf Beschreibungslogiken basiert, erweiterbar sind, sodass die Interoperabilität beteiligter Systeme gewährleistet werden kann.

2.1. Begriffsbestimmung

Die Begriffe, die in folgendem Abschnitt erläutert werden, werden häufig und in unterschiedlichen Zusammenhängen verwendet, sodass ihre Bedeutung vorab und übergreifend geklärt werden muss.

2.1.1. Daten, Information und Wissen

Wissen ist ein häufig benutzter Begriff. Seine genaue Bedeutung ist nur unzureichend bekannt. Eine Abgrenzung der Begriffe *Wissen* und *Informationen* ist erforderlich, da beide Begriffe häufig fälschlicherweise verwendet werden. Im Allgemeinen baut der Wissensbegriff auf den Begriffen *Daten* und *Informationen* auf. So definiert die VDI-Richtlinie 5610 Daten als objektive Fakten, die ohne Zusammenhang und weitere Hintergründe nicht deutbar sind [VDI09]. Sie sind als „Rohmaterial“ zu verstehen [SZ03]. Daten beinhalten lediglich einen syntaktischen Aspekt, z. B. „1“, „P“, „M8“. Es sind Zeichen vom Typ „Zahl“, „Buchstabe“ oder Zeichenketten als Mischformen aus beiden. Informationen – so weiter die VDI-Richtlinie – sind strukturierte Daten mit Relevanz und Zweck, die „in einen Kontext gebracht, kategorisiert, kalkuliert werden können“ [VDI09]. Demnach beinhalten Informationen Syntax und Semantik (Form und Inhalt) wie z. B. M8 = Gewinde-Nenndurchmesser und schränken somit den Interpretationsspielraum der Daten ein. Wissen ist letztendlich vernetzte Informationen, die es ermöglichen, Vergleiche anzustellen, Verknüpfungen herzuleiten und Entscheidungen zu treffen [VDI09]. Ein Beispiel: die Verwendung einer M12-Schraube (Information) anstatt einer M8-Schraube (Information)

erlaubt die Erhöhung der statischen und dynamischen Betriebskraft bei gleichbleibender Festigkeitsklasse (Wissen). Aus Informationen wird erst dann Wissen, wenn die anwendungs- oder situationsbezogene Bedeutung der Informationen erkannt wird und wenn aus den vielen strukturierten sowie unstrukturierten Informationen die relevanten herausgefiltert und in einer bedeutungsgerechtern Weise strukturiert werden [AS99], [PRR06]. Der „Mehrwert“ von Wissen im Vergleich zu reinen Daten und Informationen entsteht durch die Erfahrungen und durch das Gelernte. Darüber hinaus können diese Informationen in Zusammenhang mit eigenen Erfahrungen wieder neues Wissen generieren. Die VDI-Richtlinie verdeutlicht diesen Sachverhalt beispielhaft anhand des Wirkungsgrads einer Maschine in Abbildung 2.1.

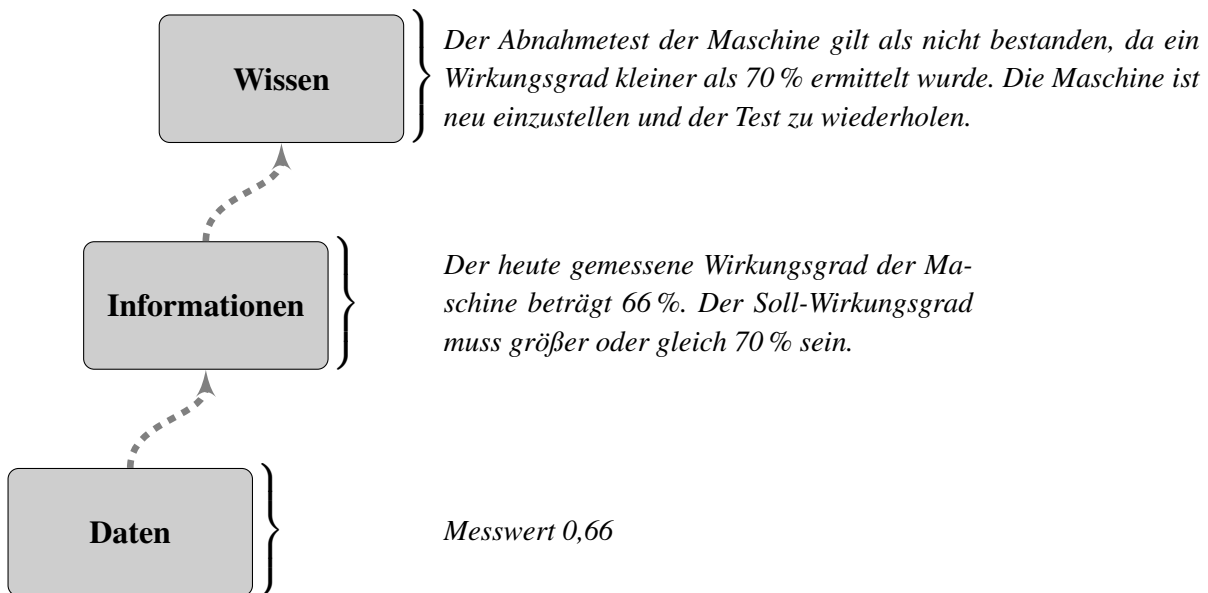


Abbildung 2.1.: Zusammenhang zwischen Daten, Informationen und Wissen nach [VDI5610]

„Wissen“ wird in dieser Arbeit als das Ergebnis eines Transformationsprozesses verstanden, an dessen Beginn unstrukturierte, isolierte und weitgehend kontextunabhängige Daten stehen. Durch die Analyse dieser Daten und die Berücksichtigung ihrer Bedeutung gelingt es, Informationen zu gewinnen. Die Vernetzung und Interpretation dieser Informationen in einem spezifischen Kontext führt schließlich zum Wissen. Die Art der Vernetzung ergibt sich einerseits aus dem Kontext, d. h. aus der begrifflichen Verbindung, in der eine Information steht, und dem Sach- und Situationszusammenhang, aus dem heraus diese Information verstanden werden soll. Der Bedeutungsgehalt ist somit auf den Kontext der Situation bezogen, hängt folglich vom Anwendungszweck ab und unterliegt ebenso subjektiven Einflussfaktoren.

2.1.2. Wissensarten

Die Einteilung von Wissen in verschiedene Wissensarten kann nach unterschiedlichen Kriterien vorgenommen werden [Rud98]. Im Rahmen dieser Arbeit sind folgende Wissensarten relevant:

Wissensart	Kriterium	Erklärung
Erscheinungsform	<i>explizites Wissen</i>	Wissen ist auf verschiedenen Ebenen formalisierbar (z. B. Sprache, Schrift), damit kommunizierbar und in verschiedenen Medien speicherbar.
	<i>implizites Wissen</i>	Wissen ist an Personen gebunden, schwer kommunizierbar und kaum formalisierbar.
Inhalt	<i>deklaratives Wissen</i> (statisches Wissen)	Beschreibungswissen für beliebige Objekte und deren Eigenschaften innerhalb eines bestimmten Anwendungsbereiches. Dieses Wissen kann hierarchisch oder netzwerkartig strukturiert sein.
	<i>prozedurales Wissen</i> (dynamisches Wissen)	Wissen ist methoden- oder ablauforientiert, d. h. es beschreibt Folgen von Arbeitsschritten (z. B. Algorithmus, Verfahren etc.). Prozedurales Wissen bezieht sich immer auf eine konkrete Aufgabe und umfasst neben der Aufgabenspezifikation alle Informationen zur Lösungsfindung.
pragmatische Kategorie	<i>terminologisches Wissen</i>	Dadurch wird die Gesamtheit der Begriffsbildung innerhalb einer Anwendungsdomäne, die Relationen zwischen den Begriffen sowie deren hierarchische Gliederung beschrieben.
	<i>Faktenwissen</i>	Dadurch werden Gegenstände, Erscheinungen und Zusammenhänge, die objektiv sind, beschrieben. Die Fakten sind atomar, d. h. nicht aus anderen Fakten zusammengesetzt.

Tabelle 2.1.: Wissensarten [Rud98]

2.1.3. Produktinformation/Produktwissen

Ein Produkt wird beschrieben durch Produktmerkmale, die durch den Produktentstehungsprozess festgelegt werden. Nach der DIN-Norm 2330 [DIN79] können Beschaffenheitsmerkmale, Funktionsmerkmale und Relationsmerkmale unterschieden werden. *Beschaffenheitsmerkmale* geben Eigenschaften an, die am Produkt selbst festgestellt werden können (z. B. Gestalt, Lage,

Verbindungsart, Werkstoff). *Funktionsmerkmale* beschreiben den gewollten Zweck des Produkts (z. B. Stoff transportieren, Energie umformen). *Relationsmerkmale* umfassen Eigenschaften des Produkts, die erst in Relation zu anderen Größen relevant werden (z. B. ergonomische Eigenschaften, die die Beziehung zum Bediener beschreiben, oder der kalkulierte Preis, der eine Beziehung zum Markt zeigt) [GRG98].

Diese Art der Beschreibung eines Produkts ist im Sinne der Rechnerverarbeitung ungeeignet. Eine bessere Darstellung sieht eine Unterteilung in Produktinformationen und Produktwissen vor, die in Anlehnung an die Begriffe „Daten“ und „Wissen“ hinsichtlich VDI 5610 [VDI09] definiert werden können.

Definition 1: *Produktinformationen sind implizite oder explizite Merkmale eines Produkts, die in einer Datenbank gespeichert und abgerufen werden können. Sie entsprechen somit den Produktmerkmalen. Implizite Merkmale sind Merkmale, die aus expliziten und anderen impliziten Merkmalen ermittelt werden können. Explizite Merkmale sind Merkmale, die ohne weitere Verknüpfungen oder Transformationen direkt abrufbar sind.*

Definition 2: *Produktwissen ist Produktinformation mit impliziten und expliziten Angaben über seine Verwendung zur Entwicklung von technischen Produkten.*

Definition 3: *Unter Produktentwicklungswissen wird das Wissen verstanden, das benötigt wird, um eine Produktentwicklungsaufgabe erfolgreich zu lösen.*

Produktwissen unterscheidet sich von Produktinformation dadurch, dass es mit Anleitung über seine Verwendung gekoppelt ist. Damit ist gleichzeitig festgelegt, dass der Übergang von Produktinformation zu Produktwissen fließend ist. Produktwissen ist eine Teilmenge des Produktentwicklungswissens.

2.2. Informationsintegration

Informationstechnische Unterstützungssysteme bilden das Rückgrat zahlreicher Prozesse innerhalb eines Unternehmens, beginnend bei der Bürotechnik- und -kommunikation über kaufmännische Anwendungen bis zu den technischen Anwendungen in Entwicklung, Konstruktion, Arbeitsvorbereitung und Fertigung [KFG07]. Gerade im Bereich produzierender Unternehmen ist die Informationsintegration ein zentraler Faktor, da deren Geschäftstätigkeiten durch ein hohes Maß an Engineering und Zusammenarbeit vieler Marktteilnehmer auf verschiedenen Wertschöpfungsstufen gekennzeichnet sind [Eng05]. Darüber hinaus ist die Informationsintegration stets dann relevant, wenn zahlreiche heterogene Datenbestände existieren und eine integrierte Sicht auf die Informationen anstrebenswert ist. Dieser Sachverhalt kommt in der Produktentwicklung vor, wenn einzelne Anwendungssysteme zur Unterstützung des Produktlebens mit-

einander interagieren.

Im Folgenden werden zunächst die Grundlagen der Informationsintegration betrachtet, die vorwiegend im Zusammenhang mit föderierten Datenbanksystemen entwickelt wurden. Ziel dieser Betrachtung ist die Identifizierung von grundlegenden Herausforderungen, die bei der Integration mehrerer Datenbestände entstehen.

2.2.1. Grundlagen

Die Vision eines Informationsaustauschs zwischen beteiligten Informationssystemen innerhalb eines Unternehmens oder einer Organisationseinheit, ohne dass diese Systeme eng miteinander gekoppelt sein müssen, ist nicht neu. Erste Versuche auf diesem Gebiet fanden Mitte der 70er-Jahre statt, als man grundlegende Entwicklungen im Bereich verteilter Datenbanksysteme unternommen hat [CHKT06], [SSH10]. Dabei wurden zwei Arten der Informationsintegration unterschieden: *materialisierte Integration* und *virtuelle Integration*. Die Abbildung 2.2 gibt eine allgemeine Zusammenfassung der Ansätze zur Informationsintegration wieder.

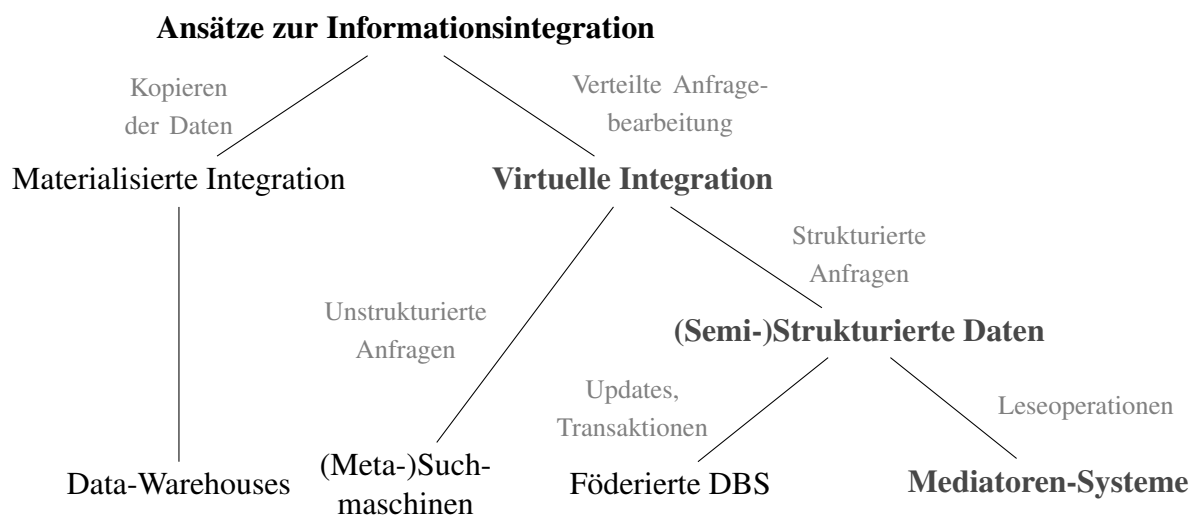


Abbildung 2.2.: Ansätze zur Informationsintegration: virtuell vs. materialisiert nach [Kud07]

Man verfolgte das Ziel, eine Menge autonomer Datenquellen so zu vereinen, dass diese nach außen als ein System mit einem integrierten Datenbestand wirkten. Eine vereinfachte Architektur eines solchen Integrationssystems auf Basis eines Drei-Schichten-Modells mit entsprechenden verteilten Datenträgern ist in der Abbildung 2.3 dargestellt.

Hierbei handelt es sich um eine *virtuelle* Integration, die im Gegensatz zu materialisierter Integration ihren zentralen Datenbestand in virtueller und nicht in einer physischen Form enthält. Die Daten werden also nicht zentral in einer Datenbank gehalten, sondern alle Anfragen werden an das Integrationssystem gestellt und von dort aus in Anfragen an die einzelnen Datenquellen überführt. Somit werden die Daten nur in Teilmengen während der Anfragebearbeitung von

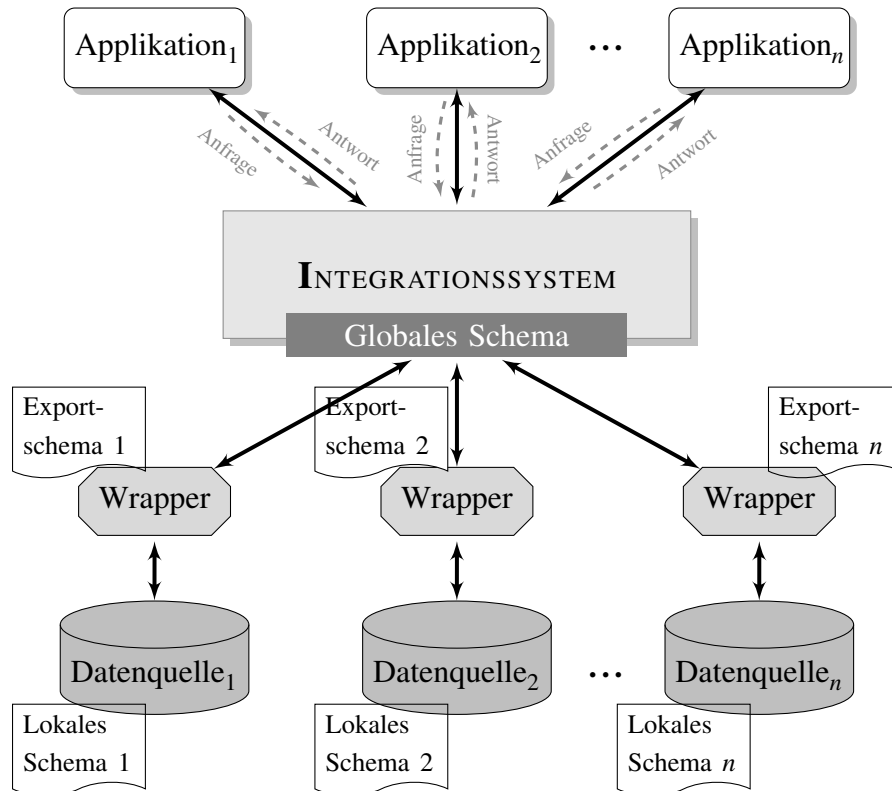


Abbildung 2.3.: Vereinfachte Architektur einer mediatorbasierten Informationsintegration

den Datenquellen zum Integrationssystem transportiert und nach der Berechnung der Antwort wieder verworfen. Der integrierte Informationsbestand existiert damit nur virtuell. Zwar liegt aus Anwendersicht (Applikationssicht) ein homogener Bestand vor, tatsächlich aber findet die Integration dynamisch bei jeder neuen Anfrage statt. Die grundlegende Integrationsproblematik ist aber bei beiden Vorgehensweisen vergleichbar ähnlich-, unabhängig davon, ob die Daten in einer Datenbank materialisiert (engl. *data warehouse*) oder mittels virtuellen Ansatzes in einem Mediator-basierten Informationssystem integriert werden.

Bei der Informationsintegration ist es erforderlich, mehrere Schemata in Betracht zu ziehen bzw. zueinander in Beziehung zu setzen. Einerseits sind es lokale Schemata der Informationsquellen, die a priori als gegeben zu betrachten sind und nicht geändert werden dürfen. Jeder Informationsquelle gehört ein *Wrapper*-Programm an, das dazugehöriges lokale Schema in ein Exportschema überführt. Die Exportschemata der Quellen sind dazu da, um eine Teilmenge der für das globale Schema relevanten Daten in einem einheitlichen Datenformat abzubilden. Andererseits setzt das globale Schema die Exportschemata der einzelnen Datenquellen zusammen. Folglich wird das globale Schema auch als *integriertes* Schema bezeichnet [Con02]. Der Vorgang zum Entwurf eines globalen Schemas wird *Schemaintegration* genannt [CHKT06]. Bei der Durchführung der Schemaintegration treten mehrere Integrationskonflikte auf, die zu lösen sind [HST99], [ALSS03], [AKS05], [LN07]:

Beschreibungskonflikte liegen vor, wenn gleiche Objekte in den Schemata mittels unterschiedlicher Eigenschaften beschrieben werden. So kann beispielsweise die Darstellung einer Typvariante eines Fahrzeugs entweder durch ein Attribut oder anhand mehrerer Attribute (z. B. Hubraum, Leistung, Getriebe etc.) zusammengestellt werden. Unter die Kategorie der Beschreibungskonflikte fallen unter anderem Homonyme, Synonyme, unterschiedliche Attribute, unterschiedliche Einheiten, unterschiedliche Wertebereiche etc.

Extensionale Konflikte treten auf, wenn zwei Schemata die gleichen Objekte der realen Welt modellieren, die dazugehörigen Datenbestände (die *Extensionen*) jedoch unterschiedlich sind. Damit extensionale Konflikte vorliegen, müssen die Extensionen der zu integrierenden Datenbanken in einer Teilmengenbeziehung zueinander stehen, sich überlappen oder sie können disjunkt sein.

Strukturelle Konflikte entstehen, wenn innerhalb desselben Datenmodells unterschiedliche Modellierungskonzepte verwendet wurden, um gleiche Ausschnitte der realen Welt zu modellieren. So kann z. B. ein Objekt einerseits mittels einer Datenbankrelation, andererseits durch ein Datenbanktupel modelliert werden.

Heterogenitätskonflikte liegen vor, wenn die zu integrierenden Schemata mittels unterschiedlicher Datenmodelle beschrieben sind (z. B. relational, objektorientiert, XML).

In Bezug auf die Mediator-basierte Architektur sind Heterogenitätskonflikte dann gelöst, wenn das Integrationssystem und die Datenquellen dasselbe Datenmodell verwenden. Strukturelle Konflikte und Beschreibungskonflikte sind gelöst, wenn semantisch identische Konzepte auch strukturell gleich modelliert wurden und wenn beide – sowohl das Integrationssystem als auch die Datenquelle – unter der verwendeten Bezeichnung für Schemaobjekte auch tatsächlich dasselbe bedeuten. Da die Datenquellen häufig niemals vollständig sind, enthält die Extension eines lokalen Schemas nur eine Teilmenge der Extension des globalen Schemas. Daher sollen die Extensionen aus den unterschiedlichen Quellen so zusammengeführt werden, dass alle für das gleiche reale Objekt zu einer Extension zusammengefasst werden.

In einem virtuell integrierten System erfolgen Anfragen an ein globales Schema, dessen Datenbestand aber nur virtuell existiert. Die Schemaintegration ist also ein wesentlicher Bestandteil der Informationsintegration. Die Beziehungen zwischen Datenquellen und dem globalen Schema müssen daher durch Korrespondenzen (eng. *correspondence assertions*) spezifiziert werden [SPD92]. Zur Anfragezeit ist es notwendig, die globale Anfrage in einzelne Anfragen, sogenannten Anfragepläne, zu zerlegen, die die benötigten Daten aus den verschiedenen Quellen holen, vereinen und transformieren [LN07]. Darüber hinaus müssen bei der Ausführung der Pläne besondere Kostenfaktoren berücksichtigt werden, die durch die Übertragung von relevanten Daten über ein Netzwerk entstehen. Des Weiteren soll der Fall einbezogen werden, dass

Quellen vorübergehend nicht verfügbar sein können.

Ferner ergeben sich folgende Aufgaben, die ein Informationintegrationssystem zu behandeln hat [CGL⁺09]:

- Bestimmung der Datenquellen, in denen die relevanten Informationen zu finden sind
- Zerlegung einer Anfrage an das globale Schema in einzelne Anfragen an die lokalen Datenquellen
- Interpretation und Zusammenführung der Daten, die von den lokalen Datenquellen stammen

Zum ersten Punkt muss ergänzend festgestellt werden, welche Teile einer globalen Anfrage durch welche Datenquellen beantwortet werden können. Zur Lösung dieses Problems sind Anfragekorrespondenzen zu spezifizieren, die die semantischen Beziehungen zwischen Elementen verschiedener Schemata modellieren. Als Elemente kommen einzelne Attribute, Relationen oder Anfragen in Frage, wobei die Art der semantischen Beziehungen typischerweise die extensionale Inklusion ist. Ferner handelt es sich hierbei um ein Modellierungsproblem.

Beim zweiten Punkt liegt das Problem darin, dass es oftmals mehrere Möglichkeiten gibt, eine globale Anfrage zu zerlegen, wodurch für jede Teilanfrage wiederum mehrere mögliche Datenquellen infrage kommen. Daraus ergibt sich in den meisten Fällen eine Vielzahl von logisch äquivalenten, aber unterschiedliche Ergebnisse generierenden Anfrageplänen. Letztendlich müssen die Teilantworten nach der Ausführung verschiedener Anfragepläne zu einer homogenen Antwort auf die ursprüngliche globale Anfrage zusammengeführt werden.

2.2.2. Herausforderungen der Informationsintegration

Die im vorhergehenden Abschnitt aufgezeigten Probleme der Schema-Modellierung und der Anfragenbehandlung sind nicht die einzigen, die die Informationsintegration erschweren. Ein grundlegendes Problem ist hierbei die Heterogenität der Systeme. Wir bezeichnen zwei Systeme als *heterogen*, wenn diese nicht die exakt gleichen Methoden, Modelle und Strukturen zum Zugriff auf ihre Daten anbieten [LN07]. Folglich muss jedes Informationssystem über ein spezielles Wrapper verfügen, das die Daten aus den Quellen extrahiert und in ein einheitliches Format übersetzt. Des Weiteren liegt die Heterogenität dann vor, wenn das Integrationssystem Zugriffe auf die integrierten Daten mittels bestimmter Anfragesprache gestattet, gleichzeitig aber Quellen beinhaltet, auf die nur über Formulare zugegriffen werden kann [Haa07], [Hal09]. Das Hauptaugenmerk der Forschung im Bereich der Informationsintegration liegt auf der Fragestellung nach dem Modellierungsaspekt und der Anfragebearbeitung (engl. *query processing*). Im Folgenden sollen diese Problemstellungen genauer betrachtet werden; einen vollständigen Überblick über die Thematik liefern [Hal01], [Len02], [Hal09] und [ÖV11].

Modellierungsaspekt

Das Grundgerüst eines Informationsintegrationssystems (IIS) besteht aus (1) einer globalen Sicht (engl. *global view*) zur Darstellung der Anwendungsdomäne, (2) einer Menge von Informationsmodellen (engl. *information source models*) zur Repräsentation der Datenstruktur in relevanten Quellen und schließlich (3) einer Menge semantischer Korrespondenzen (engl. *mappings*), um die globale Sicht mit Informationsmodellen der Quellen in Beziehung zu setzen [Len02]. Hierbei fokussiert die Modellierung sowohl die Frage nach der Darstellung einzelner Schemata als auch die Art der Formalisierung der Beziehungen zwischen diesen.

Die weitverbreiteten Darstellungsformen der Schemata sind relationale oder objektorientierte Datenmodelle. Beim relationalen Ansatz wird eine Erweiterung von Datalog¹ verwendet, um die Schemata und ihre Beziehungen untereinander zu beschreiben [KLSS], [GKD97].

BERGAMASCHI ET AL. benutzen objektorientierte bzw. semistrukturierte Datenmodelle, um die Heterogenität der Quellen zu beherrschen. Hierzu wird eine Erweiterung der Sprache ODL² zur Beschreibung der Schemastrukturen eingesetzt [BCVB01]. Neuere Ansätze nutzen auf Beschreibungslogiken basierende Ontologien, um eine konzeptionelle Darstellung der Datenquellen und der globalen Sicht abzubilden [CGLR08], [CDGL⁺09], [MST⁺10], [RMC11].

Die Entwicklung einer IIS-Architektur sieht die Definition von Beziehungen bzw. Abhängigkeiten zwischen Datenquellen und dem Modell der Anwendungsdomäne vor. Dabei sind zwei Hauptansätze zur Modellierung der Abhängigkeiten in der Literatur bekannt [Ull00], [Hal01]: Der *Global-as-View* (GaV)-Ansatz definiert das globale Schema bzw. die Sicht als eine Menge der Anfragen über darunterliegenden Quellen. Folglich wird eine Relation des globalen Schemas als Vereinigung mehrerer Relationen der Quellen beschrieben. Der *Local-as-View* (LaV)-Ansatz definiert hingegen die Schemata der Datenquellen als Sicht auf das globale Schema in Form einer Menge der Anfragen über die globale Sicht. Folglich kann eine Relation in einer Datenquelle als nur ein Teil einer Relation des globalen Schemas beschrieben werden.

Um den Unterschied der beiden Ansätze bezüglich der Beziehungsfestlegung besser erläutern zu können, wird im Folgenden eine formale Definition des zentralisierten Informationsintegrationssystems eingeführt. Diese Definition basiert auf der Arbeit von M. LENZERINI [Len02].

¹*Datalog* (*Database + Logic*) Eine deklarative Sprache mit dem Ziel, möglichst kurze Anfragedauer bei möglichst wenig Einschränkung an Sprachkonstrukten und Funktionsumfang zu erreichen [CGT90].

²*Object Definition Language* ist eine Objektmodellierungssprache. Sie dient der Beschreibung der Typen des Datenbankschemas [PRCD94].

Definition 4: Das Informationsintegrationssystem (IIS) ist als ein Tripel $\langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ definiert, wobei:

- \mathcal{G} ist das globale Schema der Anwendungsdomäne, das in Sprache $\mathcal{L}_{\mathcal{G}}$ über dem Alphabet $\mathcal{A}_{\mathcal{G}}$ beschrieben ist. Dabei enthält das Alphabet ein Symbol für jedes Element von \mathcal{G} .
- \mathcal{S} ist die Menge von Datenmodellen der Quellen, die in Sprache $\mathcal{L}_{\mathcal{S}}$ über dem Alphabet $\mathcal{A}_{\mathcal{S}}$ beschrieben ist. Das Alphabet $\mathcal{A}_{\mathcal{S}}$ ³ enthält ein Symbol für jedes Element der Quellen.
- \mathcal{M} sind Korrespondenzen (mappings) zwischen \mathcal{G} und \mathcal{S} . Grundsätzlich enthält \mathcal{M} eine Menge von Abbildungen folgender Form:

$$q_{\mathcal{S}} \rightsquigarrow q_{\mathcal{G}},$$

$$q_{\mathcal{G}} \rightsquigarrow q_{\mathcal{S}},$$

wobei $q_{\mathcal{G}}$ und $q_{\mathcal{S}}$ zwei Anfragen über \mathcal{G} und \mathcal{S} repräsentieren. Die Anfragen $q_{\mathcal{S}}$ sind in der Anfragesprache $L_{\mathcal{M}, \mathcal{S}}$ über dem Alphabet $\mathcal{A}_{\mathcal{S}}$ formuliert. Analog, sind Anfragen $q_{\mathcal{G}}$ in der Anfragesprache $L_{\mathcal{M}, \mathcal{G}}$ über dem Alphabet $\mathcal{A}_{\mathcal{G}}$ formuliert. Intuitiv bedeutet die Abbildung $q_{\mathcal{S}} \rightsquigarrow q_{\mathcal{G}}$, dass durch die Anfrage $q_{\mathcal{S}}$ repräsentiertes Konzept einer Quelle auf das globale Konzept abgebildet wird. Das Konzept des globalen Schemas ist entsprechend mittels der Anfrage $q_{\mathcal{G}}$ beschrieben. Analog dazu ist die Abbildung $q_{\mathcal{G}} \rightsquigarrow q_{\mathcal{S}}$ definiert. In beiden Fällen müssen sowohl $q_{\mathcal{G}}$ als auch $q_{\mathcal{S}}$ die gleiche Stelligkeit haben.

Eine ausführliche Gegenüberstellung der beiden Ansätze liefern [Ul197], [Hal01], [Len02]. Der GaV-Ansatz ist der prozeduralen Vorgehensweise ähnlich, bei der die Daten ad hoc integriert werden, indem eine Menge vordefinierter Anforderungen beschreibt, wie die Daten aus den Quellen extrahiert werden sollen. Der Vorteil dieser Vorgehensweise ist, dass bestimmtes Wissen über Anwendungsdomäne genutzt werden kann, um Annahmen zwecks Leistungsverbesserung der Anfrageausführung treffen zu können. Der LaV-Ansatz entspricht dagegen einem deklarativen Vorgang. Die Idee ist, die Daten in den Quellen mit einer geeigneten Sprache so zu modellieren, dass eine einheitliche Darstellung für Anfragen gewährleistet ist. Der Vorteil dieses Ansatzes liegt darin, dass der Mechanismus zur Anfragebeantwortung generisch ist. Folglich liefert der LaV-Ansatz einerseits eine flexible und skalierbare Lösung, andererseits ist die Umsetzung einer Anfrage an das globale Schema in eine Anfrage an die Quellen wesentlich komplizierter als beim GaV-Ansatz. Daher wird dieser Sachverhalt im nächsten Abschnitt eingehender betrachtet.

³ $\mathcal{A}_{\mathcal{G}}$ und $\mathcal{A}_{\mathcal{S}}$ sind disjunkt, d. h. $\mathcal{A}_{\mathcal{G}} \cap \mathcal{A}_{\mathcal{S}} = \emptyset$.

Anfragebearbeitung

Anfragen des Benutzers oder einer Anwendung werden an das IIS in Begrifflichkeiten der Anwendungsdomäne formuliert. Die Berechnung einer Antwortmenge für die gestellte Anfrage bezeichnet man als Anfragebearbeitung (engl. *query processing*), die stark von der Modellierungsart der Korrespondenzen abhängig ist.

Gemäß der Definition 4 muss eine Anfrage q_G über das integrierte Schema in eine Anfrage q_S über die Quellschemata übersetzt werden. Beim GaV-Ansatz kann diese Aufgabe mittels einer Art Auffaltens (engl. *unfolding*) der Anfrage gelöst werden [Len02]. Hierzu ist eine Anfrage über \mathcal{G} gegeben; jedes Anfrageelement der q_G wird durch die rechte Seite der dazugehörigen Abbildung aus \mathcal{M} substituiert. Dabei erhält man eine Anfrage q_S , die mithilfe der Datenquellen beantwortet werden kann. Der beschriebene Vorgang funktioniert korrekt, wenn wir von der Prämisse ausgehen, dass die Quellen vollständig sind und über keine Integritätsbedingungen verfügen. Im Fall, dass die Sprache \mathcal{L}_G Integritätsbedingungen erlaubt und die Korrespondenzen korrekt sind, wird die Anfragebearbeitung mit GaV wesentlich komplizierter [CLR03], [CCGL04].

Die Anfragebearbeitung mithilfe von LaV kann als eine Art logische Schlussfolgerung mit unvollständigen Informationen betrachtet werden [vdM98]. Die Quellschemata sind in diesem Fall als Sichten auf das integrierte Schema modelliert. Folglich besteht die Aufgabe darin, die Anfrage mittels der Sichten zu beantworten. Das Ziel hierbei ist, eine gegebene Anfrage q in eine äquivalente Anfrage q' bestehend nur aus Sichten umzuschreiben (engl. *rewrite*). Es kann jedoch vorkommen, dass eine äquivalente Umschreibung nicht existiert, sodass die meisten Algorithmen in diesem Bereich eine maximal enthaltene (engl. *maximally contained*) Umschreibung für q erzielen.

Definition 5: Gegeben seien eine Anfrage q und eine Menge von Sichtendefinitionen (LaV-Korrespondenzen) $F = f_1, \dots, f_n$. Dann ist eine Anfrage q' eine äquivalente Umschreibung von q unter Verwendung von F , wenn:

- q' sich nur auf die Sichten in F bezieht und
- $q' \cup F$ zu q äquivalent ist.

Eine Anfrage q' ist eine maximal enthaltene Umschreibung von q unter Verwendung von F und gegebener Anfragesprache \mathcal{L} , wenn:

- q' eine Anfrage in \mathcal{L} ist, die sich nur auf die Sichten in F bezieht,
- $q' \cup F$ in q enthalten ist und
- es keine Umschreibung $q'' \in \mathcal{L}$ gibt, sodass $q' \cup F \subseteq q'' \cup F \subseteq q$ gilt und $q' \cup F$ nicht äquivalent zu $q'' \cup F$ ist.

In der Literatur ist das Problem der LaV-Anfragebearbeitung als Sichten-basierte Anfragebearbeitung (engl. *view-based query processing*) bekannt [Ull00], [Hal01] und kann grundsätzlich in zwei Ansätze unterteilt werden, nämlich Sichten-basierte Anfragebeantwortung (engl. *view-based query answering*) und Sichten-basierte Anfrageumschreibung (engl. *view-based query rewriting*) [Len02]. Der Unterschied der Ansätze liegt hauptsächlich darin, dass eine Sichten-basierte Anfrageumschreibung aus zwei Schritten besteht: (1) die Originalanfrage in Begriffe gegebener Anfragesprache über Quellschema zu umschreiben und (2) umgeschriebene Anfrage auszuwerten. Im Gegensatz dazu gibt es bei Sichten-basierter Anfragebeantwortung keine Einschränkungen hinsichtlich der Anfragebearbeitung; des Weiteren soll die gesamte Information, über die das System verfügt, genutzt werden, um die Antwort zu berechnen [CGLV00]. Zu drei bekannten Vertreter für praktische Algorithmen im Umfeld Sichten-basierter Anfragebeantwortung gehören „*the bucket algorithm*“ [LRO96], „*the inverse-rules algorithm*“ [GKD97] [AD98] und „*the MiniCon algorithm*“ [PH01].

2.2.3. Eigenschaften des Integrationsformalismus

Das Integrationssystem wird durch den Integrationsformalismus konfiguriert. Dieser Formalismus enthält eine Integrationsfunktion, mithilfe derer Informationen heterogener Informationsquellen auf die entsprechenden Strukturen des Integrationssystems abgebildet werden. Im Folgenden sollen die wesentlichen Eigenschaften des Integrationsformalismus untersucht werden, da sie das Design des Integrationssystems beeinflussen.

Frei von strukturellen und semantischen Heterogenitätskonflikten

Der Integrationsformalismus bietet die nötige Ausdruckstärke, um Abbildungen zu formulieren, die die strukturelle Heterogenität überwinden. Hierbei muss nicht ein neuer Formalismus entwickelt werden, sondern es kann ein Bestehender als Grundlage dienen. Die semantische Heterogenität ist durch Informationstransformation anhand des Kontextes behoben. An dieser Stelle sind domänenspezifische Konvertierungsfunktionen insbesondere für die Berücksichtigung der fehlenden Daten und alternativen Berechnungen zuständig.

Feingranulare Modularisierung

Ein hoher Modularisierungsgrad erleichtert die Komplexität der Integrationskonflikte und insbesondere deren Kombinationen zu beherrschen. Die Lösung von Integrationskonflikten unterschiedlicher Art kann in kleine modulare Einheiten unterteilt werden. Des Weiteren erlaubt die Modularisierung eine effiziente Anpassung von Änderungen. Mithilfe der hohen Modularität verfügt der Integrationsformalismus über einen strukturierten Integrationsansatz.

Partielle Repräsentation

Der Integrationsformalismus trennt die Aspekte der Integration, indem unterschiedliche Mechanismen zur Beseitigung von Heterogenitätskonflikten auseinandergehalten werden. So liegt eine explizite Separierung der Integrationsabbildung in einen strukturorientierten Teil zur Lösung struktureller Heterogenität und einen kontextorientierten Teil zur Unterstützung semantischer Interoperabilität vor. Die Vermischung beider Aspekte erschwert nicht nur die Wiederverwendung und Wartung der Integrationsabbildung, sondern beeinträchtigt auch den Integrationsformalismus hinsichtlich Skalierbarkeit und Flexibilität. Ferner erhöht die partielle Repräsentation die Modularisierung durch die Möglichkeit einer Zerlegung von Integrationsabbildung in noch kleinere, atomare Einheiten (Operationen).

Deklarativität und Angemessenheit

Der Integrationsformalismus ist deklarativ und angemessen, wenn eine einfache Bedienung und Wartbarkeit gegeben ist. Die Sicherstellung der Deklarativität erfolgt mittels der Transparenz operationaler Ebene. Somit ist sowohl strukturelle als auch semantische Integration unabhängig von Mechanismen zu logischer Schlussfolgerung beschrieben. Die Angemessenheit ist sichergestellt, wenn Elemente der heterogenen Informationsquellen mit Elementen des globalen Schemas in Beziehung gesetzt sind. Basierend auf dieser Zuordnung nimmt das Integrationsystem die Zerlegung einer globalen Anfrage vor.

Skalierbarkeit und Flexibilität

Der Integrationsformalismus ist dann skalierbar und flexibel, wenn sich neue Informationsquellen problemlos integrieren lassen, sodass keine Änderungen an der bestehenden Beschreibung vorgenommen werden müssen. Darüber hinaus sind Änderungen innerhalb einer Informationsquelle mit geringem Aufwand anpassbar.

2.3. Formale Wissensrepräsentation und Inferenz

In den vorangegangenen Abschnitten wurden Grundlagen und Herausforderungen bei der Informationsintegration verteilter und heterogener Informationsquellen betrachtet und analysiert. In den folgenden Abschnitten 2.3.1 und 2.3.2 werden wichtige Begriffe und Konzepte zur formalen Wissensrepräsentation und -verarbeitung in die Diskussion eingebracht, um eine Basis für semantische Modellierung von Kontextinformationen in Zusammenhang mit ontologiebasierten Informationsintegration zu schaffen. Darauf aufbauend wird auf die Familie der Beschreibungslogiken eingegangen. Zentrale Aspekte hierbei sind Ausdrucksmächtigkeit, Eigenschaften und Schlussfolgerungsalgorithmen. Abschließend wird der Schwerpunkt der Betrachtung auf die Beschreibungslogik \mathcal{ALC} bzw. \mathcal{SHOIN} gelegt, da sie die Grundlage der *Semantic Web* Sprache OWL bildet. Ein weiterer Schwerpunkt liegt auf der Betrachtung von Inferenzproblemen für Beschreibungslogiken.

2.3.1. Wissensbasierte Systeme

Der wichtigste Aspekt eines wissensbasierten Systems ist die Trennung der Darstellung des Wissens über den betreffenden Problembereich (Wissensbasis) und der Verarbeitung dieses Wissens (Wissensverarbeitung) [BKI06]. Konventionelle Systeme halten fallabhängigen Daten und domänenabhängigen Algorithmen auseinander, wissensbasierte Systeme hingegen differenzieren zwischen fallabhängigen Daten, einem domänenabhängigen Wissensmodell und einem domänenunabhängigen Reasoningalgorithmus [Pup93]. Folglich ist für die Entwicklung einer neuen Anwendung ein entsprechendes Wissensmodell der Anwendungsdomäne neu zu erzeugen. Der Mechanismus zur Wissensverarbeitung kann jedoch wiederverwendet werden, sodass fallabhängigen Daten im Kontext des neuen Wissensmodells verarbeitet und interpretiert werden können (siehe Abb. 2.4).

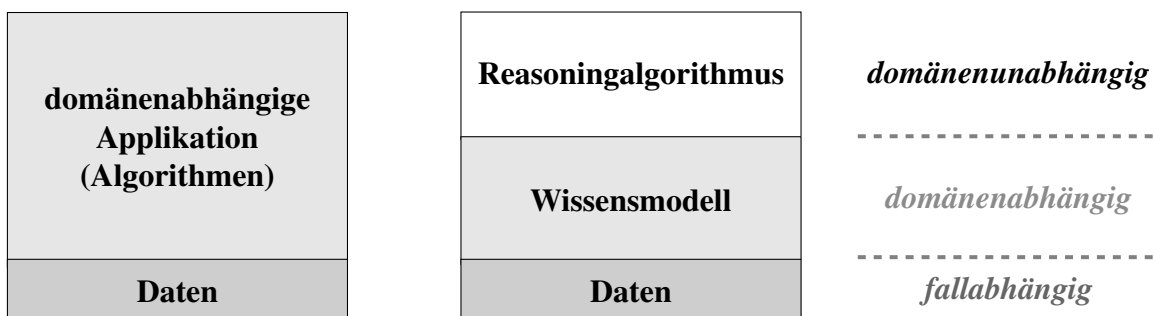


Abbildung 2.4.: Konventionelle Systeme vs. wissensbasierte Systeme [Pup93]

Somit sorgt das Wissensmodell für das gewünschte Verhalten der Anwendung. Um jedoch Wissensmodelle erstellen zu können, wird in der Regel die sogenannte Wissensrepräsentationsprache eingesetzt. Darüber hinaus ist der Reasoningalgorithmus genau dann hilfreich, wenn es

sich um unvollständige oder nur teilweise erfassbare Daten handelt [RN10]. An dieser Stelle ist es notwendig, die Begriffe *Wissensrepräsentation* und *Reasoning* näher zu betrachten, da sie grundlegend für wissensbasierte Systeme sind [SH09], [Bos11].

Wissensrepräsentation kann als Relation bzw. als eine Art Abbild zweier Bereiche betrachtet werden. Dabei werden Elemente des ersten Bereichs *Symbole* d. h. Zeichen oder Zeichenketten - aus einem gegebenen Alphabet auf die Elemente des zweiten Bereichs abgebildet. Die Elemente des zweiten Bereichs stellen Konzepte der realen Welt dar (siehe Abb. 2.5). Die Spezifikation, die festlegt, welche der Symbole in einer Wissensrepräsentationssprache vorkommen sollen, und wie daraus Sätze gebildet werden können, bezeichnet man als *Syntax* der Wissensrepräsentationssprache. Ferner wird die Zuordnungsvorschrift, die die Symbole und Sätze der Repräsentationsebene auf die Konzepte der repräsentierten Welt abbildet, *Semantik* genannt. Basierend auf dieser Relation erhalten die Sätze der Wissensrepräsentationssprache eine Bedeutung, die bestimmt, ob ein Satz der Sprache in einer gegebenen Welt eine wahre oder falsche Aussage ist.

Folglich setzt sich die Wissensrepräsentation mit der Verwendung von Symbolen auseinander mit dem Ziel, eine Menge von Aussagen formal darzustellen, die für eine Wissensdomäne wahr ist. Hierzu ist es nicht erforderlich alle für eine Wissensdomäne wahren Aussagen aufzuzählen - deren Zahl u. U. unendlich groß sein kann. Im Gegenteil: man stellt nur eine endliche Zahl von Aussagen explizit dar; mithilfe des Reasoning leitet man weitere wahre Aussagen implizit ab.

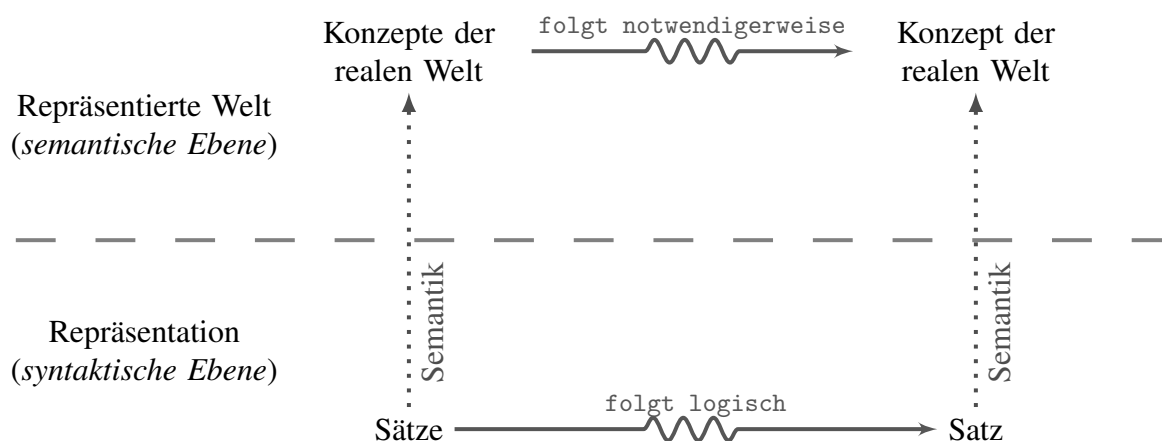


Abbildung 2.5.: Syntaktische und semantische Ebene [SS11]

Reasoning ist die formale Manipulation der Repräsentation von Aussagen (Symbolen, Sätzen), um die Repräsentation weiterer Aussagen zu generieren, die notwendigerweise aus diesen folgen [SS11]. Typischerweise verwendet man den Begriff *Reasoning* als allge-

meine Form zur Beschreibung des Vorgangs, die mit dem Schlussfolgerungen erzeugt werden.

Als Beispiel können die Sätze „*Ein iPad ist ein Produkt*“ und „*Ein Produkt hat einen Preis*“ mittels Operationen an den Symbolen zu dem Satz „*Ein iPad hat einen Preis*“ umgeformt werden. Diese Art des Reasoning bezeichnet man als (*logische*) *Inferenz*, weil das Ergebnis eine logische Schlussfolgerung der beiden Ausgangssätze ist. Eine andere Art des Reasoning ist z. B. topologisches Schlussfolgern auf Graphenstrukturen, indem die Nachbarschaftsbeziehung der Knoten betrachtet wird. Durch das Auffinden aller Knoten, die von einem betrachteten Knoten erreichbar sind, entsteht ein neuer Graph, der die Erreichbarkeitsbeziehung repräsentiert.

Zusammenfassend kann festgestellt werden, dass ein wissensbasiertes System aus zwei zentralen Komponenten besteht: einer Wissensbasis (engl. *knowledge base*) und einem Reasoningalgorithmus. Die Wissensbasis verfügt über eine Menge von Sätzen, die mit einer Wissensrepräsentationssprache formuliert sind. Der Reasoningalgorithmus ist für das Ableiten weiterer Sätze aus bereits vorhandenen verantwortlich.

Ferner kristallisieren sich aus dieser Erkenntnis zwei grundlegende Fragestellungen heraus:

- In welcher Form soll das Wissen in der Wissensbasis repräsentiert werden?
- Wie soll das Ableiten neuer Sätze aus vorhandenen Sätzen erfolgen?

Dabei ist zu beachten, dass der Reasoningaspekt in einer engen Beziehung mit dem Gegenstand der Repräsentation des Wissens steht. Bei der Informationsintegration verteilter und heterogener Datenquellen nehmen besonders Verständlichkeit, Skalierbarkeit und Ausdrucksmächtigkeit der zugrunde liegenden Wissensmodelle eine wesentliche Rolle ein. Aus diesem Grund werden in nächsten Abschnitten logikbasierte Formalismen und damit in Zusammenhang stehende Inferenzmechanismen näher betrachtet.

2.3.2. Logikbasierte Wissensrepräsentation

Die Logik ist eine wissenschaftliche Disziplin, die sich mit der Bildung korrekter Schlussfolgerungen beschäftigt. Das Hauptaugenmerk mathematischer Logik liegt auf der Formalisierung von Wissen und Argumentation. Dieser Aspekt macht die Logik für die Bereiche Wissensrepräsentation und Reasoning besonders interessant. Die bekannten Vertreter mathematischer Logik sind Aussagenlogik, Prädikatenlogik erster Stufe und Beschreibungslogiken. Logik ist im Grunde die Gesamtheit zweier Komponenten, nämlich Syntax und Semantik, wobei beide mathematisch eindeutig definiert sind. Der Hauptunterschied der Logik zu anderen formalen Sprachen besteht im Vorhandensein formaler Semantik. Die Programmiersprache C++ verfügt beispielsweise über keine formale Semantik, da die Bedeutung der Konstruktoren bei C++ im

Standard festgehalten ist. Der Standard der Sprache C++ ist aber in einer natürlichen (Englischen) Sprache verfasst. Basierend darauf stellen wir fest, dass das Reasoning – typischerweise in der Logik als Inferenz bzw. als logische Schlussfolgerung bezeichnet – sich formalisieren lässt.

Syntax und Semantik

Logikbasierte Wissensrepräsentation setzt die Definition der Syntax und Semantik voraus. Die syntaktische Ebene wird anhand der *Terme* und *Formeln* gebildet. Die semantische Ebene legt die *Interpretationen* fest, die einerseits für die Zuordnung syntaktischer Elemente einer Wissensbasis zu den Konzepten repräsentierter Welt verantwortlich sind, andererseits die Gültigkeit bzw. die Erfüllbarkeit einer Formel in einer Interpretation angeben [Ert09]. Die Abbildung 2.6 veranschaulicht diesen Sachverhalt.

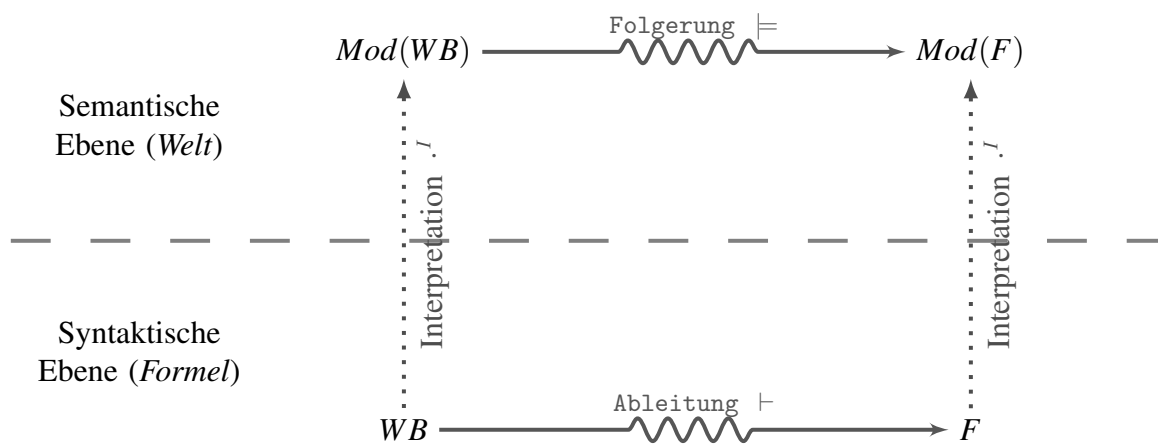


Abbildung 2.6.: Syntaktische Ableitung und semantische Folgerung. $Mod(F)$ steht für die Menge der Modelle einer Formel F [Ert09]

Die Syntax beschreibt basierend auf einem Vokabular, d. h. auf einer Menge von Namen, den Satzbau in einer Wissensbasis. Eine solche Menge wird auch *Signatur* Σ genannt; sie ist in der Regel domänenabhängig. Im Folgenden wird die syntaktische Struktur festgelegt:

Definition 6: Sei V eine Menge von Variablen, K eine Menge von Konstanten und F eine Menge von Funktionssymbolen mit $V \cap K \cap F = \emptyset$. Die Menge der Terme wird rekursiv definiert:

- Alle Variablen und Konstanten sind (atomare) Terme.
- Sind t_1, \dots, t_n Terme und f ein n -stelliges Funktionssymbol, so ist auch $f(t_1, \dots, t_n)$ ein Term.

Um logische Beziehungen zwischen Termen herstellen zu können, bauen wir aus Termen auf Basis bestimmter Regeln Formeln auf. Diese Regeln sind domänenunabhängig.

Definition 7: Sei P eine Menge von Prädikatsymbolen. Logische Formeln sind wie folgt aufgebaut:

- Sind t_1, \dots, t_n Terme und p ein n -stelliges Prädikatsymbol, so ist $p(t_1, \dots, t_n)$ eine (atomare) Formel.
- Sind A und B Formeln, so sind auch $\neg A$, (A) , $A \wedge B$, $A \vee B$, $A \Rightarrow B$, $A \Leftrightarrow B$ Formeln.
- Ist x eine Variable und A eine Formel, so sind auch $\forall x A$ und $\exists x A$ Formeln. \forall ist der All- und \exists der Existenzquantor.
- $p(t_1, \dots, t_n)$ und $\neg p(t_1, \dots, t_n)$ heißen Literale
- Formeln, bei denen jede Variable im Wirkungsbereich eines Quantors ist, heißen Aussagen oder geschlossene Formeln. Variablen, die nicht im Wirkungsbereich eines Quantors sind, heißen freie Variablen.

Somit stellt die $\text{Formel}(\Sigma)$ die Menge der Formeln dar, die in einer bestimmten Logik über der Menge Σ erzeugt werden kann. Folglich ist eine Wissensbasis WB eine Menge von Formeln über Σ , also $WB \subseteq \text{Formel}(\Sigma)$.

Die semantische Ebene definiert die Bedeutung von Formeln rekursiv über den Formelaufbau, indem den Elementen der Signatur Σ Objekte der repräsentierten Welt zugeordnet werden. Dazu führen wir den Begriff *Interpretation* oder *Belegung* ein.

Definition 8: Eine Interpretation I (oder auch Belegung \mathcal{B} genannt) ist definiert durch:

- eine nichtleere Grundmenge Δ^I (auch: Domäne, Trägermenge, Universum) und
- eine Abbildung $\cdot^I : \Sigma \rightarrow \Delta^I$, die jedem Namen in Σ ein Element aus Δ^I zuweist.

Basierend auf den $\text{Formel}(\Sigma)$ und $I(\Sigma)$ kann eine Relation definiert werden, die festlegt, ob eine Formel unter der Belegung \mathcal{B} gültig ist.

Definition 9: Eine (atomare) Formel F ist **erfüllbar** ($\text{Mod}(F)$) in einer Interpretation I , wenn es eine Belegung \mathcal{B} der in F vorkommenden Variablen gibt, die die Formel F **wahr** macht, d. h. für die $\mathcal{B}(F) = 1$ gilt.

Der Erfüllbarkeitsbegriff dient somit als Grundlage zur Definition der *logischen Folgerung* (engl. *entailment*):

Definition 10: Eine (atomare) Formel F **folgt** aus einer (atomaren) Formel G (oder $G \models F$), wenn jede Belegung \mathcal{B} , die G erfüllt, auch F erfüllt.

Hierbei ist zu beachten, dass in diesem Zusammenhang die Formel G wahr sein muss. Ist das nicht der Fall, und G ist nicht wahr (falsch), lässt sich entsprechend Definition 10 jede beliebige Aussage ableiten. Ein wichtiger Aspekt, der in diesem Kontext erläutert werden soll, ist die Interpretation der Aussage $G \not\models F$. Bei der Annahme einer geschlossenen Welt (engl. *closed world assumption*, *CWA*) wird diese Aussage als $G \models \neg F$ interpretiert [Rei77], [Lif85]. Das bedeutet, das Wissen, das nicht explizit als wahr bewiesen werden kann, wird als falsch angenommen (engl. *negation as failure*) [She84], [DM02]. Diese Annahme vertreten beispielsweise Datenbanken oder die Programmiersprache Prolog.

Die Annahme einer offenen Welt (engl. *open world assumption*, *OWA*), wertet dagegen die Aussage $G \not\models F$ nicht zwangsläufig als $G \models \neg F$ [GHMS09]. Die Aussage $G \models \neg F$ trifft nur dann zu, wenn diese explizit für alle Modelle abgeleitet werden kann (engl. *negation as unsatisfiability*) [RDH⁺04], [TBHH07]. Man geht hier davon aus, dass das vorhandene Wissen *unvollständig* ist. Klassische Logiken und damit auch die Prädikatenlogik und Beschreibungslogiken treffen eine OW-Annahme. In diesem Zusammenhang wird zur Verdeutlichung des Unterschieds das folgende Beispiel betrachtet:

Beispiel

Sei eine Datenbank DB mit einer Relation (Tabelle) `Projektmitarbeiter(id, name, projektleiter_id)` gegeben. Die Tabelle `Projektmitarbeiter` beinhaltet alle Mitarbeiter, die an einem Projekt beteiligt sind und eine Beziehung (Referenz) auf dazugehörigen Projektleiter. Darüber hinaus sei eine Einschränkung definiert, die besagt, dass jeder Projektmitarbeiter einen Projektleiter haben muss. Schließlich sei angenommen, dass die Tabelle lediglich über einen Datensatz (Tupel) verfügt: $\langle 001, \text{Müller}, \text{null} \rangle$.

Da für die Datenbank *CWA* gilt, bedeutet dieser Zustand der Datenbank, dass Projektmitarbeiter Müller keinen Projektleiter hat (er ist nicht in der DB vorhanden – *null*) und somit inkonsistent ist. Dabei ist die Anzahl der Projektmitarbeiter – laut Semantik der Datenbank – gleich eins. Die einfache SQL-Anfrage: `SELECT count(*) FROM Projektmitarbeiter;` liefert diese zurück.

Nun wird dieselbe Situation im Kontext einer Wissensbasis WB betrachtet, die im Gegensatz zu DB Offene-Welt-Annahme trifft. In diesem Fall kann festgestellt werden, dass WB konsistent ist. Anders ausgedrückt: Nur weil der Projektleiter von Müller nicht bekannt ist, kann nicht angenommen werden, dass es diesen nicht gibt. Des Weiteren kann eine Anwendung, die auf der Wissensbasis operiert, mindestens zwei Objekte der betrachteten Anwendungsdomäne entnehmen – Müller und seinen Projektleiter (trotz unbekannter ID von dem zweiten, wissen wir dass es ihn gibt).

Der beschriebene Sachverhalt illustriert einen wichtigen Unterschied der Verwendung von *CWA* und *OWA*:

- *CWA* ist nützlich bei der Prüfung der Datenkonsistenz
- *OWA* ist nützlich bei der Bearbeitung logischer Schlussfolgerung

Inferenz

Im vorangegangenen Abschnitt wurden Syntax und Semantik eines logischen Systems definiert, sodass eine solche Logik um eine Inferenzprozedur bzw. einen *Kalkül* erweitert werden kann. Ein Kalkül besteht aus einer Menge von logischen *Axiomen* und *Inferenzregeln* [BF92], [SA94]. Die Axiome sind entweder eine Menge von elementaren Tautologien oder eine Menge von elementaren Widersprüchen. Die Inferenzregeln eines Kalküls bilden eine Menge von Vorschriften, nach denen aus Formeln weitere Formeln abgeleitet werden können [BKI06]. Ist eine Formel F aus den Formeln F_1, \dots, F_n durch eine Folge von Anwendungen von Inferenzregeln eines Kalküls K ableitbar, so gilt $F_1, \dots, F_n \vdash_K F$. Dabei sind F_1, \dots, F_n die *Bedingungen* oder auch die *Prämissen*, und F ist die *Schlussfolgerung* oder auch *Konklusion* der Regel. In der Literatur unterscheidet man folgende Arten der Inferenz [Rud98], [Roo02], [BKI06], [Los06], [JLJ⁺10]:

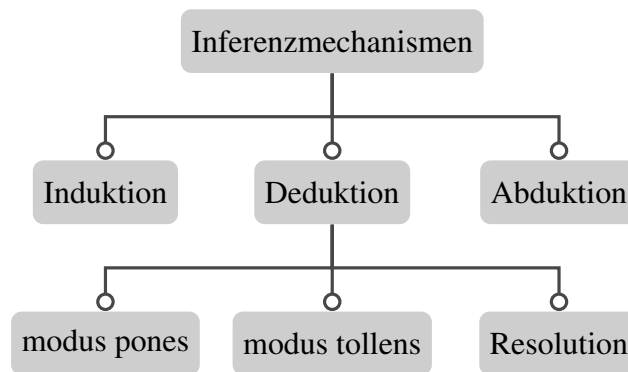


Abbildung 2.7.: Inferenzmechanismen

Unter *Induktion* wird die Ableitung einer Gesetzmäßigkeit aus vielen Einzelbeobachtungen, z. B. Lernen aus Erfahrung verstanden [Rud98]. Induktion ist also eine Schlussform, bei der vom Einzelfall auf das Allgemeine geschlossen wird (siehe Abb. 2.8); sie ist somit (wissens-)informationserweiternd [Los06]. Induktive Schlüsse sind Wahrscheinlichkeitsschlüsse, da die Konklusion nicht mit logischer Notwendigkeit aus den Einzelbeobachtungen (Prämisse) folgt, sondern nur mit gewisser Wahrscheinlichkeit [Roo02].

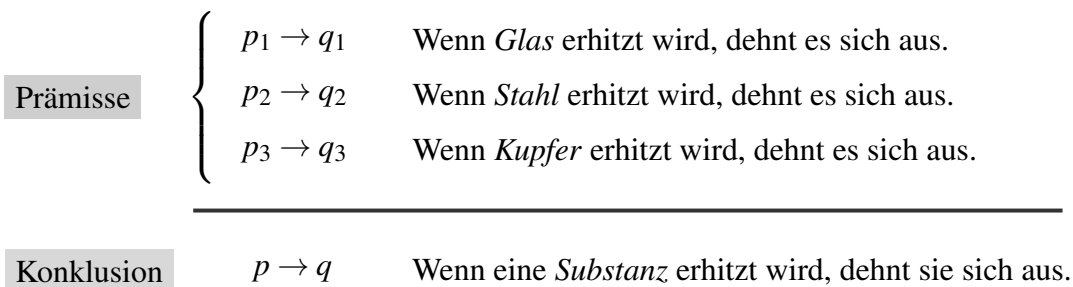


Abbildung 2.8.: Beispiel eines induktiven Schlusses (vgl. [Los06])

Unter *Abduktion* wird das Schließen aus Kausalwissen und Beobachtungen verstanden [Los06]. Abduktive Schlüsse sind also Schlüsse, mittels derer von einem beobachteten Ereignis auf dessen vermutete Ursachen geschlossen wird (siehe Abb. 2.9) [BKI06]. Der Einsatz dieses Schlusses erfolgt in Ingenieurwissenschaften im Wesentlichen in Bereichen der Diagnose, Fehlerbehebung und bei Planungsproblemen, da er nicht allgemeingültig ist [Los06], [JLJ⁺10].

Prämisse	$\left\{ \begin{array}{ll} p \rightarrow q & \text{Wenn } x \text{ ist aus Aluminium, dann korrodiert } x \text{ nicht.} \\ q & \text{(es ist gefordert, dass) } x \text{ nicht korrodiert.} \end{array} \right.$
Konklusion	$p \quad x \text{ besteht (muss hergestellt werden) aus Aluminium.}$

Abbildung 2.9.: Beispiel eines abuktiven Schlusses (vgl. [Los06])

Unter *Deduktion* wird der Beweis einer Aussage aus Axiomen durch formale Schlussverfahren verstanden [Los06]. Die Gültigkeit eines deduktiven Schlusses beruht nicht auf der Wahrheit seiner (gedanklichen) Inhalte, sondern hängt einzig und allein von der formalen Struktur des Schlusses ab (siehe Abb. 2.10) [Rud98].

Prämisse	$\left\{ \begin{array}{ll} p \rightarrow q & \text{Wenn } x \text{ aus Aluminium besteht, dann korrodiert } x \text{ nicht.} \\ p & x \text{ ist bzw. besteht aus Aluminium.} \end{array} \right.$
Konklusion	$q \quad x \text{ korrodiert nicht.}$

Abbildung 2.10.: Beispiel eines deduktiven Schlusses (vgl. [Los06])

Ein logischer Schluss ist nur dann gültig (also wahr), wenn *alle* Prämissen wahr sind [Los06]. Denn nur unter der Hinzunahme dieser Restriktion kann die Logik eine wahre bzw. gültige Konklusion *garantieren* [Los06], [Roo02]. Hierzu sind drei formale Schlussverfahren bekannt [Rud98], [BKI06], [Los06]:

- Der *Modus ponens* besagt, dass die Aussage A und die Aussage $A \rightarrow B$ zur Aussage B führt.
- Der *Modus tollens* besagt, dass die Aussage $A \rightarrow B$ und die Aussage $\neg B$ zur Aussage $\neg A$ führt.
- Die *Resolution* umfasst Modus ponens und Modus tollens und besagt, dass die Aussage $A \vee B$ und die Aussage $B \rightarrow C$ (äquivalent zu $\neg B \vee C$) zur Aussage $A \vee C$ (der sogenannten Resolvente) führt.

Korrektheit und Vollständigkeit der Kalküle

Um sicherzustellen, dass ein Kalkül fehlerfrei ist, werden im Folgenden zwei fundamentale Eigenschaften von Kalkülen definiert.

Definition 11: Ein Kalkül heißt *korrekt*, wenn jede abgeleitete Aussage auch semantisch folgt, d. h., wenn für Formeln F und G gilt

$$\text{falls } G \vdash_K F \text{ dann } G \models F$$

Ein Kalkül heißt *vollständig*, wenn alle semantischen Folgerungen abgeleitet werden können, d. h., wenn für Formeln F und G gilt

$$\text{falls } G \models F \text{ dann } G \vdash_K F$$

Folglich ist die einzige Art des *korrekten* Schließens die Deduktion, indem die logische Folgerung \models mittels der syntaktischen Ebene \vdash reproduziert wird. Sowohl Induktion als auch Abduktion stellen dagegen keine sichere bzw. korrekte Schlussart dar [Rud98], [Los06].

Beispiel für deduktive Kalküle gibt es für die Aussagenlogik sowie für Prädikatenlogik erster Stufe in Form von aussagen- bzw. prädikatenlogischer Resolution. Beide Beweisverfahren basieren auf dem Widerspruchsbeweis des Deduktionstheorems.

Definition 12: $G \models F$ gilt genau dann, wenn $G \wedge \neg F$ unerfüllbar ist.

Darüber hinaus bietet das logische Programmieren (im Vergleich zu klassischen Programmiersprachen C oder *Pascal*) die Möglichkeit, Zusammenhänge elegant, kompakt und deklarativ zu beschreiben und mittels Resolution zu entscheiden, ob eine Anfrage logisch aus einer Wissensbasis folgt.

Entscheidbarkeit und Komplexität von Problemen

Ein korrekter und vollständiger Kalkül ist beim Beweis einer wahren Aussage sehr hilfreich, denn nach endlicher Zeit stellt man aufgrund der Vollständigkeit fest, dass die Aussage wirklich wahr ist [Ert09]. Eine ganz andere Frage ist, ob zu einer gegebenen Logik überhaupt Kalküle existieren. Das heißt, ob es einen Algorithmus gibt, der die Frage nach der Erfüllbarkeit oder Gültigkeit von Formeln einer Logik beantwortet. Es hängt davon ab, ob diese Fragestellung für die gegebene Logik überhaupt *entscheidbar* ist [BKI06]. Im Folgenden wird der grundlegende Begriff der Entscheidbarkeit definiert:

Definition 13: Es sei Σ ein Alphabet und $M_1 \subseteq M_2 \subseteq \Sigma^*$ definiert. Die Menge M_1 heißt **entscheidbar** relativ zu M_2 , wenn es einen Algorithmus $A_{M_1, M_2} : M_2 \rightarrow \{\text{true}, \text{false}\}$ gibt, mit dessen Hilfe man zu jedem Element $w \in M_2$ feststellen kann, ob es zu M_1 gehört oder nicht; kurz:

$$\forall w \in M_2 : A_{M_1, M_2}(w) ::= \begin{cases} 1, & \text{falls } w \in M_1 \\ 0, & \text{falls } w \notin M_1 \end{cases}$$

Analog hierzu heißt die Menge M_1 **semientscheidbar** relativ zu M_2 , wenn

$$\forall w \in M_2 : A_{M_1, M_2}(w) ::= \begin{cases} 1, & \text{falls } w \in M_1 \\ \text{nicht definiert,} & \text{falls } w \notin M_1 \end{cases}$$

Entscheidbarkeit ist die Eigenschaft einer Menge und nicht etwa eines einzelnen Objektes oder anderer Dinge. Oft spricht man auch von entscheidbaren (d. h. algorithmisch lösbaren) *Problemen*, etwa: „Es ist entscheidbar, ob eine vorgegebene natürliche Zahl eine Primzahl ist oder nicht.“

Die Entscheidbarkeit untersucht also, ob sich ein Problem algorithmisch lösen lässt. Die *Komplexitätstheorie* fragt darüber hinaus nach dem Aufwand, mit dem dies möglich ist. Unter *Komplexität* eines Problems wird der geringstmögliche Aufwand verstanden, den man mit irgendeinem Algorithmus für das Problem erreichen kann. Ferner werden Probleme und Algorithmen verschiedener *Komplexitätsklassen* hinsichtlich ihres Speicher- und Zeitaufwands $s(n)$ bzw. $t(n)$ zugeordnet.

Ein Algorithmus \mathcal{A} für ein Problem \mathcal{P} benötigt den Aufwand $t(n)$, wenn er im schlimmsten Fall $t(n)$ Schritte benötigt, um eine Eingabe des Umfangs n zu bearbeiten [Goo97]. Es könnte auch eine Eingabe geben, für die weniger Schritte erforderlich sind; der Aufwand von \mathcal{A} ist eine obere Schranke. Das Problem \mathcal{P} hat die Komplexität $t(n)$, wenn es einen Algorithmus \mathcal{A} mit Aufwand $t(n)$ für \mathcal{P} gibt, aber keine schnelleren Algorithmen. Damit ist die Komplexität von \mathcal{P} nach unten begrenzt. Man kann oft die Komplexität für andere Probleme \mathcal{P}' abschätzen, indem man ein Problem \mathcal{P} bekannter Komplexität auf \mathcal{P}' zurückführt (reduziert), in Zeichen $\mathcal{P} \preceq \mathcal{P}'$ [Goo97].

Zu jedem Problem gibt es in der Regel verschiedene Lösungsalgorithmen, die sich in ihrer Komplexität unterscheiden. Letztendlich bestimmt die gesamte Komplexität die Effizienz einzelner Algorithmen. Ein Algorithmus heißt *effizient* für ein Problem, wenn er dieses mit der minimalen Komplexität löst [LR02]. Ein anderes Kriterium für Algorithmen ist ihre *Durchführbarkeit* (engl. *tractability*), denn selbst ein Algorithmus mit exponentiellem Zeitaufwand kann effizient sein. In der Regel gelten Algorithmen bzw. Probleme bis zu polynomialer Komplexität als durchführbar und Algorithmen mit exponentiellem Aufwand sind hingegen nicht mehr handhabbar. Zu beachten ist, dass diese Angaben nur asymptotisch für große n gelten.

Zusammenfassend kann festgehalten werden, dass alle Probleme und ihre Lösungsalgorithmen in zwei große Gruppen eingeordnet werden können, nämlich mit polynomialem oder exponentiellem Aufwand. Ferner ist es ersichtlich, dass bestimmte exponentiell beschränkte Probleme nur dann *tractable* sind, wenn ein Lösungsvorschlag zufällig aufgestellt und in polynomialer Zeit überprüft werden kann [LR02]. Insofern kann dies als eine Art Überführung in ein Entscheidungsproblem betrachtet werden (Definition 13).

2.4. Ontologien und Beschreibungslogiken

Dem aufmerksamen Leser wird nicht entgangen sein, dass sowohl im Bereich „Informationsintegration“ als auch im Bereich „Wissensrepräsentation“ eine allgemeine Beschreibung des Diskursbereiches durch gemeinsames Vokabular ein zentrales Thema in Bezug auf Entwicklung und Wiederverwendung von Wissen ist. Hierzu werden Ontologien, die allgemeines Wissen über die zu modellierende Welt enthalten, entwickelt und wiederverwendet. Dieser Begriff stammt aus der Philosophie und bezeichnet eine Disziplin, die sich mit dem Grundsätzlichsten des Seins beschäftigt. Der Ausdruck setzt sich aus dem griechischen Partizip *on* (\equiv seiend) und *logos* (\equiv Lehre, Wort) zusammen.

2.4.1. Ontologie

Um den Begriff „Ontologie“ einerseits vom philosophischen Verständnis abzugrenzen und andererseits ein Bezug zu logikbasierten Wissensrepräsentation herzustellen, sprechen wir von *formaler Ontologie*. Die wichtigste und am häufigsten zitierte Definition stammt von TOM GRUBBER aus dem Jahr 1993:

Definition 14: „*A specification of a representation vocabulary for a shared domain of discourse – definitions of classes, relations, functions, and other objects – is called an **ontology**“ [Gru93].*

„*An **ontology** is an explicit specification of a conceptualization. The term is borrowed from philosophy, where an ontology is a systematic account of Existence. For knowledge-based systems, what 'exists' is exactly that which can be represented“ [Gru94].*

TOM GRUBBERS Ontologiedefinition wurde von STUDER ET AL. unter Verlagerung des Schwerpunkts angepasst [SBF98]:

Definition 15: *Eine Ontologie ist eine formale, explizite Spezifikation einer gemeinsamen Konzeptionalisierung.*

Zum einen steht die Hervorhebung der Interoperabilität im Vordergrund, da eine Ontologie

als „gemeinsame“ (engl. *shared*) Konzeptionalisierung dargestellt wird. Die Eigenschaft *shared* fordert, dass Ontologien zwischen Personen, Gruppen von Personen oder Anwendungssystemen geteilte Sicht auf eine Domäne bereitstellen. Zum anderen bedeutet die Anforderung „formal zu sein“, dass Ontologien in einer maschinell verarbeitbaren Sprache definiert werden müssen, wobei noch nicht geklärt ist, ob eine derartige Sprache eine formal definierte Semantik besitzt oder nicht. Ontologien stellen also das gemeinsame Verständnis einer Domäne zur Verfügung und ermöglichen eine Kommunikation zwischen menschlichen und maschinellen Akteuren. GRUBER fasst diese Aspekte in der aktuellen Enzyklopädie zusammen:

[...] an ontology defines a set of representational primitives with which to model a domain of knowledge or discourse. The representational primitives are typically classes (or sets), attributes (or properties), and relationships (or relations among class members). The definitions of the representational primitives include information about their meaning and constraints on their logically consistent application. In the context of database systems, ontology can be viewed as a level of abstraction of data models, analogous to hierarchical and relational models, but intended for modeling knowledge about individuals, their attributes, and their relationships to other individuals. [Gru09]

Wie nun deutlich wird, sagen die vorgestellten Definitionen nichts über die Wissensrepräsentationssprache aus, mittels derer eine Ontologie spezifiziert werden soll. In Wirklichkeit kann der Formalitätsgrad, mit dem eine Ontologie beschrieben wird, vielfältig sein. In der Literatur werden folgende Formalitätsgrade von Ontologien unterschieden [Edm94], [Usc03]:

informal: Ausgedrückt durch natürliche Sprache, beispielsweise in einem Standardisierungsdokument.

semi-informal: Ausgedrückt in einer strikteren und strukturierteren Form der natürlichen Sprache. Dadurch soll die einheitliche Verwendung bestimmter Begriffe eindeutig sein.

semi-formal: Ausgedrückt in einer künstlichen, formal definierten Sprache. Dabei ist die Semantik relevanter Begriffe implizit festgelegt.

formal: Die Semantik der verwendeten Begriffe ist explizit und in maschinenverstehbarer Form formuliert. Die Begriffe erfüllen die Eigenschaften wie Korrektheit und Vollständigkeit.

Im Rahmen einer semantischen Informationsintegration heterogener Datenbestände ist eine *formale* Formalisierung erforderlich, die sowohl für den Menschen als auch für die Maschine verständlich und verarbeitbar ist. In dem nachfolgenden Abschnitt werden die *Beschreibungslogiken* beschrieben, die als Wissensrepräsentationssprache für derartige Ontologien dienen.

2.4.2. Beschreibungslogiken und Ontologiesprachen

Beschreibungslogiken (engl. *Description Logics DL*) sind eine Familie von Formalismen zur Wissensrepräsentation. Ihre Ursprünge lassen sich bis in die Zeitspanne 1965 – 1980 zurückverfolgen, als informale Ansätze zur Wissensrepräsentation wie z. B. *semantischen Netzen* und *Konzeptrahmen* erforscht wurden [Qui67], [Min81]. Man hat diese Ansätze als eine intuitive und praktisch besser anwendbare Alternative zur Prädikatenlogik erster Stufe betrachtet. Zeitnah kamen jedoch die Nachteile aufgrund fehlender formaler Semantik zum Vorschein, sodass verschiedene Systeme trotz derselber Formalismen unterschiedliches Verhalten aufwiesen [BCM⁺03]. Daraufhin hat man beide Welten miteinander in Beziehung gebracht, indem man intuitive Modellierungskonzepte von semantischen Netzen und Konzeptrahmen durch eine wohldefinierte Semantik erweiterte.

Im Zuge dieser Kopplung stellte man fest, dass zur Definition einer Semantik für konzeptuelle Strukturen die Prädikatenlogik erster Stufe als Grundlage verwendet werden konnte. Hierzu setzte man nur *entscheidbare Fragmente* der PL1 ein, um die Ausdrucksmöglichkeit der Wissensrepräsentationssprache zu bestimmen. Darauf basierend konnte die logische Schlussfolgerung automatisiert und die Entwicklung sogenannter *Konzeptsprachen* in die Wege geleitet werden. Aus Konzeptsprachen entstanden später die Beschreibungslogiken. Dabei fokussierte man sich auf das gegenseitige Verhalten von Ausdruckstärke einer Beschreibungslogik und damit verbundener Komplexität der Verfahren zur Lösung von Inferenzproblemen. Die Komplexität hängt direkt mit der Auswahl von Konstrukten zusammen, die man in die Sprache aufnimmt. Die tatsächlichen Realisierungen der Inferenzverfahren zeigen, dass die ungünstigsten Fälle (engl. *worst case*) in der Praxis selbst bei sehr ausdrucksstarken Beschreibungslogiken nur selten auftreten [BCM⁺03]. Zusammenfassend kann festgehalten werden, dass die *DL*-Sprachfamilie unterschiedliche Logiken darstellt, die anhand von Konstrukten zur Konzeptbeschreibung gebildet und charakterisiert werden [The01], [Fuc08].

Beschreibungslogische Wissensbasis

Mithilfe von Beschreibungslogiken kann eine Ontologie spezifiziert werden, die dementsprechend eine beschreibungslogische Wissensbasis bildet. Die Abbildung 2.11 zeigt eine beschreibungslogische Wissensbasis \mathcal{WB} , die aus zwei Komponenten, nämlich der \mathcal{T} -Box (*terminological box*) und der \mathcal{A} -Box (*assertional box*) besteht $\mathcal{WB} = (\mathcal{T}, \mathcal{A})$. Eine TBox \mathcal{T} ist für die Beschreibung der Anwendungsdomäne anhand des dazugehörigen Vokabulars verantwortlich. Eine ABox \mathcal{A} spezifiziert hingegen die Zusicherungen (engl. *assertions*) über Individuen der betrachteten Domäne unter der Hinzunahme des Vokabulars. Das Vokabular setzt sich aus *Konzepten* und *Rollen* zusammen. Ein Konzept ist eine Bezeichnung für die Menge der *Individuen*, Rollen hingegen sind nichts anderes als binäre Relationen, die Individuen zueinander in

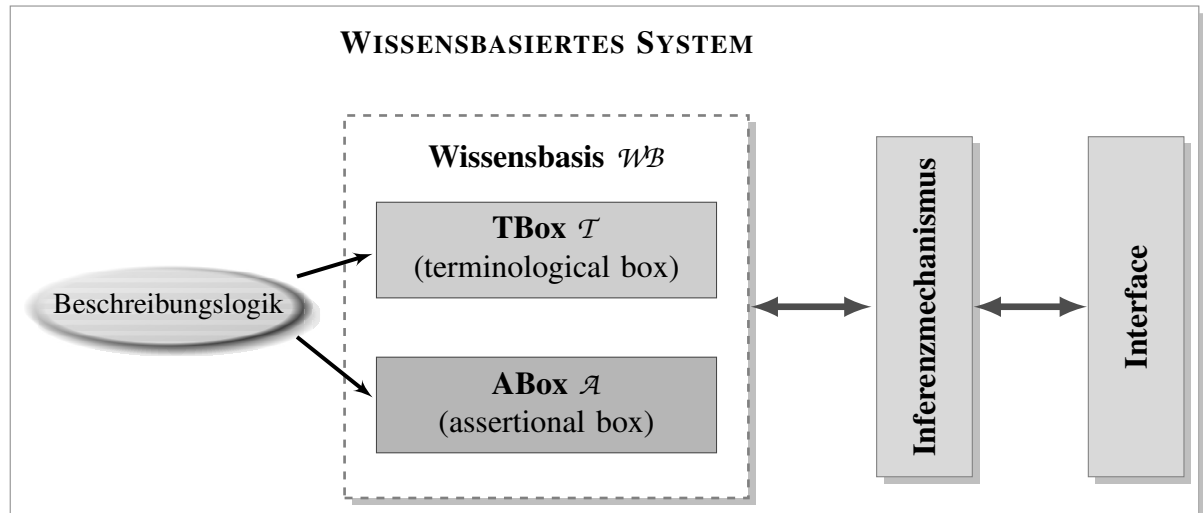


Abbildung 2.11.: Wissensbasiertes System mit Beschreibungslogiken

Beziehung setzen. Alle Beschreibungslogiken verfügen über die Möglichkeit, mittels atomarer Konzept- und Rollennamen komplexe Konzepte und Rollen zu definieren.

Neben einer Wissensbasis stellt ein wissensbasiertes System mit Beschreibungslogiken einen *Inferenzmechanismus* zum automatischen Schlussfolgern zur Verfügung. Die Inferenz kann nur in Bezug auf $TBox$ allein oder auch in Bezug auf die gesamte Wissensbasis ($TBox$ und $ABox$) erfolgen. In diesem Zusammenhang treten viele Inferenzprobleme für die $TBox$ auf, wie *Konzepterfüllbarkeit*, *Subsumptions-Test*. Zu den wesentlichen Inferenzproblemen hinsichtlich der gesamten Wissensbasis gehören die *Konsistenzprüfung* und die *Instanzprüfung*. Im Folgenden Abschnitt sollen die oben beschriebenen Inferenzprobleme diskutiert werden.

Die einfache Beschreibungslogik \mathcal{ALC}

Um eine Beschreibungslogik \mathcal{DL} aufstellen zu können, müssen Syntax und Semantik definiert werden. Die Syntax beschreibt, ob die Ausdrücke (Konzepte, Rollen, Axiome etc.) richtig (im Sinne von korrekt) in der Logik aufgestellt sind. Die Semantik weist darauf hin, wie diese Ausdrücke interpretiert werden sollen, d. h. sie gibt die formale Bedeutung an. Im Folgenden wird die einfache Beschreibungslogik \mathcal{ALC} (*Attributive Language with Complement*) eingeführt, an der die wichtigsten Grundlagen für Beschreibungslogiken erläutert werden sollen.

Definition 16: Die Syntax der Logik \mathcal{ALC} ist folgendermaßen induktiv definiert:

- die Symbole \top und \perp sind Konzepte, genannt universelles und leeres Konzept;
- beliebiges atomares Konzept A , genannt Konzeptname;
- wenn C ein Konzept ist, dann ist $\neg C$ auch ein Konzept, genannt Negation;

- wenn C ein Konzept und R eine atomare Rolle ist, dann sind $\exists R.C$ und $\forall R.C$ auch Konzepte;
- alle anderen Ausdrücke sind keine Konzepte.

Nachfolgend wird zur Syntaxbeschreibung eine bündige Schreibweise verwendet. So kann die Syntax für Konzepte der Logik \mathcal{ALC} wie folgt formuliert werden:

$$\top \mid \perp \mid A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists R.C \mid \forall R.C, \text{ wobei}$$

A ein atomares Konzept, R eine atomare Rolle und C, D beliebige Konzepte sind.

Beispiel

Es seien atomare Konzepte `product`, `producer`, `customer`, `supplier` und atomare Rollen `hasPart`, `isSatisfied`. Dann gehören folgende Konzepte der Logik \mathcal{ALC} an:

- `producer` \sqcup `supplier`
- `product` \sqcap `customer`
- $\forall \text{hasPart} . \neg \text{product}$
- `producer` \sqcap $\exists \text{isSatisfied} . (\text{supplier} \sqcap \text{customer})$
- ...

Die Semantik der Logik \mathcal{ALC} wird mittels Interpretation festgelegt.

Definition 17: Eine Interpretation ist ein Paar $I = (\Delta^I, \cdot^I)$ bestehend aus einer nichtleeren Grundmenge Δ^I und der Interpretationsfunktion \cdot^I . Die Interpretationsfunktion \cdot^I setzt in Beziehung:

- jedem atomaren Konzept A eine beliebige Menge $A^I \subseteq \Delta$;
- jeder atomaren Rolle R eine beliebige Menge $R^I \subseteq \Delta \times \Delta$.

Die Interpretationsfunktion wird für komplexe Konzepte anhand induktiver Definition erweitert:

$$\begin{array}{l}
 \top^I = \Delta, \quad \perp^I = \emptyset, \quad (\neg C)^I = \Delta \setminus C^I \\
 (C \sqcap D)^I = C^I \cap D^I, \quad (C \sqcup D)^I = C^I \cup D^I \\
 (\exists R.C)^I = \{a \in \Delta \mid \exists b \in \Delta : (a, b) \in R^I \wedge b \in C^I\} \\
 (\forall R.C)^I = \{a \in \Delta \mid \forall b : (a, b) \in R^I \Rightarrow b \in C^I\}
 \end{array}$$

Im Folgenden sollen logische Formeln der Logik \mathcal{ALC} näher betrachtet werden. Diese werden als Axiome bezeichnet und geben einerseits Auskunft darüber, in welcher Beziehung komplexe Konzepte und Rollen zueinander stehen; andererseits geben sie Auskunft darüber, wie die Zusicherungen bzgl. Individuen spezifiziert sind. Es sind drei Typen von Axiomen zu unterscheiden:

1. Die Axiome vom Typ einer „Untermenge“ bzw. sie können in den Begriffen des Objektorientierten Paradigmas (OOP) als Unterklasse (Vererbung) bezeichnet werden. Diese Axiome haben die Form $C \sqsubseteq D$, wobei C und D beliebige u. a. auch komplexe Konzepte sind.
2. Die Axiome vom Typ „Element der Menge“. Auch sie lassen sich in der OOP-Welt modellieren, indem man dazu „Instanzen einer Klasse“ sagt. Diese Axiome haben die Form $a : C$, wobei a ein Element (Objekt) und C die Menge (eine Klasse) ist.
3. Die Axiome vom Typ „Element der Rolle“. Die Axiome dieser Art haben die Form $(a, b) : R$, wobei a, b zwei Elemente (Objekte) sind und R eine beliebige Rolle.

Die Menge von Axiomen des 1. Typs bilden die TBox. Die Axiome des Typs 2. und 3. sind die Bestandteile der ABox. Darüber hinaus müssen Axiome ebenfalls mit einer Semantik versehen werden. In diesem Zusammenhang sollen die dazugehörigen Interpretationen betrachtet werden:

1. Eine Interpretation I erfüllt das Axiom $C \sqsubseteq D$, wenn $C^I \subseteq D^I$ gilt.
2. Eine Interpretation I erfüllt das Axiom $a : C$, wenn $a^I \in C^I$ gilt.
3. Eine Interpretation I erfüllt das Axiom $(a, b) : R$, wenn $(a^I, b^I) \in R^I$ gilt.

Wird ein Axiom A von einer Interpretation I erfüllt, so ist A ein Modell (daher die Bezeichnung „Modelltheoretische Semantik“). Grundsätzlich kann gesagt werden: Eine Interpretation I erfüllt die TBox (oder ist ein Modell der TBox), wenn sie ein Modell für alle Axiome der

TBox ist. Analog dazu definiert man ein Modell für die ABox. Schließlich: Eine Interpretation I ist ein Modell einer Ontologie, wenn sie ein Modell der TBox sowie der ABox ist.

SCHMIDT-SCHAUSS und SMOLKA schlugen zur exakten Benennung einer Beschreibungslogik folgende Namenskonvention vor [SSS91]: Jeder Konstruktor wird durch einen bestimmten Großbuchstaben dargestellt. So wird beispielsweise die Erweiterung von \mathcal{ALC} um transitive Rollen als \mathcal{S} , die Rollenhierarchie (Rollenenklusion) als \mathcal{H} bezeichnet. Die funktionalen oder inversen Rollen werden entsprechend durch die Buchstaben \mathcal{F} , \mathcal{I} bezeichnet.

Die Entscheidbarkeit sowie die Komplexität hängen direkt mit den Konstruktoren zusammen, die hinzugenommen bzw. weggelassen werden können. Ferner ist eine derart präzise und strikte Namenskonvention von essenzieller Bedeutung hinsichtlich Inferenzprobleme.

2.5. Inferenzprobleme für Beschreibungslogiken

Das logische Schlussfolgern auf dem formal repräsentierten Wissen ist der Hauptbestandteil eines beschreibungslogischen wissensbasierten Systems. Wie bereits festgestellt wurde, verfügt eine beschreibungslogische Wissensbasis neben dem explizit repräsentiertem Wissen über implizites Wissen, das mittels der Semantik definiert ist. Die logischen Schlussfolgerungsmechanismen haben die Aufgabe, dieses implizite Wissen explizit zu machen. In diesem Zusammenhang treten in der Regel vier grundlegende Inferenzprobleme auf:

- *Konsistenz der Wissensbasis*: Gibt es für eine Wissensbasis \mathcal{WB} mindestens ein nichtleeres Modell, also $\mathcal{WB} \models \top \neq \perp$. Mit anderen Worten: Es handelt sich um ein Problem, das entscheidet, ob eine Wissensbasis sinnvoll ist ($\mathcal{WB} \models \text{false}$?).
- *Konzeptkonsistenz*: Existiert ein Modell von \mathcal{WB} , in dem Konzept C als nichtleere Menge interpretiert wird, also $\mathcal{WB} \models C \neq \perp$. Anders ausgedrückt: Es handelt sich um ein Problem, das entscheidet, ob ein Konzept C leer sein muss ($C \equiv \perp$?).
- *Konzeptinklusion* (engl. *Subsumption*): Ein Konzept C ist in einem Konzept D bzgl. \mathcal{WB} enthalten, wenn für jedes Modell von \mathcal{WB} die Interpretation von C eine Teilmenge von der Interpretation von D ist, also $\mathcal{WB} \models C \sqsubseteq D$. Mit anderen Worten: Es handelt sich um ein Problem, das entscheidet, ob eine Wissensbasis strukturiert ist.
- *Konzeptzugehörigkeit*: Gegeben sei eine Instanz a und ein Konzept C . Das Individuum a gehört dem Konzept C an, wenn für jedes Modell von \mathcal{WB} die Interpretation von a in der Interpretation von C enthalten ist, also $\mathcal{WB} \models C(a)$. Anders ausgedrückt: Es handelt sich um ein Problem, das entscheidet, ob ein Element (Individuum) a in der Menge (Konzept) C ist.

Ein wichtiger Aspekt hierbei ist, dass die oben aufgeführten Inferenzprobleme sich auf die Konsistenzüberprüfung einer Wissensbasis zurückführen lassen. Ferner sei a ein Individuum

aus \mathcal{WB} und b ein Individuum, das nicht in \mathcal{WB} ist; also $a \in \mathcal{WB}, b \notin \mathcal{WB}$. Dann können Inferenzprobleme folgendermaßen gelöst werden:

- *Konzeptkonsistenz*: $\mathcal{WB} \models C \not\equiv \perp \Leftrightarrow \mathcal{WB} \cup \{C(b)\}$ ist konsistent;
- *Konzeptinklusion*: $\mathcal{WB} \models C \sqsubseteq D \Leftrightarrow \mathcal{WB} \cup \{C \sqcap \neg D(b)\}$ ist nicht konsistent;
- *Konzeptzugehörigkeit*: $\mathcal{WB} \models C(a) \Leftrightarrow \mathcal{WB} \cup \{\neg C(a)\}$ ist nicht konsistent.

Die Ausdruckstärke bei der Spezifikation einer Beschreibungslogik hängt mit der Wahl der Konstruktoren zusammen. Daher besteht das Ziel darin, die Konstruktoren so auszuwählen, dass einerseits die Ausdrucksstärke möglichst maximal, andererseits beschriebene grundlegende Inferenzprobleme entscheidbar sind.

Dieses Forschungsgebiet ist aktuell und neue Ergebnisse können der Website *Description Logic Complexity Navigator* entnommen werden [Zol10]. Der angegebene Aufwand ist jedoch als eine Worst-Case-Komplexität zu verstehen. In praxisrelevanten Anwendungen ist die tatsächliche Komplexität in der Regel wesentlich niedriger.

2.5.1. Konjunktive Anfragen für Beschreibungslogiken

In diesem Abschnitt werden Probleme betrachtet, die bei der Bearbeitung von Anfragen bezüglich einer \mathcal{DL} -Wissensbasis vorkommen. Konjunktive Anfragen an eine \mathcal{WB} sind Konjunktionen von Anfrageatomen der Form $Q(\vec{x}) \leftarrow Z_1(\vec{t}_1) \wedge \dots \wedge Z_n(\vec{t}_n)$. Dabei gilt die Annahme, dass (1) jedes Atom im Anfragekörper (engl. *body*) unär oder binär ist, (2) ein unäres Prädikat in der Anfrage einem Konzept in \mathcal{WB} zugeordnet ist, (3) ein binäres Prädikat in der Anfrage einer atomaren Rolle in \mathcal{WB} entspricht und schließlich (4) der Anfragekopf (engl. *head*) $Q(\vec{x})$ nicht in \mathcal{WB} vorkommt. Beispielsweise kann die folgende Anfrage an die Wissensbasis gestellt werden: Gefragt ist ein Produkt mit dazugehörigen Baugruppen und Unterbaugruppen:

$$Q(u, w, z) \leftarrow \text{Produkt}(u) \wedge \text{hatTeil}(u, w) \wedge \text{hatTeil}(w, z) \quad [2.1]$$

Bevor die Problemstellung im Zusammenhang mit der Bearbeitung einer konjunktiven Anfrage formal definiert wird, soll zunächst der Sachverhalt informal an einem intuitiven Beispiel betrachtet werden.

Die TBox \mathcal{T} enthält folgende Axiome:

$$\text{Komponente} \sqcap \text{Top} \sqsubseteq \text{Produkt} \quad [2.2]$$

$$\exists \text{hatTeil}.\text{Baugruppe} \sqsubseteq \text{Komponente} \quad [2.3]$$

$$\exists \text{hatTeil} \sqsubseteq \text{Baugruppe} \quad [2.4]$$

Die ABox \mathcal{A} der Wissensbasis verfügt über folgende Zusicherungen:

$$Top(\text{Audi S4 Limousine}) \quad [2.5]$$

$$hatTeil(\text{Audi S4 Limousine}, \text{Motor3.0 TFSI quattro}) \quad [2.6]$$

$$hatTeil(\text{Motor3.0 TFSI quattro}, \text{V6-Zylinder}) \quad [2.7]$$

Sei Q [2.1] eine Anfrage an die Wissensbasis. Dann kann leicht festgestellt, dass \mathcal{WB} anhand von [2.7] und [2.4] folgendes impliziert:

$$Baugruppe(\text{Motor3.0 TFSI quattro}) \quad [2.8]$$

Analog dazu impliziert \mathcal{WB} mittels [2.8], [2.6] und [2.3]

$$Komponente(\text{Audi S4 Limousine}) \quad [2.9]$$

Schließlich kann anhand von [2.9], [2.5] und [2.2]

$$Produkt(\text{Audi S4 Limousine}) \quad [2.10]$$

hergeleitet werden. Des Weiteren wird ausgegeben: $\mathcal{WB} \models \{[2.10], [2.7], [2.6]\}$ und intuitiv bedeutet dies in Bezug auf \mathcal{WB} , dass Audi S4 Limousine ein Produkt, Motor3.0 TFSI quattro ein Teil davon ist, und V6-Zylinder ein Teil von Motor3.0 TFSI quattro ist. Es ist trivial zu verifizieren, dass die ermittelten Individuen die Anfrage Q erfüllen. Darüber hinaus ist das Tripel $\langle \text{Audi S4 Limousine}, \text{Motor3.0 TFSI quattro}, \text{V6-Zylinder} \rangle$ eine *sichere Antwort* der Anfrage Q über \mathcal{WB} .

Die Menge von Klauseln im Anfragekörper kann als eine Formel in Prädikatenlogik erster Stufe betrachtet werden [Fit96]. Zusätzlich werden die in dieser Arbeit behandelten Beschreibungslogiken als ein Fragment der PL1 betrachtet. Somit kann die konjunktive Anfrage Q und die Wissensbasis \mathcal{WB} in den äquivalenten PL1-Formalismus überführt werden. Darauf basierend wird eine konjunktive Anfrage über \mathcal{WB} in Begriffen der Prädikatenlogik definiert.

Definition 18: *Es sei eine konjunktive Anfrage $Q = \langle Q_h, Q_b \rangle$ und eine Wissensbasis \mathcal{WB} gegeben, dann ist die Menge sicherer Antworten folgendermaßen definiert:*

$$ans(Q, \mathcal{WB}) = \{ \vec{a} \mid \varpi(\mathcal{WB}) \wedge \sigma(Q_b) \models Q_h(\vec{a}) \},$$

wobei \vec{a} ein Tupel mit Konstanten, das in \mathcal{WB} vorkommt und $\varpi(\mathcal{WB})$, $\sigma(Q_b)$ entsprechende Transformation von \mathcal{WB} und Q_b in PL1-Formalismus ist.

Mit der Definition 18 ist es nun möglich, Probleme bei der Bearbeitung von konjunktiven Anfragen über die Wissensbasis \mathcal{WB} sowie die in diesem Zusammenhang stehende Entscheidungsprobleme, sogenannte Folgerungsprobleme (engl. *entailment*), zu betrachten.

- *Lösung einer konjunktiven Anfrage Q* : ist ein Problem hinsichtlich der Berechnung einer Menge von sicheren Antworten $\text{ans}(Q, \mathcal{WB})$ auf die konjunktive Anfrage Q über die Wissensbasis \mathcal{WB} .
- *Implikation einer konjunktiver Anfrage Q* : ist ein Problem bei dem entschieden werden soll, ob ein Tupel von Konstanten \vec{a} in $\text{ans}(Q, \mathcal{WB})$ bzgl. der konjunktiven Anfrage Q und der Wissensbasis \mathcal{WB} vorhanden ist.

2.5.2. Umschreiben konjunktiver Anfragen für \mathcal{DL}

Die im vorherigen Abschnitt aufgezeigten Probleme der Beantwortung von konjunktiven Anfragen über bestimmte \mathcal{DL} können mittels der Anfrageumschreibung (engl. *query rewriting*) gelöst werden. Intuitiv kann das Umschreibeverfahren auf folgende Art erläutert werden: Gegeben sind eine konjunktive Anfrage Q und eine TBox \mathcal{T} . Der Grundgedanke besteht aus der Berechnung einer Anfrage Q' – eine Umschreibung der Anfrage Q in Bezug auf \mathcal{T} –, sodass die Auswertung von Q über $\mathcal{WB} = (\mathcal{T}, \mathcal{A})$ für beliebige \mathcal{A} auf die Auswertung von Q' nur über \mathcal{A} reduziert werden kann. Dieser Ansatz ist aus praxisrelevanter Perspektive sehr interessant. Dadurch ist es möglich, traditionelle Datenbanksysteme zur Speicherung von \mathcal{A} einzusetzen und gleichzeitig Q' auszuwerten.

Formal wird das Umschreiben einer konjunktiven Anfrage Q in Bezug auf TBox \mathcal{T} wie folgt definiert:

Definition 19: Eine Anfrage Q' ist eine Umschreibung der konjunktiven Anfrage Q in Bezug auf TBox \mathcal{T} genau dann, wenn für jede ABox \mathcal{A} , $ans(Q, \mathcal{T} \cup \mathcal{A}) = ans(Q', \mathcal{A})$ gilt.

Das Ergebnis der Umschreibung einer konjunktiven Anfrage kann auch eine disjunktive Anfrage (*disjunctive query*) sein, da die Ergebnisse aus mehreren Datenquellen zusammengefasst werden müssen. Basierend auf der Definition 19 wird ein aktuelles Problem zum Umschreiben konjunktiver Anfrage für Beschreibungslogiken \mathcal{DL} vorgestellt.

- *Umschreiben konjunktiver Anfrage:* ist ein Problem der Berechnung konjunktiver Anfrage Q in Bezug auf TBox \mathcal{T} .

Im Kapitel 5 wird dieses Problem genauer betrachtet und ein Lösungsansatz erarbeitet.

2.6. Zusammenfassung

In diesem Kapitel wurden grundlegende Begriffe eingeführt und Ansätze samt der in diesem Kontext stehender Herausforderungen der Informationsintegration betrachtet. Aus Gründen der Übersichtlichkeit wurde das Gebiet der formalen Wissensrepräsentation in komprimierter Form vorgestellt und die für die vorliegende Arbeit relevanten Formalismen eingeführt. Weiterhin wurde der Begriff Ontologie eingeführt und der Zusammenhang zu den Beschreibungslogiken aufgezeigt. Der Schwerpunkt dieses Kapitels lag dabei auf der beschreibungslogischen Wissensbasis. In Bezug auf die Bereitstellung von Anfrageergebnissen aus einem wissensbasierten System wurden Inferenzprobleme für Beschreibungslogiken behandelt. Das Ziel dieses Kapitels war es die Problematik bei der Beantwortung konjunktiver Anfragen an eine Wissensbasis zu beleuchten und eine Möglichkeit zur Lösung dieses Problem basierend auf dem Umschreiben konjunktiver Anfragen aufzuzeigen.

Ein entscheidendes Ergebnis dieses Kapitels ist die Definition des Informationsintegrationssystems. Im Rahmen dieser Definition wurden wissensbasierte Systeme betrachtet. Es wurde festgestellt, dass Beschreibungslogiken einerseits entscheidbar sind, andererseits eine gute Balance zwischen der Ausdruckstärke der Sprache und der Komplexität von Inferenzmechanismen darstellen. Basierend darauf wurde einerseits ein mögliches Einsatzpotenzial von beschreibungslogischen Wissensbasen im Rahmen der Informationsintegration erkannt; andererseits, dass auf diesem Gebiet aktueller Forschungsbedarf besteht. Diese Erkenntnis erlaubt es nun, im folgenden Kapitel 3 relevante Ansätze, Verfahren und Technologien aus dem Bereich der Forschung und Industrie auszuwählen und zu betrachten.

3. Stand der Forschung und der Technik

Die Analysen der Teiltätigkeiten des Ingenieurs haben ergeben, dass insbesondere bei Aufgaben mit hohen Innovationsforderungen etwa 50 % der gesamten Arbeitszeit eines Ingenieurs auf Informationsbeschaffung und Wissenserwerb verwandt werden [KFG07]. Hierfür steht in der Produktentwicklung eine unüberschaubare Vielfalt und Menge an heterogenen Datenquellen zur Verfügung, deren vollständige Auswertung nicht möglich ist. Die fehlende Interoperabilität zwischen den Datenbeständen untereinander und zum Anwender hin erschwert oder verhindert einen ordentlichen Informationsaustausch. Dies führt dazu, dass relevante Information bzw. das Wissen ungenutzt bleibt oder gar verloren geht.

Die komplexe Vernetzung von Informationen im Produktentwicklungsprozess findet derzeit keine geeignete Abbildungsstruktur. Produktentwickler haben mit unbefriedigenden Informationsstrukturen zu kämpfen, die langsame und komplizierte Informationsflüsse erzeugen. Redundante Datenbestände verursachen einen hohen Aufwand im Zusammenhang mit deren Abgleich und Aktualisierung. Folglich beschäftigen sich seit längerer Zeit viele Unternehmen und Forschungseinrichtungen mit der problemspezifischen Bereitstellung von Informationen und Wissen aus heterogenen Datenbeständen zur richtigen Zeit, ohne dabei den Empfänger mit einer Informationsflut zu überschwemmen. In diesem Kontext wurden zahlreiche Integrationslösungsansätze erarbeitet, die von einer einfachen Schnittstellenimplementierung bis hin zu komplexen Standard-Datenmodellen reichen.

3.1. Allgemeine IT-Integrationsansätze

3.1.1. Punkt-zu-Punkt Verbindungen

Unter einer Punkt-zu-Punkt-Verbindung versteht man eine dedizierte Verbindung zwischen jedem Paar von Anwendungssystemen, die miteinander kommunizieren wollen [ES09]. Zur Integration einer Vielzahl von Anwendungssystemen sind entsprechend zahlreiche solcher Punkt-zu-Punkt-Verbindungen zwischen den jeweiligen Anwendungspaaren erforderlich [WRW01]. Bei der Erstellung von Punkt-zu-Punkt-Verbindungen muss der Entwickler über eine weitreichende Kenntnis der Programmlogik und des internen Ablaufs des Quell- und Zielsystems verfügen [Gou99].

Kurzfristige Vorteile der P2P-Integration wie die schnelle Realisierung benötigter Schnittstellen oder die individuelle Gestaltungsmöglichkeit einzelner Schnittstellen, werden auf lange Sicht

durch die Nachteile kompensiert [ABM06]. Die wesentlichen Nachteile dieses Ansatzes sind [WRW01]:

- hoher Betriebsaufwand für Wartung und Instandhaltung der Schnittstellen
- hoher Aufwand bei der Integration zusätzlicher Anwendungen
- invasive Programmierung, d.h.: zur Realisierung der Schnittstellen wird in die einzelnen IT-Systeme eingegriffen
- nicht optimale Nutzung der Ressourcen
- inhomogene und inkonsistente Daten
- zeitliche Verzögerung bei der Datenbereitstellung

Angesichts dieser Situation haben sich damals Unternehmen sowie Fachleute zum ersten Mal Gedanken über effiziente und nachhaltige Integrationsansätze gemacht und die neue Disziplin „Systemintegration“ (engl. *Integration Engineering*) etabliert, die seitdem mehrere Evolutionsstufen hinter sich gebracht hat [ES09]. Viele Unternehmen setzen gegenwärtig – trotz zahlreicher kommerzieller Lösungen – nach wie vor Punkt-zu-Punkt-Verbindungen zur Koppelung ihrer IT-Systeme ein. Der Grund dafür ist, dass hauseigene Realisierungen von Schnittstellen-Lösungen in der Regel preiswerter sind [Aie06].

3.1.2. Middleware-basierte Integration

Die Middleware-basierte Integration verwendet zur Lösung des Integrationsproblems eine vermittelnde Softwareschicht, die sogenannte Middleware, die zwischen zwei oder mehrere Systeme geschaltet wird [Kai02]. Diese ermöglicht den angebotenen Anwendungen einen hersteller- und evtl. plattformunabhängigen Datenaustausch und schafft somit die Konnektivität zwischen verschiedenen Anwendungen [Aie06]. Die semantische Abstimmung zwischen den Anwendungen oder ein Prozessmanagement leistet Middleware nicht [WRW01].

Middleware wird vielfach in Punkt-zu-Punkt-Systemlandschaften eingesetzt, um zwei Anwendungen miteinander zu verbinden [ABM06]. Der Vorteil von Middleware gegenüber der klassischen P2P-Verbindung liegt in der standardisierten Form [Aie06]. Dies verringert den Integrationsaufwand, setzt jedoch gleichzeitig voraus, dass sich alle beteiligten Systeme bzw. Applikationen dem gewählten Standard unterwerfen [Aie06]. Insbesondere bei Altsystemen ergeben sich hier Probleme aufgrund fehlender Middleware-konformer Schnittstellen [SM01].

Im Nachfolgenden werden Standards und Technologien vorgestellt, die den beschriebenen Aufbau eines Informationssystems sowie den übergreifenden Aufruf von Diensten unterstützen.

Common Object Request Broker Architecture (CORBA)

Die *Common Object Request Broker Architecture* (CORBA) ist ein offener Standard der Object Management Group (OMG)¹ für eine Middleware Plattform (siehe Abb. 3.1). Die OMG ist ein internationales Konsortium aus mehr als 800 Mitgliedern aus dem IT-Gewerbe. CORBA agiert als sogenannter Middleware-Vermittler zwischen Client und Server. Objekte (Client und Server) kommunizieren über den Object Request Broker (ORB)² miteinander. Hierzu unterstützt CORBA sprachunabhängige Schnittstellendefinitionen von Komponenten und die transparente Verteilung von Objekten [Mer02]. Durch den ORB als Vermittler wird in CORBA eine Sprachen- und Ortsunabhängigkeit geschaffen, da der Client unbeeinflusst von System und Architektur bleibt, auf denen der Server läuft. Diese Trennung wird erreicht, indem Objekte nur über Schnittstellen kommunizieren, die der ORB ihnen bietet. Diese Schnittstellen werden in einer deskriptiven Sprache (IDL) beschrieben.

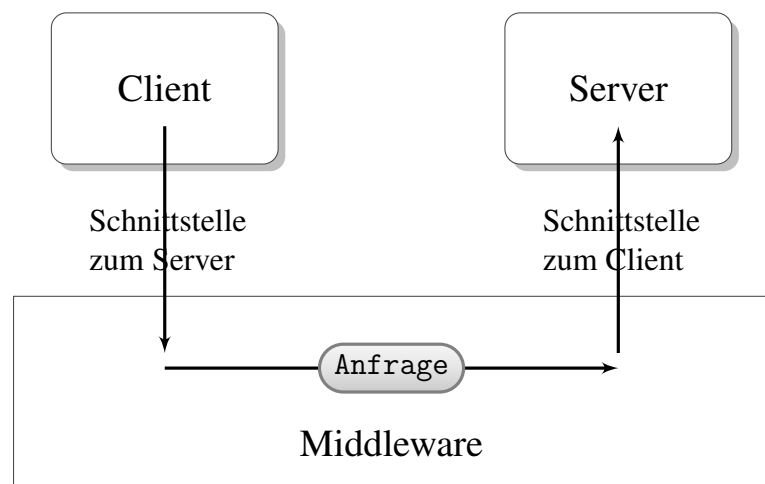


Abbildung 3.1.: CORBA Architektur [MR97]

Auf jedem Rechner, den ein CORBA-basiertes Informationssystem nutzt, ist ein CORBA-konformer ORB implementiert. Ein Dienst auf einem entfernten Rechner wird aufgerufen, indem eine den Dienst repräsentierende Schnittstelle, das sogenannte *stub interface*, aufgerufen wird [Eng05]. Das *stub interface* leitet den Aufruf an den lokal installierten ORB weiter, der das sogenannte *skeleton interface* des Dienstes aufruft. Durch den ORB des aufrufenden Dienstes wird letztendlich das Resultat zurückgeliefert [BB04].

Um Dienste unterschiedlicher Plattformen aufrufen zu können, wurde das *General Inter-ORB-Protocol* (GIOP) entwickelt [MR97]. Es definiert allgemeine Protokollspezifikationen wie die Darstellung von Datentypen. Darauf basierend wurden seitens OMG Implementierungsregeln

¹<http://www.omg.org>

²Object-Request-Broker (ORB) stellt innerhalb der Middleware eine zentrale Vermittlungsschicht dar, die den Aufruf von Methoden entfernter Objekte über Netzwerk-, Programmiersprachen- und HW/SW-Plattformgrenzen hinweg für aufrufende Objekte möglich macht [Kar12].

für TCP/IP entwickelt, die im Internet-Inter-ORB-Protokoll (IIOP) zusammengefasst wurden [Say99]. Das IIOP behandelt Unterschiede zwischen ORB-Implementierungen wie die Byte-Reihenfolge und -Anordnung auf Maschinenebene [FC99].

CORBA verliert zunehmend an Bedeutung, weil einerseits die Schutzeinrichtungen gegen Rechner (sog. Firewalls) zum größten Teil den Einsatz von CORBA einschränken, andererseits aufgrund des hohen Entwicklungsaufwands, das einfach durch die Anwendung anderer Technologien wie Webservices vermieden werden kann.

Webservices

Ein Webservice ist eine über ein Netzwerk zugängliche Schnittstelle zu Informationssystemfunktionen, die mithilfe von Standardtechniken des Internets realisiert wird [STK02]. Die folgenden drei Technologien bilden das Fundament von Webservices [Kil10]:

- *Simple Object Access Protocol* (SOAP): Ein Standard, der ein XML-basiertes Protokoll spezifiziert, welches synchrone oder asynchrone Interaktion zwischen verteilten Komponenten unterstützt.
- *Webservices Description Language* (WSDL): Ein Standard, der eine Sprache zur Beschreibung von Webservice-Schnittstellen spezifiziert.
- *Universal Description, Discovery and Integration* (UDDI): Eine XML-basierte Registrierungsdatenbank, die es Unternehmen ermöglicht, Informationen über sich und ihre angebotenen Webservices zu veröffentlichen.

Webservices sind das Resultat von Forschung und Entwicklung von Microsoft, IBM und Ariba der Jahre 1998 bis 2000. Später wurden diese Ergebnisse vom W3C-Komitee³ und von UDDI.org, einer unabhängigen Industrieinitiative, übernommen.

Die Abbildung 3.2 zeigt das Zusammenspiel zwischen den drei oben gezeigten Technologien, die das der dynamischen Integration zugrunde liegende Prinzip veranschaulicht [STK02]. Ein Dienstanbieter veröffentlicht eine Beschreibung der von ihm angebotenen Dienste, indem diese bei einem Dienstmakler registriert werden. Der Dienstanbieter durchsucht die Dienstmakler-Registry nach einem Dienst, der seinen Anforderungen entspricht [Eng05]. Als Dienstanbieter kann dabei ein menschlicher Benutzer oder auch ein Programm agieren. Der Gebrauch eines Webservices seitens Dienstanbieter ist in der Abbildung 3.2 durch die Verbindung *Nutzung* gekennzeichnet.

³<http://www.w3c.org>

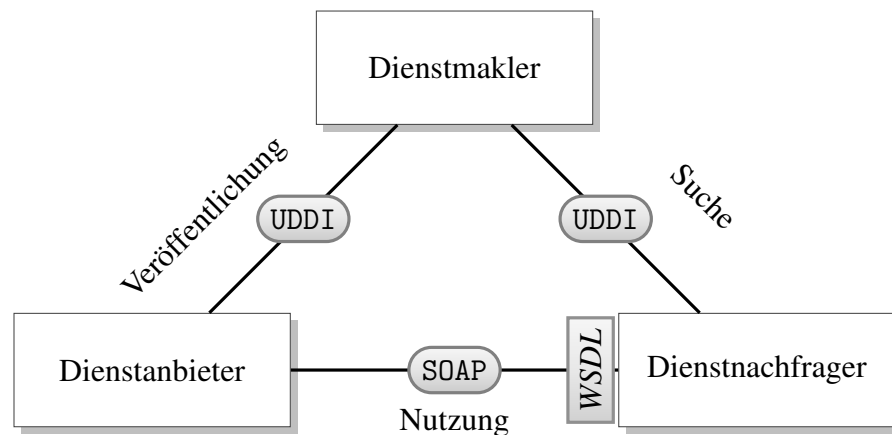


Abbildung 3.2.: Webservice-Architektur [STK02]

Die Webservice-Technologie gewinnt im industriellen Einsatz zunehmend an Bedeutung [WJ06]. Die Plattform- und Programmiersprachenunabhängigkeit erlaubt einen breiten Einsatz dieser Technologie. Darüber hinaus bieten Webservices die Möglichkeit, verteilte Systeme zu realisieren [BH03]. Allerdings gehen dabei die Meinungen bezüglich der Eignung dieser Innovation in diesem Aufgabengebiet weit auseinander [CJ02], [BH03].

Der Nachteil von Webservices liegt vor allem in der *Zustandslosigkeit* jeder Verbindung. Dadurch ist es z. B. sehr schwierig, Webservices mit transaktionaler Semantik anzusprechen bzw. an verteilten Transaktionen zu beteiligen [LN07]. Zwar existieren dazu Vorschläge z. B. die *Webservices Coordination and Transaction-Spezifikation* von IBM, bisher konnte sich aber kein durchsetzen [RLSR⁺06]. Darüber hinaus sorgt die Umhüllung jeder Nachricht in XML für einen erheblichen Leistungsverlust, vor allem für Applikationen mit sehr hohem Nachrichtendurchsatz.

.NET Framework

Das .NET Framework der Firma *Microsoft* ist ein innovativer komponentenbasierter Ansatz zur Entwicklung von Anwendungen. Bei diesem Ansatz erfolgt die Integration von Objekten (u. a. heterogener Natur) mittels der Schnittstellen, die diese Objekte oder Teile von Programmen als unabhängige Komponenten darstellen [BBM⁺06]. Auf diese Weise wird die Entwicklung und das Zusammenwirken von Softwarekomponenten in einer heterogenen Umgebung erleichtert. Ein wesentlicher Vorteil ist die Möglichkeit der praktischen Umsetzung des Prinzips „Jede Entität ist ein Objekt“ in einer heterogenen Softwarelandschaft. Dies ist vor allem auf das verbesserte *Common Type Systems* (CTS) zurückzuführen [Cou10].

Die grundlegende Technologie zur Sicherstellung der Skalierbarkeit und Interoperabilität von Anwendungen ist im .NET-Framework die Webservice-Technologie. Die Skalierbarkeit ist durch die Verteilung von Rechnerressourcen zwischen dem Server, worauf beispielsweise die Da-

ten gespeichert sind, und dem Benutzerrechner gewährleistet. Die Möglichkeit der integrierten Verarbeitung von heterogenen Daten aus unterschiedlichen Anwendungen sorgt für die Interoperabilität. Daher werden Webservices von der Firma Microsoft als eine der Kernpunkte der .NET-Technologie bezeichnet. Damit wird im .NET Framework die gleiche Technologie für den internen wie den externen Aufruf von Diensten verwendet [BBM⁺06].

Das .NET Framework ist eine Art Aufsatz auf einem Betriebssystem und bietet einige Werkzeuge zur Gestaltung und Implementierung von Software. Aktuell umfasst das .NET Framework:

- vier offizielle Programmiersprachen: C#, VB.NET, Managed C++ und JScript.NET
- objektorientierte Laufzeitumgebung *Common Language Runtime* (CLR)
- einer Reihe von untereinander in Beziehung stehenden Klassenbibliotheken unter dem Namen *Framework Class Library* (FCL)

Die wesentlichen Bestandteile der Architektur des .NET Frameworks sind in der Abbildung 3.3 dargestellt. Eine der wichtigsten Komponenten des .NET Frameworks ist die Laufzeitumgebung CLR, in der Programme ausgeführt werden. Die Hauptaufgabe von CLR besteht darin, die Datentypen von .NET zu erkennen, zu laden und gemäß dem empfangenen Befehl zu steuern [Cou10]. CLR enthält eine virtuelle Maschine, in vieler Hinsicht ähnlich einer *Java Virtual Machine*.

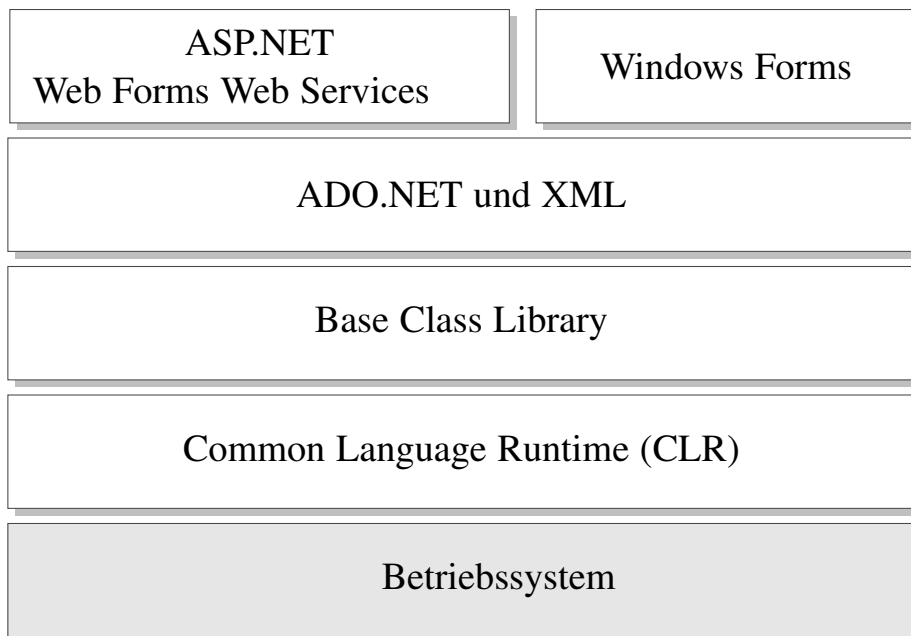


Abbildung 3.3.: Architektur des .NET Frameworks [Cou10]

Oberhalb der CLR befindet sich ein Satz von Basisklassen der Plattform *Base Class Library*. Darüber ist eine Schicht von Datenklassen und XML sowie eine Schicht von Klassen zum Erstellen von Webservices, Web- und Windows-basierten Anwendungen (*Web Forms* und *Windows Forms*). Die Sammlung dieser Klassen ist unter der Bezeichnung *Framework Class Library* (FCL) bekannt. Dies ist eine der größten Klassenbibliotheken in der Programmierungsgeschichte [BBM⁺06].

Das .NET Framework unterstützt die Kommunikation zwischen verschiedenen Informationssystemen. Hierbei ist durch die Fokussierung auf Webservices eine hohe Flexibilität bei der Realisierung von Schnittstellen gewährleistet, allerdings nur sofern beide Systeme die Webservice-Technologie unterstützen. Die Implementierung einer Schnittstelle und insbesondere die Datentransformation an einer Schnittstelle ist bei .NET eine projektindividuelle Tätigkeit mit hohem Aufwand.

3.1.3. Enterprise Application Integration (EAI)

Enterprise Application Integration (EAI) umfasst Standards und Lösungen zur Integration einer Vielzahl unterschiedlicher, heterogener Anwendungen, die unabhängig voneinander entwickelt worden sind, sodass diese für einen Benutzer möglichst einheitlich aussehen (vgl. [ABM06]). Die Kernidee von EAI ist es, eine zentrale Plattform bereitzustellen, die Applikationen über entsprechende, zum Teil vorgefertigte Adapter anbindet [ABM06]. BUHL, CHRIST und PAPE grenzen EAI als Integration von Anwendungen über unterschiedliche technische und logische Infrastrukturen hinweg ab [BCP01]. Der zentralisierte Integrationsansatz des EAI wird u. a. als Rückgrat der unternehmensweiten Informationsverarbeitung bezeichnet [BCP01]. Die Anzahl der Schnittstellen zwischen Anwendungskomponenten kann aufgrund der zentralisierten EAI-Struktur von $\frac{n*(n-1)}{2} O(n^2)$ auf $O(n)$ reduziert werden [LKS99]. Dabei bezeichnet n die Anzahl der zu integrierenden Schnittstellen der Anwendungssysteme und $O(n)$ in Anlehnung an die Komplexitätstheorie die Abschätzung des Aufwandes des Integrationsproblems [Mar03].

EAI umfasst zwei Gesichtspunkte: zum einen die zentrale Komponentenvermittlung der Anwendersysteme, zum anderen zentrale Ablaufsteuerungen. Beide Aspekte zusammen führen zu einer Hub-and-Spoke-Architektur, bei der eine Verbindung zwischen zwei Knoten über einen Zentralknoten geführt wird [Kop01]. Neben dem zentralisierten Ansatz zur Kopplung von Anwendungssystemen werden dezentrale Ansätze diskutiert. Als weitverbreitete Vertreter dieses Softwaretyps werden im Folgenden *Microsoft BizTalk* und *IBM WebSphere* kurz vorgestellt:

Microsoft BizTalk

BizTalk war ursprünglich eine Bibliothek von XML⁴ Schemata und vorgefertigten XML Dokumenten, die den elektronischen Handel und die Integration von Applikationen ermöglichen sollte [And00]. Das BizTalk-Framework wurde von der Firma Microsoft in Zusammenarbeit mit führenden Organisationen aus einer Vielzahl von Branchen ins Leben gerufen, um die Erstellung und Wartung von XML-Datenschemata für den elektronischen Handel und die Anwendungsintegration zu fördern [And00], [Eng06].

Das BizTalk-Framework spezifiziert eine Kernmenge von XML-Literalen, die in XML-Dokumenten verwendet werden sollen [Lof07]. Diese Spezifikation ist notwendig für die Konzeption und Entwicklung von Softwarelösungen auf Basis von XML, um die Interaktion zwischen den Anwendungen mit Standard-Internettechnologien zu etablieren. XML-Dokumente, die der BizTalk-Framework-Spezifikation genügen, werden auch als BizTalk-Dokumente bezeichnet.

Der BizTalk-Server ist eine Implementierung der Firma Microsoft des BizTalk-Frameworks für die Windows-Plattform [And00]. Er stellt die Technologie dar, die auf dem BizTalk-Framework basierend den Austausch von Dokumenten zwischen Geschäftspartnern unterstützt [Lof07]. Dieser Prozess wird im Folgenden näher beschrieben (siehe Abb. 3.4):

1. Der *Receive Adapter* erhält eine Nachricht durch einen der vielen verfügbaren Protokolle von der Außenwelt.
2. Die Nachricht wird durch *Receive Pipeline* (mit den Möglichkeiten der Transformation und Validierung) weitergeleitet.
3. Die Nachricht gelangt in die *Message Box* – eine Nachrichtendatenbank auf Basis von MS SQL Server.
4. Ein Agent erfasst den Nachrichteneingang und übergibt diese der *Orchestration*.
5. Die *Orchestration* führt notwendige Bearbeitungsschritte zur Verarbeitung der Nachricht durch und erzeugt eine neue Nachricht, die an die *Message Box* zurückgesendet wird. In der Regel unterscheidet sich die Struktur der neuen Nachricht von der Struktur der ursprünglichen Nachricht.
6. Ein Agent stellt erneut den Nachrichteneingang in der *Message Box* fest und sucht nach dem entsprechenden Nachrichtenempfänger. In diesem Fall ist es der *Send Adapter*.
7. Die Nachricht wird an den *Send Adapter* in Verbindung mit der *Send Pipeline* übergeben, sodass bei Bedarf die Nachricht validiert, transformiert und an die Außenwelt mittels verfügbarer Protokolle versendet wird.

⁴Extensible Markup Language, eine Sprache zur Darstellung hierarchisch strukturierter Daten in textueller Form [W3C12]

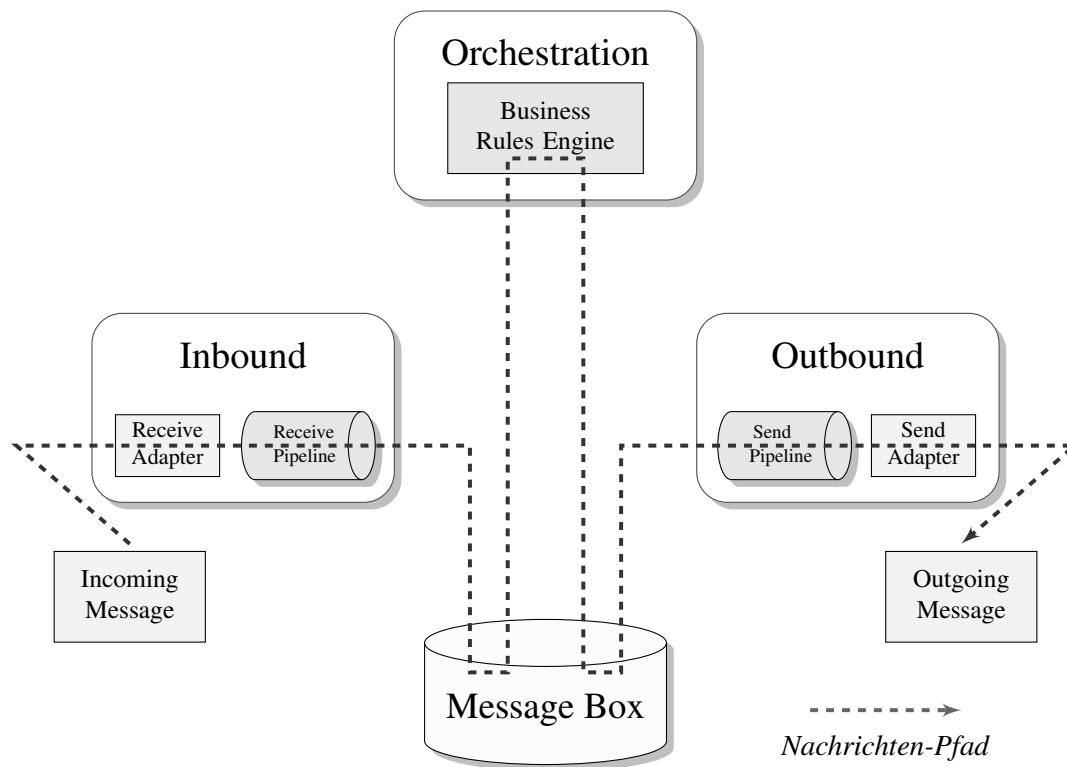


Abbildung 3.4.: BizTalk Server Architektur nach [Cha07]

BizTalk ist stark informationsorientiert und verfügt über kein Prozessbewusstsein [Lof07]. Darüber hinaus unternimmt das BizTalk-Framework keine Anstrengungen, um Handelspartner eine bestimmte Geschäftssemantik aufzuzwingen [And00].

IBM WebSphere

Die Firma IBM bietet mit *WebSphere Business Integration* ein Lösungsportfolio für Integrationsanforderungen wie Anwendungsintegration und Geschäftsprozessintegration [Eng06]. WebSphere stellt eine „*business on demand*“-Strategie zur Verfügung; dadurch werden Geschäftsprozesse sowohl innerhalb des Unternehmens als auch mit wichtigen Partnern, Lieferanten und Kunden integriert. Hierbei ist es möglich, Geschäftsprozesse schnell auf alle Anforderungen seitens des Benutzers, an die Erweiterungen des Marktes sowie an externe Einflüsse anzupassen. Darüber hinaus kann WebSphere zum Aufbau und Monitoring der Infrastruktur eingesetzt werden, mit dem Ziel einer besseren Unterstützung von Geschäftsprozessen sowie dem Ausbau eigener Anwendungen, die in dieser Infrastruktur agieren [Ebe03]. Die Abbildung 3.5 zeigt die Möglichkeiten von WebSphere um Anwendungen integrieren, automatisieren und optimieren zu können [SBF⁺08].

- Die *Integration von Menschen* ermöglicht es den Kunden und Geschäftspartnern, mit Anwendungen, Geschäftsprozessen und Informationen jederzeit und überall zu interagie-

ren. Durch die Verwendung dieser Features können beispielsweise die Funktionen von Callcentern automatisiert werden.

- Die *Integration von Prozessen* ermöglicht es, Geschäftsprozesse zu modellieren, zu überwachen und gemäß den strategischen Zielen eines Unternehmens zu optimieren. Dadurch kann z. B. ein Hauptprozess modelliert werden, um diesen im Anschluss zu simulieren, ggf. zu verbessern und in Betrieb nehmen zu können.
- Die *Informationsintegration* ermöglicht eine konsistente und einheitliche Sicht auf strukturierte und unstrukturierte Informationen aus heterogenen Quellen sowie diese Information zu verwalten und zu synchronisieren. Dadurch ist beispielsweise die Suche über alle Informationsquellen, einschließlich Dateisysteme, Collaboration-Portale und Webseiten möglich.
- Die *Integration von Anwendungen* bietet eine breite Palette von Services, um zuverlässige und flexible Informationsflüsse innerhalb der Anwendungen sicherstellen zu können. Dadurch kann z. B. der Nachrichtenaustausch zwischen Anwendungen erfolgen.
- Die *Applikationsinfrastruktur* von WebSphere ermöglicht die Erstellung, Implementierung, Integration und Verbesserung neuer und bestehender Anwendungen.
- *Accelerators* – vorgefertigte Lösungen, mit deren Hilfe der Mangel an Integrationserfahrung kompensiert werden kann. Accelerators sorgen für eine solide Grundlage für zukünftige Erweiterungen.

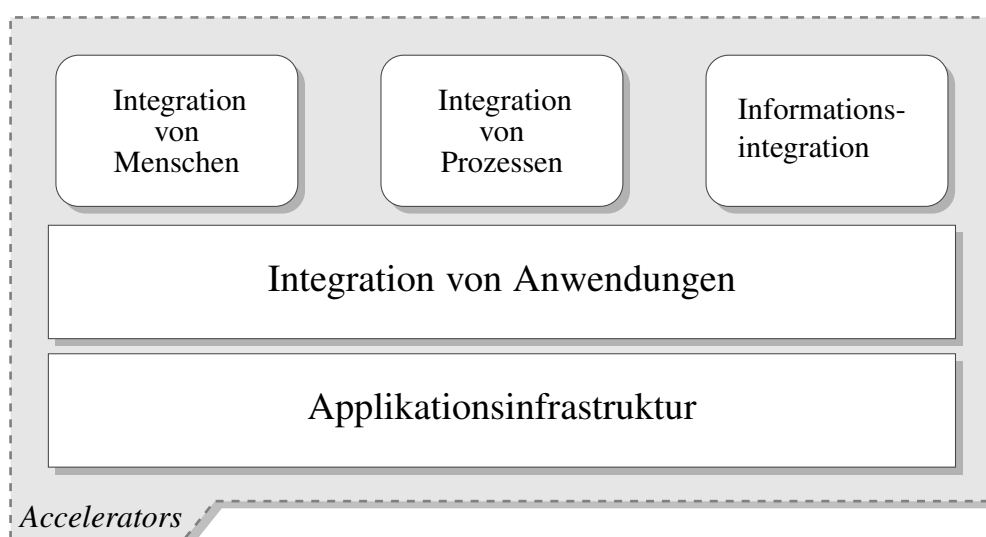


Abbildung 3.5.: Möglichkeiten von IBM WebSphere nach [SBF⁺08]

Im Gegensatz zu BizTalk ist jedoch nicht eine Bibliothek mit vorgefertigten Datenformaten die Grundlage für das Lösungsportfolio, sondern in unterschiedlichen Projekten entstandene Softwarelösungen [Eng06]. Der Schwerpunkt der IBM WebSphere Business Integration ist die unternehmensweite Integration, wobei die Realisierung von Schnittstellen eines nicht von IBM unterstützten Systems nicht mit Prozessmodellierung verbunden ist. Folglich muss diese mit hohem Aufwand verbundene Tätigkeit individuell umgesetzt werden.

3.1.4. Serviceorientierte Architektur (SOA)

SOA ist keine neue Technologie, sondern vielmehr ein Konzept, um verteilte IT-Services in Geschäftsanwendungen einzubinden oder diese anzubieten [FZ09]. Der Kerngedanke serviceorientierter Architekturen liegt darin, unterschiedliche Systeme über Services miteinander zu koppeln und dadurch neue Anwendungen mittels Komposition zu entwickeln [FZ09]. Es können sowohl bestehende als auch neu entwickelte Systeme integriert werden [FZ09]. Zurzeit existiert noch keine einheitliche Definition einer SOA [Mel10]. Bei allen momentan gebräuchlichen Definitionen bestehen zwar immer wieder Überlappungen, es fehlen allerdings häufig in einer Definition Aspekte, die von einer anderen Definition als entscheidend angesehen werden [Mel10]. MELZER schlägt die folgende Definition für SOA vor [Mel10]:

Definition 20: *Unter einer SOA versteht man eine Systemarchitektur, die vielfältige, verschiedene und eventuell inkompatible Methoden oder Applikationen als wiederverwendbare und offen zugreifbare Dienste repräsentiert und dadurch eine plattform-und sprachenunabhängige Nutzung und Wiederverwendung ermöglicht.*

Die Abbildung 3.6 stellt das Grundkonzept der SOA, den sogenannten SOA-Tempel dar. Der Grundstein setzt sich aus offenen Standards, Sicherheit und Einfachheit zusammen. Die verteilten Dienste, die lose Kopplung, die Plattformunabhängigkeit und die prozessorientierte Struktur sind die tragenden Säulen von SOA [Mel10].

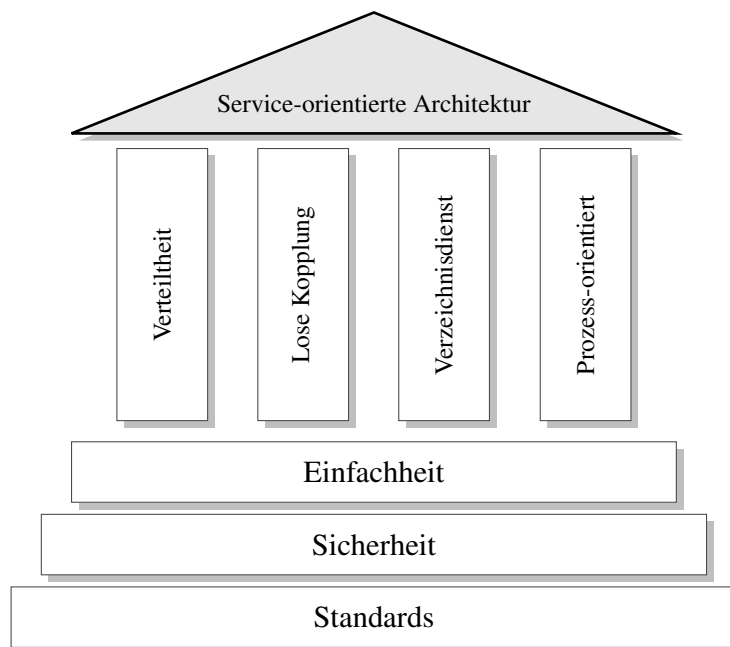


Abbildung 3.6.: SOA Tempel [Mel10]

Eine Übersicht über die grundlegenden Merkmale einer SOA zeigt die Tabelle 3.1 [Mel10]. Für den Einsatz in einer serviceorientierten Architektur sind Webservices aufgrund ihres Konzepts und der Standards optimal geeignet [FZ09]. Der Aufbau einer serviceorientierten Architektur richtet sich in erster Linie nach den etablierten Geschäftsprozessen eines Unternehmens. Hierzu werden Prozesse in der standardisierten Sprache BPEL (*Business Process Execution Language*) beschrieben. BPEL basiert auf XML und somit auf der gleichen technologischen Grundlage wie Webservices [FZ09]. Der Sprachstandard BPEL wurde von dem OASIS-Komitee verabschiedet [OAS12].

Merkmale	Beschreibung
<i>Lose Kopplung der Dienste</i>	Dienste werden von Anwendungen oder anderen Diensten bei Bedarf dynamisch gesucht, gefunden und eingebunden.
<i>Dynamisches Binden von Funktionalitäten</i>	Es handelt sich um eine Bindung zur Laufzeit. Das heißt, dass zum Zeitpunkt der Übersetzung des Programms in der Regel nicht bekannt ist, wer oder was zur Laufzeit aufgerufen wird. Durch Benutzerpräferenzen oder externe Einflüsse kann es sogar möglich sein, dass die Wahl des aufgerufenen Dienstes nicht unter der Kontrolle der Anwendung ist.
<i>Verzeichnisdienst oder Registry</i>	Zur Verfügung stehende Dienste werden im Verzeichnis oder Registry eingetragen. Dieser gestattet die Suche nach Methoden, die von einer Anwendung gerade benötigt werden.
<i>Verwendung von Standards</i>	Damit nach einer erfolgreichen Suche die Nutzung des gefundenen Dienstes möglich ist, muss der Aufrufer in der Lage sein, sich mit diesem zu unterhalten. Die essenzielle Forderung dazu ist, dass alle Schnittstellen in maschinenlesbarer Form beschrieben sind. Eine wichtige Voraussetzung ist, dass offene Standards genutzt werden, damit der Nutzer den Dienst eines unbekanntem Anbieters auch verstehen kann.
<i>Einfachheit & Sicherheit</i>	Die Einfachheit einer SOA erfüllt viele Anforderungen und wird eine Umsetzung schnell vorantreiben. Es ist wichtig, den Aspekt Sicherheit im Kontext eines SOA-Ansatzes von Anfang an zu betrachten. Genau genommen kann festgestellt werden, dass Sicherheit kein eigentliches Merkmal einer SOA ist, sondern dass Einfachheit, Sicherheit und Akzeptanz notwendige Voraussetzungen für eine SOA sind, wobei der letzte Punkt durch die anderen beiden bedingt wird.

Tabelle 3.1.: Merkmale einer SOA [Mel10]

Die grundlegende Idee von serviceorientierter Architektur ist nicht neu und die verschiedenen Bestandteile können als Erweiterungen von bereits bestehenden Techniken aufgefasst werden [Mel10]. Hierzu wird SOA von allen aktuell relevanten EAI-Plattformen der marktführenden Hersteller wie Microsoft .NET, IBM WebSphere, Oracle SOA Suite unterstützt. Diese Plattformen verwenden den Sprachstandard BPEL und besitzen jeweils eine Laufzeitumgebung für BPEL Prozesse, BPEL Process Engine sowie grafische Prozess-Modellierungswerkzeuge

[FZ09]. SOA ist in den letzten Jahren äußerst populär geworden und gilt als die zukünftige Basistechnologie zur Integration von Anwendungen. Wie EAI adressieren die mit Webservices verbundenen Technologien vornehmlich die Integration von Prozessen und nicht die von Daten [LN07].

3.1.5. Semantische Technologien

Semantische Technologien stellen eine Grundlage dar, den Umgang mit dem Problem der Informations- und Anwendungsintegration zu erleichtern [SSFH08]. Hierzu versprechen semantische Technologien neben der Integration von heterogenen Informationsquellen eine höhere Flexibilität bei der Modellierung sowie die Ableitung von Wissen (Reasoning) im Vergleich zu datenbankgestützten Ansätzen [VSA12]. Das Hauptaugenmerk liegt dabei auf der Sicherstellung der Interoperabilität von dispersen Datenbeständen und Anwendungen auf technischer, organisationaler und semantischer Ebene, ohne dass die Autonomie der einzelnen Teilsysteme aufgehoben wird [BT06]. Durch die vereinheitlichte Konzeptualisierung von Prozessen, Daten und den Geschäftslogiken können einzelne Softwareapplikationen u. U. Alt-Systeme (sog. *Legacy-Systeme*) gekapselt und auf diese Weise mit anderen Anwendungen gekoppelt werden [Joh06]. Ein weiteres wichtiges Einsatzgebiet ist das Auffinden von neuen relevanten Dokumenten im Intranet und Internet sowie die Reduktion von Medienbrüchen bei der Informationssuche in Unternehmen [Bai08]. Hierfür müssen Informationen in einer gleichermaßen maschinenlesbaren als auch dem Menschen verständlichen Form gehalten werden, um sowohl automatische Auswertungen als auch eine nutzerfreundliche Präsentation der Daten auf der ständig steigenden Anzahl von Ressourcen zu unterstützen [Joh06].

Genau an dieser Stelle setzt die Vision des *Semantic Web* ein, die von TIM BERNERS-LEE im Jahr 2001 formuliert wurde. Die Grundidee des Semantic Web ist die Evolution des Web hin zu einem Informationssystem, in dem verfügbare Informationen wesentlich besser als heute durch Programme benutzt werden können [LN07]. Die Grundtechnik zur Erreichung dieses Ziels beruht auf der Erweiterung des herkömmlichen Web um Zusatzinformationen (sog. *Metadaten*). Dabei basiert das Semantic Web auf zwei einfachen Grundprinzipien: Zum einen werden die im World Wide Web verfügbaren Ressourcen wie Webseiten, Bilder oder Audiodateien über einen eindeutigen Bezeichner identifiziert, zum anderen werden eben diese Ressourcen semantisch annotiert [Joh06]. Viele Bausteine, die bei der Realisierung dieses Ansatzes helfen sollen, wurden in den letzten Jahren durch das W3C standardisiert [LN07]. Ordnet man diese Bausteine nach der zunehmenden Abstraktionsebene, so erhält man die folgende Schichtung von Techniken und Sprachen (Abb. 3.7) [BLHL01]: Auf der untersten Schicht sorgen zum einen *Unicode*-Standards für eine einheitliche Kodierung von Zeichen, zum anderen der URI-Standard für eindeutige Identifikation von Ressourcen. Auf der zweiten Schicht sorgen XML, Namespaces

und XML Schema für den Austausch strukturierter Informationen auf Basis einer einheitlichen Syntax. Auf der dritten Schicht befindet sich das graphbasierte Datenmodell RDF (*Resource Description Framework*), mit dessen Hilfe Aussagen über Werte von Eigenschaften und von Ressourcen festgehalten werden können⁵. Auf der nächsten Schicht sorgt *Ontology Web Language* (OWL) für die Interoperabilität verschiedener RDF(S)-Dokumente und erweitert RDF um weitere ausdrucksstärkere Sprachkonstrukte⁶. Auf der fünften und sechsten Schicht wird die Wissensrepräsentation um beschreibungslogische Inferenzformalismen ergänzt. Da es keinen speziellen bzw. strikten Unterschied zwischen den Begriffen *Logic* und *Proof* im Semantic Web gibt, betrachtet man diese zwei Schichten daher gemeinsam [Pas04], [AvH08]. Die siebte Schicht sorgt für technische Vorkehrungen zur Bildung von Vertrauen. Hierfür kommen unterschiedliche Techniken wie digitale Unterschriften und asymmetrische Schlüssel zum Einsatz.

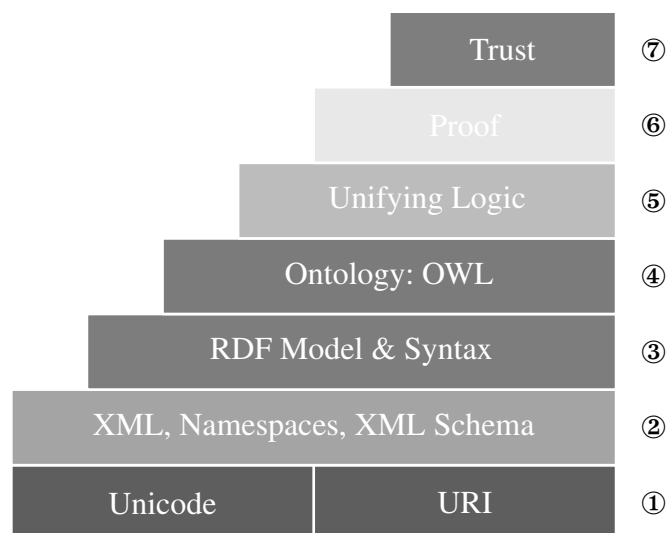


Abbildung 3.7.: Das Semantic Web in Schichten nach TIM BERNERS-LEE [BLHL01]

Eine vom Zentrum für Europäische Wirtschaftsforschung GmbH (ZEW) und vom Fraunhofer-Institut für System- und Innovationsforschung (Fraunhofer ISI) durchgeführte Studie [Bai08] über den Einsatz von semantischen Technologien im Wissensmanagement zeigt, dass Unternehmen heute semantische Technologien überwiegend zur Verbesserung ihrer Innovations- und Wettbewerbsfähigkeit einführen. Konkret sind es die in Folgendem aufgezählten Punkte, für die semantische bzw. Semantic-Web-Technologien nützlich sein können [AMN⁺06], [ST06], [Joh06], [Bai08], [Nie12]:

- ermöglicht ein schnelleres und leichteres Finden von relevantem Wissen, das in den Daten kodiert ist

⁵vgl. hierzu die RDF- und RDFS-Beschreibung in Anhang A

⁶Eine detaillierte Auflistung der OWL-Sprachkonstrukte kann dem Anhang B entnommen werden

- die Zugriffszeit auf Daten verkürzt sich und der damit verbundene Aufwand ist vergleichsweise gering
- Sichtbarmachung von versteckten und impliziten Zusammenhängen zwischen Daten
- durch semantische Suche angezeigte Treffer können auch visuell nach Fachgebieten strukturiert angezeigt werden
- Möglichkeit auf eine aktuelle Datenbank anstatt auf mehrere zurückzugreifen, was die Gefahr von Doppelarbeit reduziert
- Möglichkeit einer inhaltlichen Integration und somit eines einheitlichen Zugriffs auf verteilte Anwendungen
- Produktivitätssteigerung von bis zu 50 % durch die Zeitersparnis

Die Entwicklung der semantischen Technologie hat einen Zustand erreicht, der so weit gefestigt ist, dass er durchaus in den Unternehmen eingesetzt werden kann [Bai08]. Es bleibt aber anzumerken, dass semantische Technologien nur mit einem gewissen Aufwand insbesondere in der Initialphase betrieben werden können [Bai08]. Dieser höhere Aufwand wird aufgrund der besseren Qualität der Suchresultate und der damit verbundenen Produktivitätssteigerung von den Unternehmen als gerechtfertigt empfunden [Joh06], [Nie12].

3.2. Produktdatenintegration

Mitte der 80er-Jahre erkannte man, dass ein durchgängiger, rechnerunterstützter Informationsfluss die Grundlage für eine zukunftsweisende Strategie in der Produktentwicklung sein kann [GAWP93]. Dies ist damit zu begründen, dass für unterschiedlichste Problemlösungen in der Produktentwicklung eine Vielzahl spezieller rechnerunterstützter Systeme eingesetzt werden kann. Je nach Aufgabenstellung, Zielsetzung und Leistungsfähigkeit der eingesetzten Systeme werden die rechnerunterstützten Arbeiten und Problemlösungen auf Basis unterschiedlicher Informationsmengen durchgeführt, bzw. werden unterschiedliche Informationsmengen mit den jeweiligen Systemen abgebildet. Die Informationsmengen beziehen sich dabei auf das Produkt selbst sowie auf die Anforderungen und Randbedingungen, denen das Produkt während seines Produktlebenszyklus ausgesetzt ist [AT00]. Die Beschreibung des Produkts mit allen dazugehörigen Informationen, die im Produktlebenszyklus benötigt werden, kann über sogenannte Produktmerkmale erfolgen [KFG07]. Nach DIN 2330 sind Produktmerkmale über Beschaffenheitsmerkmale, Funktionsmerkmale und Relationsmerkmale definiert [DIN93]:

- Beschaffenheitsmerkmale beschreiben Eigenschaften des Produkts wie Gestalt, Material, Gewicht, Farbe, Oberflächengüte etc.

- Funktionsmerkmale beschreiben z. B. den intendierten Zweck, die Funktion, das Verhalten des Produkts
- Relationsmerkmale kennzeichnen Eigenschaften, die erst durch Beziehung z. B. zu Herstellungsschritten, Lagerung, Transport, Verkauf, Nutzung oder Recycling relevant werden.

Die Voraussetzung für die Nutzung (Speicherung, Bearbeitung) dieser Produktmerkmale mithilfe rechnerunterstützter Systeme ist die elektronische Verfügbarkeit der Daten. Der Ansatz der Produktdatenintegration greift diese Idee auf, indem zwischen den verschiedenen rechnerunterstützten Anwendungssystemen entsprechende Kommunikationssysteme auf Basis eines integrierten Produktmodells verfügbar gemacht werden.

Die Erstellung und Bearbeitung der Produktdaten in Anwendungssystemen erfolgt auf Basis einer endlichen Anzahl von Produktdaten, die untereinander in Beziehung stehen und gesamthaft für das jeweilige Anwendungssystem ein Produktdatenmodell oder abgekürzt Produktmodell bilden [GM97]. Ein Produktmodell beschreibt die Menge an Informationen und ihre Beziehungen zueinander, die die Produktmerkmale wiedergeben und die notwendigen Arbeitsoperationen zur rechnerinternen Modellerstellung und -bearbeitung erlauben [AT00], [KTA03]. Da die Produktmodelle der heute eingesetzten Systeme sowohl auf logischer als auch auf physischer Ebene weitgehend auf systemspezifischen Festlegungen basieren, ist eine Kompatibilität zwischen den verschiedenen Systemen prinzipiell nicht gegeben [ES09]. Eine umfassende Integration bzw. ein funktionierender Austausch von Produktmodellen setzt daher Folgendes voraus [GMH95], [AT00], [ADE05]:

- entweder die Verfügbarkeit komplexer integrierter Systeme auf Basis eines gemeinsamen Produktmodells (auf konzeptueller Ebene) und kompatibler Daten (auf physischer Ebene), mit denen die unterschiedlichen Aufgaben in allen Phasen entlang des Produktlebenszyklus bearbeitet werden können (proprietäre Systemintegration),
- oder die Verfügbarkeit eines integrierten und genormten Produktmodells sowie die Verfügbarkeit darauf aufbauender Systeme (Systemintegration über Standards).

3.2.1. Standard-Datenmodelle

Im Bereich Produktdatenintegration haben sich zwei Standard-Datenmodelle etabliert, STEP und die auf CORBA (siehe Abschnitt 3.1.2) aufbauenden PDM Enablers. Beide Standards sind sehr umfangreich und erfordern deshalb einen entsprechend hohen Implementierungsaufwand [ADE05].

STEP

Die Integration einmal erstellter Produktdaten in unterschiedliche rechnerunterstützte Systeme und in alle Phasen des Produktlebenszyklus ist wesentliche Zielsetzung der ISO-Normenreihe 10303 STEP (*Standard for the Exchange of Product Model Data*) [ISO94]. Mit der Entwicklung von STEP wurde Mitte der 80er-Jahre begonnen. Die damalige, weit in die Zukunft reichende Konzeption ist auch heute noch von großer Bedeutung. Die wichtigsten Zielgrößen der STEP-Entwicklung sind [Fow95], [AT00]:

- Spezifikation eines objektorientiert definierten Produktdatenmodells, das alle Produktdaten aus allen Phasen des Produktlebenszyklus enthält.
- Formale Spezifikation des Produktdatenmodells, um diese Spezifikation direkt in Softwaresysteme umsetzen zu können (*Description Methods*).
- Methoden zur Unterstützung einer normkonformen Implementierung (*Implementation Methods*).
- Methoden zur Qualitätssicherung bei der Umsetzung (*Conformance Testing Methodology and Framework*).
- Berücksichtigung von mehreren Funktionen der Produktdatenverarbeitung wie Produktdatenaustausch, Produktdatenspeicherung, Produktdatenarchivierung und Produktdatentransformation.
- Konzept des integrierten Produktmodells als Grundlage, um daraus verschiedene Anwendungen abzuleiten.

STEP wird als Normenreihe mit folgender Unterteilung (siehe Tab. 3.2) in den verschiedenen Gremien des ISO TC184/SC entwickelt. STEP stellt einen wichtigen Beitrag zur Integration von Produktmodellen dar, da es mit STEP erstmals gelang, einen umfassenden internationalen Standard zu definieren, der für die verschiedenen Anwendungsbereiche des Produktlebenszyklus sogenannte Anwendungsprotokolle (*application protocols*) festschreibt [Fow95]. Die Anwendungsprotokolle basieren auf den ebenfalls in STEP definierten integrierten Ressourcen (*integrated resources*). Diese Ressourcen bilden den Kern des STEP-Informationsmodells, der aus den branchenunabhängigen Grundmodellen (*generic resources*) und den Anwendungsmodellen (*application resources*) besteht. Die auf den integrierten Ressourcen und Anwendungsmodellen aufbauenden Anwendungsprotokolle nutzen die darin definierten Konstrukte und erweitern sie um die für die Anwendung notwendige Semantik [AT00].

Serie	Beschreibung
<i>0 Series</i>	Overview and Fundamental Principles
<i>10 Series</i>	Description Methods
<i>20 Series</i>	Implementation Methods
<i>30 Series</i>	Conformance Testing Methodology and Framework
<i>40-99 Series</i>	Integrated Resources: Generic Resources
<i>101-199 Series</i>	Integrated Resources: Application Resources
<i>201-299 Series</i>	Application Protocols
<i>301-399 Series</i>	Abstract Test Suites
<i>501-599 Series</i>	Abstraction Integrated Constructs

Tabelle 3.2.: Die Unterteilung der STEP-Normenreihe

Dies ist eine grundlegende Voraussetzung für die Integration verschiedener rechnerunterstützter Systeme sowohl innerhalb als auch außerhalb eines Unternehmens, insbesondere bei sehr komplexen Produktentwicklungsprozessen.

PDM Enabler

Product Data Management Enabler (PDM Enabler) ist eine Spezifikation zur Implementierung von Daten- und Funktionsschnittstellen für Produkt- und Prozessinformationen [OMG12a]. Die Spezifikation stellt ein Grundgerüst für die Implementierung von Schnittstellendefinitionen auf Basis von CORBA zur Verfügung [ADE05]. Das Konzept geht auf eine Initiative der Object Management Group (OMG) zurück. Im Gegensatz zu den STEP-basierten Offline-Verfahren besteht die mit dem PDM-Enabler-Standard die Möglichkeit eines Onlinezugriffs auf die Produktdaten [ES09]. Dabei handelt es sich, wie in der Abbildung 3.8 dargestellt ist, um ein aus zwölf Modulen bestehendes Konzept. Die Module sind in der Spezifikation durch die *Interface Definition Language* (IDL) beschrieben.

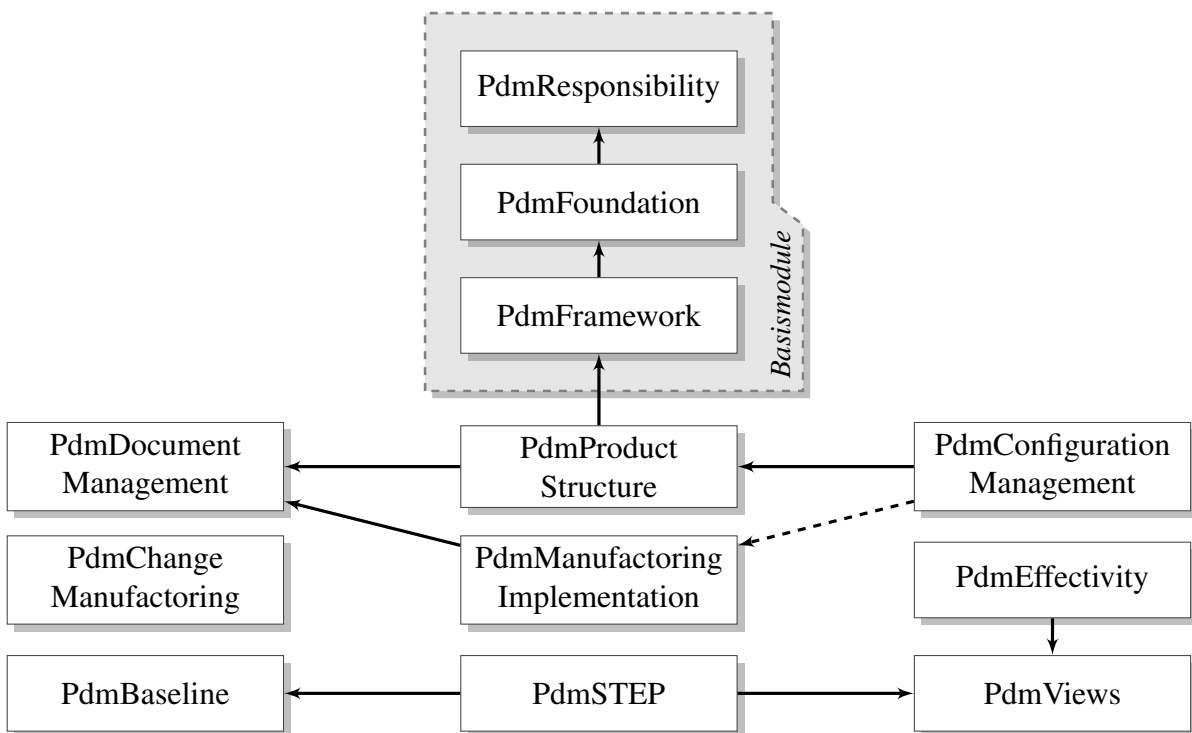


Abbildung 3.8.: Modulkonzept der PDM Enabler nach [KEME99]

Die Aufgabenbereiche für die Grundmodule lassen sich wie folgt skizzieren [ES09]:

Grundmodul	Aufgabenbereich
<i>PDM Responsibility</i>	Objekte für: - Personen, Organisationen, Programme (Actors)
<i>PDM Foundation</i>	Schnittstellen für: - Standard-Verwaltungseigenschaften (Manageable) - Identifizierungsmöglichkeiten (Identifiable) - Sperrkonzept (Lockable): Actors können Objekte sperren - Zustandsinformation (Stateble) - Sicherheitsklassifizierung
<i>PDM Framework</i>	Das Haupt-Basis-Modul für: - Dynamisches Attributieren (Attributable) - Behandeln von Sichten (Qualifiable) - Baseline-fähig (Baselineable) - Änderungen vornehmen (Changeble)

Als funktionale Module stehen folgende zentrale Konzepte zur Verfügung:

- Dokumentmanagement (Metadaten & Originale)
- Produktstruktur (Bauteile)
- Sichtenabbildung (Kontexte)
- Gültigkeiten (zeitlich, Seriennummern bezogen)
- Baseline (Einfrieren eines Standes)
- Änderungswesen (Change Management)
- Produktionsplanung
- STEP (Import/Export)
- Konfiguration (Auswahl mit zugehörigen Bedingungen)

Beim Einsatz der PDM Enabler ergibt sich jedoch eine Reihe von Problemen. So sind aufgrund der Verwendung des CORBA Relationship Service [OMG12b] vor allem bei der Arbeit mit großen Produktstrukturen Performanzprobleme zu erwarten [ES09]. Der Grund hierfür ist eine sehr große Anzahl von Rollen- und Beziehungsobjekten im Verhältnis zu den referenzierten Objekten [ES09]. Darüber hinaus ist es schwierig, eine Standardisierung des Datenaustauschs in einer firmenspezifischen Umgebung durchzusetzen.

3.2.2. PDM-basierte Integration

Der Begriff *Product Data Management* (PDM) steht für ganzheitliche Verwaltung aller Daten, die im gesamten Produktlebenszyklus eines Produkts anfallen, bearbeitet und weitergeleitet werden [ES09]. Die auf ablauforganisatorischen Vorgaben basierende Kontrolle und Steuerung des Informationsflusses für optimale Prozesse mit geringen Fehleranfälligkeiten sind weitere Ziele des PDM [AT00].

Ziel	Beschreibung
<i>Verkürzung von Entwicklungszeiten (Time to market)</i>	<ul style="list-style-type: none"> - schnelle Suche und schnelleren Zugriff auf aktuelle Daten - kurze Kommunikations- und Verteilzeiten - Synchronisation von Informationsständen in parallel laufenden Prozessen (Simultaneous Engineering, Concurrent Design) - Steuerung von Prozessen (<i>Workflow</i>) - Kontrolle von Zwischenergebnissen (z. B. Vollständigkeit digitaler mock-ups) und rechtzeitiger Einleitung geeigneter Maßnahmen
<i>Senkung der Entwicklungskosten</i>	<ul style="list-style-type: none"> - Vermeidung von redundanten Datenerfassungen - Reduzierung der Aufwände für die Informationsbeschaffung - Verbesserter Zugriff auf Wiederhol-, Katalog- und Normteile mit Vermeidung der Anlage neuer Sachnummern - Reduzierung von Fehlerquellen
<i>Verbesserung der Produktfunktionalität und Produktqualität</i>	<ul style="list-style-type: none"> - Verbesserung der Informationsbereitstellung - Reduzierung der Fehlerquellen
<i>Beherrschung der Verwaltung aller Produktvarianten- und konfigurationen</i>	<ul style="list-style-type: none"> - zur Erhaltung von gesetzlichen Vorschriften - Sicherstellung der Ersatzteilebereitstellung
<i>Unterstützung des Qualitätsmanagements nach der Normenreihe ISO 9000</i>	<ul style="list-style-type: none"> - Sicherstellung einer geordneten Daten- bzw. Langzeitarchivierung - nachvollziehbarer Abläufe
<i>Reduzierung der Aufwände und Verkürzung von Durchlaufzeiten</i>	<ul style="list-style-type: none"> - nachgelagerter Prozesse wie die Arbeitsvorbereitung durch schnellere und aktuelle Datenbereitstellung sowie als Beitrag zur Verkürzung des <i>Time to Customer</i> Prozesses

Tabelle 3.3.: PDM – übergeordnete Ziele

PDM-Systeme realisieren die Methoden des Produktdatenmanagements mithilfe von Rechner-systemen. Sie organisieren und realisieren den Zugriff auf die Daten des Produktmodells und bieten Dienste wie die Generierung der Modellschemata, Konsistenzsicherung und Bereitstellung differenzierter Zugriffsmöglichkeiten an [KFG07]. Die mit dem Produktdatenmanagement

verfolgten übergeordneten Ziele in der Produktentwicklung sind in der Tabelle 3.3 aufgelistet. Das wichtigste Ziel eines PDM-Systems ist die Integration aller organisatorischer Produkt- und Dokumentdaten sowie eine durchgängige Unterstützung der technisch orientierten Produktentwicklungsprozesse [KFG07]. Der unternehmensweite Einsatz eines PDM-Systems als Integrationsplattform für alle relevanten IT-Systeme ermöglicht somit den Aufbau eines Informationsnetzwerkes mit nahezu redundanzfreier Datenhaltung, das sämtliche Arbeitsabläufe im Bereich der Produktentwicklung steuert und dokumentiert. PDM-Systeme sind somit für die Verwaltung aller im Rahmen der Produktentwicklung entstehender Daten und Dokumente zuständig und bieten den unternehmensweiten Zugriff und die Verwaltung dieser Informationsmengen [Hor04]. Allgemein lassen sich die typischen Funktionen von PDM-Systemen in drei Gruppen aufteilen [Jen01]:

- Das Produktdaten- und Dokumentenmanagement beschreibt Funktionen, die die verwalteten Objekte betreffen, also eher die statischen Aspekte.
- Das Prozess- und Projektmanagement beschreibt die Funktionen, die die Vorgehensweise bei der Informationsverarbeitung betreffen, also eher dynamische Aspekte.
- Die Administration beschreibt die Funktionen, die den reibungslosen Betrieb des Systems gewährleisten sollen.

PDM-Systeme sind nach dem Client-Server-Prinzip aufgebaut und verfügen über eine grafische Benutzeroberfläche, das sogenannte *Graphical User Interface* (GUI). In der Regel liegt einem PDM-System ein modularer Aufbau einzelner Systemkomponenten zugrunde. Dabei bildet die Basis üblicherweise kommerziell verfügbare relationale Datenbankmanagement-Systeme. Des Weiteren bieten PDM-Systeme eine Reihe von Tools zur Systemanpassung, zur Datenrestrukturierung sowie Schnittstellen zum Datenaustausch mit anderen Systemen [SK05]. Eine PDM-Systemarchitektur gemäß der VDI-Richtlinie 2219 ist in Abbildung 3.9 dargestellt.

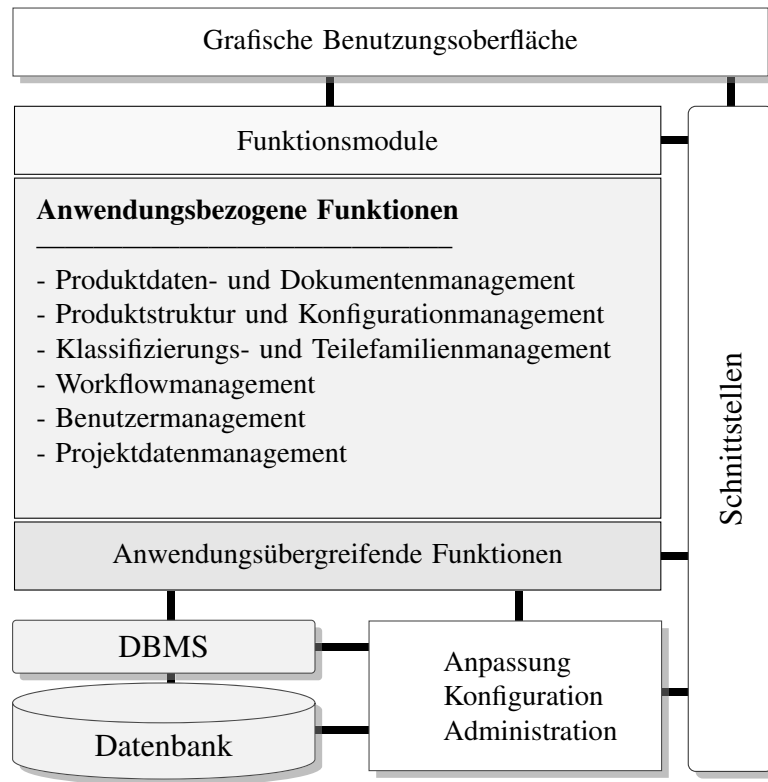


Abbildung 3.9.: PDM-Systemarchitektur [VDI02]

Die technische Stamm- und Stücklistenverwaltung, das Dokumentenmanagement und die Funktionen der Klassifizierung durch Gruppentechnik sind in der Regel als Standardfunktionen in allen marktüblichen PDM-Systemen enthalten [ES09]. Ein vollständige Konfigurations- und Projektmanagement, Publishing, Archivierungs- und Backup-Funktionen sind eher die Ausnahme [ES09].

3.2.3. ERP-basierte Integration

Die Einführung von ERP (*Enterprise Resource Planing*)-Systemen brachte vielen Unternehmen große Fortschritte bei der IT-Systemintegration [Aie06]. Hierbei handelt es sich um eine Standardsoftware, die aus einer Vielzahl von vorintegrierten Modulen besteht.

Das ERP-System ist zwar in der Lage, die wesentlichen Geschäftsprozesse umfassend zu unterstützen, dennoch erfordert die Einbindung zusätzlicher betrieblicher Anwendungssysteme eine unternehmens- bzw. branchenspezifische Anpassung [Kai02]. Beim ERP-basierten Ansatz agiert das ERP-System als eine zentrale Integrationsplattform, an die unterschiedliche Applikationen angebunden sind. In diesem Zusammenhang bezeichnet man das ERP-System auch als Backbone, wobei die Standardschnittstellen, die seitens des ERP-Systems angeboten werden, zur Integration von IT-Systemen eingesetzt werden.

Der ERP-basierte Integrationsansatz ist insbesondere dann vorteilhaft, wenn zwischen den zusätzlich einzubindenden Anwendungen kein direkter Informationsaustausch benötigt wird [Aie06]. In der Praxis treten jedoch in größeren Unternehmen trotz Standardisierung aufgrund der hohen Komplexität Probleme bei der Integration auf, sodass die Integrationsaufgabe keineswegs trivial ist [SM01]. Zudem unterstützen die Standardschnittstellen und Integrationsmechanismen der ERP-Anbieter in der Regel nur eine Integration auf Daten- oder Programmebene, und die ERP-internen Prozesse lassen sich nur schwer auf andere Anwendungen ausweiten [Kai02]. Des Weiteren kann der ERP-basierte Integrationsansatz zu Performanzproblemen bei den Backoffice-Systemen führen, wenn etwa Webanwendungen direkt und somit ohne die entzerrnde Wirkung einer zwischengeschalteten Integrationsinstanz integriert werden [Kai02], [BK10]. Als weitverbreiteter Vertreter dieses Softwaretyps wird *SAP NetWeaver* der Firma SAP AG betrachtet.

SAP NetWeaver

SAP NetWeaver ist ein Konzept für eine Technologieplattform, die als Grundlage für die Entwicklung neuer Lösungen und zur Integration der existierenden Anwendungen dient [HK07]. Hierbei bildet SAP NetWeaver eine servicebasierte Plattform für alle SAP-Lösungen und somit die Basis für die sogenannte *Enterprise Services Architecture* (ESA) [Bas03]. ESA beschreibt, wie eine Integrations- und Anwendungsplattform genutzt werden kann, um Geschäftsprozesse schnell und flexibel abzubilden und aufeinander abzustimmen [Bas03]. Dabei soll das technologische Integrationskonzept in Bezug auf die zu integrierende Anwendungen folgende Kriterien erfüllen [Wol03], [HWG05]:

- Sicherheit, Skalierbarkeit, Zuverlässigkeit und Internationalisierung als Fundament dieser Architektur.
- Offenheit in der Verwendung von Plattformen.
- Unterstützung offener und standardisierter Internettechnologien.
- Bündelung von Informationen, Anwendungen und Prozessen unter einer gemeinsamen Benutzeroberfläche.
- Vollständige Integration zur Realisierung unternehmensinterner und -übergreifender Zusammenarbeit von Geschäftsanwendungen.

SAP NetWeaver nutzt Internetstandards wie HTTP⁷, XML und Webservices [Bas03]. Dadurch wird eine offene Systemarchitektur geschaffen, die kompatibel mit Java aber auch zu .NET ist [HWG05]. Die SAP-NetWeaver-Plattform ist, wie in Abbildung 3.10 dargestellt, in vier fachliche Bereiche eingeteilt [HK07]:

⁷Hypertext Transfer Protocol: ein Protokoll zur Übertragung von Daten in einem Netzwerk

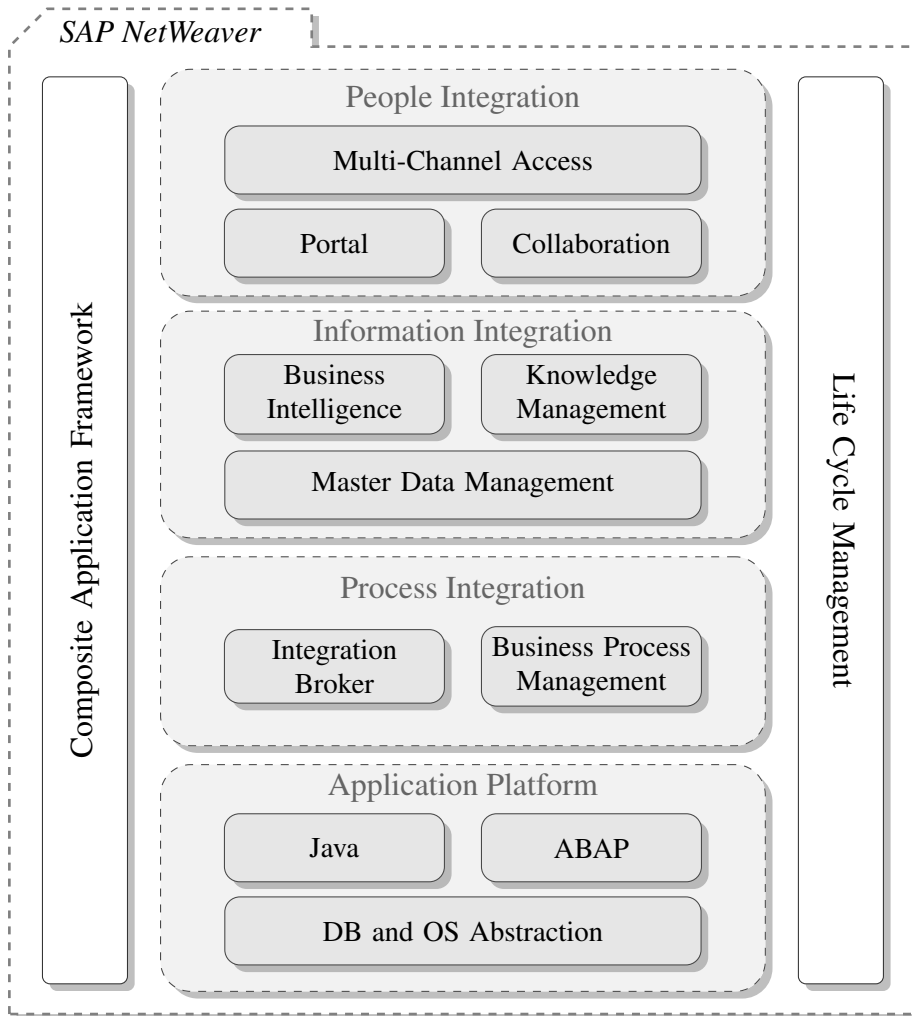


Abbildung 3.10.: SAP NetWeaver Architektur nach [AG12]

- *People Integration* ermöglicht es, personenrelevante Information zur Verfügung zu stellen.
- *Information Integration* fasst die Komponenten zusammen, um Mehrwert durch Informationsintegration zu erzeugen.
- *Process Integration* ermöglicht das Zusammenwirken von heterogenen Komponenten innerhalb eines Geschäftsprozesses.
- *Application Platform* stellt eine gemeinsame Umgebung für ABAP und Java-konforme Komponenten zur Verfügung.

Für alle vier Bereiche gibt es das *Life Cycle Management* und das *Composite Application Framework*. Das Composite Application Framework erlaubt die funktionsübergreifende Abbildung

von Geschäftsprozessen über unterschiedliche Applikations-, Service- und Organisationsebenen hinweg [HWG05]. Life Cycle Management umfasst das Design, die Entwicklung, den Test, den fortlaufenden Betrieb der Applikation und dessen Administrations- bzw. Change-Management [HK07].

SAP NetWeaver bietet eine umfangreiche Technologie an, auf deren Basis ein Unternehmen die unternehmensinterne und unternehmensübergreifende Integration von Informationssystemen realisieren kann [Wol03]. Allerdings existiert heute noch ein Bruch zwischen der Konzeption und der Realisierung von Integrationslösungen [HK07].

3.3. Relevante Forschungsaktivitäten

In Anbetracht der Zahl beteiligter Anwendungen in der Produktentwicklung wurde das Potenzial von Integrationstechnologien in mehreren Forschungsprojekten erkannt und deren Verwendung angedacht. Exemplarisch werden im Folgenden – zum Teil basierend auf den Betrachtungen von EIGNER und STELZER [ES09] – einige dieser Forschungsprojekte illustriert.

3.3.1. iViP

Die übergreifende Zielsetzung des von BMBF geförderten Leitprojekts *Innovative Technologien und Systeme für die integrierte virtuelle Produktentstehung* (iViP) war die Entwicklung und industrielle Einführung innovativer Softwarewerkzeuge für die neuartige Produktentstehung auf Basis virtueller Produkte [KTA03]. Der Ansatz zur Erreichung dieses Ziels fokussierte die durchgehende Digitalisierung des Produktentstehungsprozesses, die als Hauptmerkmal zur Steigerung der Wettbewerbsfähigkeit betrachtet wird.

Technologische Grundlage dieser Zielsetzung ist eine im Rahmen des iViP-Projektes entwickelte CORBA-basierte Plattform, die als Kommunikationsgrundlage für alle am Produktentstehungsprozess beteiligten Softwarewerkzeuge dient [Ehr04]. Dabei stellt die Plattform einen Integrationsbus, an dem sich Services registrieren können sowie einen generischen Client mit sämtlichen Funktionen zur Verfügung (Abbildung 3.11). Zur werkzeugübergreifenden Kommunikation dient der auf CORBA-Basis von der OMG entwickelte Schnittstellenstandard PDM Enablers [OMG12a].

PDM-Systeme, die innerhalb eines iViP-Szenarios als Datenquelle dienen sollen, werden über Wrapper-Komponenten an den Integrationsbus angeschlossen. Die Wrapper bilden PDM-Enablers-Aufrufe, die von iViP-Werkzeugen oder dem Client gestartet werden, auf die proprietäre Schnittstelle des PDM-Systems ab. Auf diese Weise können auch bereits vorhandene Systeme in eine iViP-Softwareumgebung integriert werden.

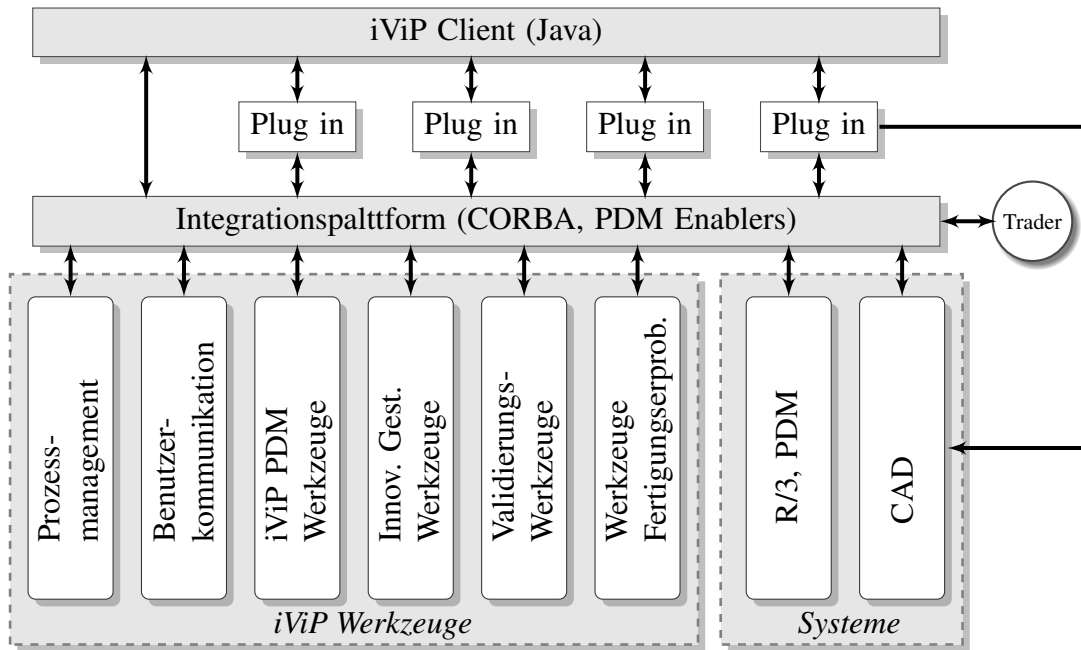


Abbildung 3.11.: iViP-Plattformarchitektur [KTA03]

Praktische Versuche im Rahmen der Konzeptvalidierung haben ergeben, dass ein hoher Aufwand bei der Anbindung der Systeme entsteht. Der Grund dafür lag vor allem darin, dass bei der Abbildung vom proprietären Informationsmodell eines spezifischen PDM-Systems auf die PDM-Enablers-Spezifikation ein aufwendiges Mapping stattfinden muss [Ehr04]. Innerhalb eines Unternehmens kann zwar eine iViP-Plattform bzw. darauf aufbauende Konzepte zur Kopplung unterschiedlicher PDM- und Dokumentenverwaltungssysteme eingesetzt werden, für flexible, unternehmensübergreifende Produktentwicklung ist der iViP-Ansatz jedoch aufgrund des hohen Implementierungsaufwands ungeeignet [Ehr04].

3.3.2. PDTnet

Das Verbundprojekt *Product Data Technology and Communication in an OEM an Supplier Network* (PDTnet) setzt sich zum Ziel, die neutrale systemunabhängige Produktdatenkommunikation zwischen Automobilherstellern und Zulieferern auf der Basis des Standards STEP AP214 und verfügbarer Internettechnologien durch geeignete, im Produkt neu entwickelte, auf XML-basierende Produktdatenstandards und Softwarewerkzeuge zu unterstützen [Pro02].

Dieses Ziel verfolgt das Projekt, indem Data Exchange-Szenarien unterstützt sowie eine Data-Sharing-Schnittstellen realisiert werden (Abb. 3.12). Dabei ist sowohl der Datenaustausch als auch der Onlinezugriff auf Basis eines gemeinsamen XML-Schemas sichergestellt, das vom STEP Standard AP 214 abgeleitet wurde.

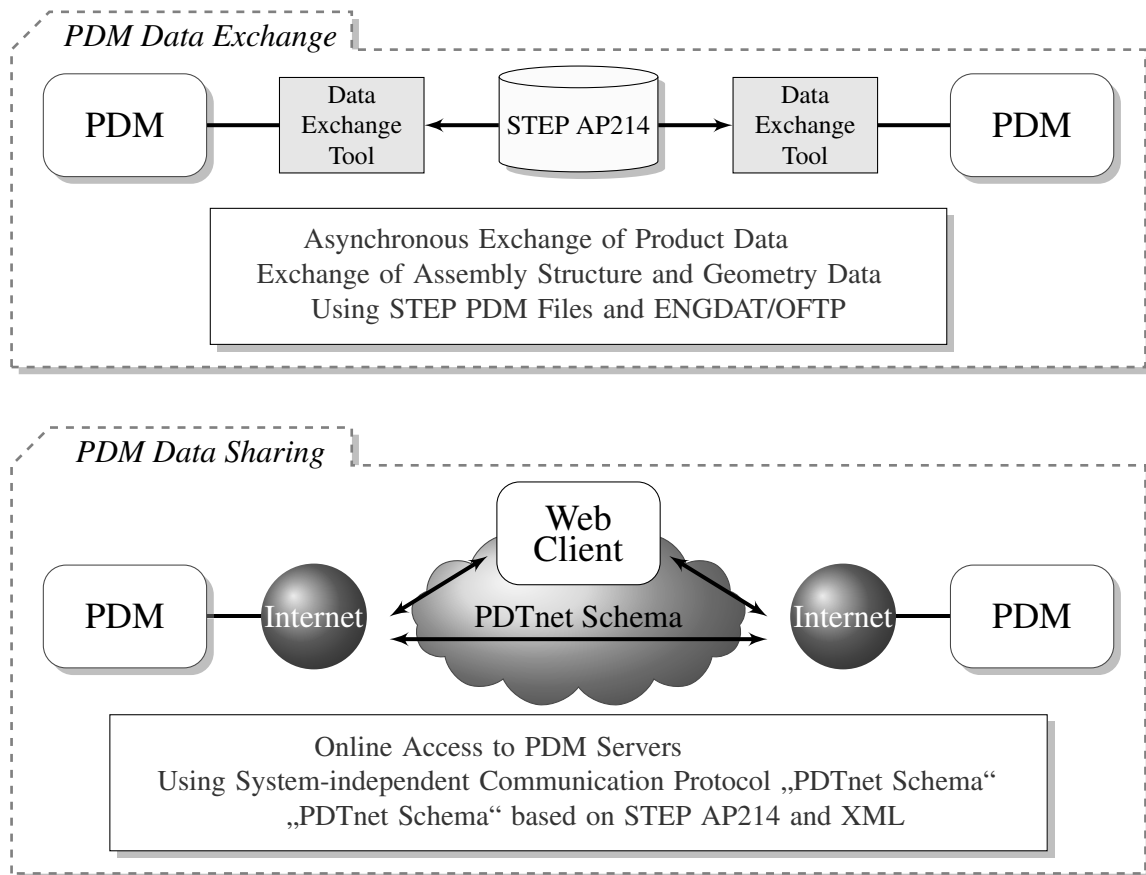


Abbildung 3.12.: PDTnet - Umsetzung von Data Exchange und Data Sharing [Pro02]

Für Data Exchange-Szenarien wird dieses Schema mit konkreten Produktdaten des Quellsystems instanziiert, anschließend mithilfe herkömmlicher Netzwerkprotokollen (HTTP, FTP etc.) transportiert und in das Zielsystem importiert. Der Einsatz der XML-Technologie unterscheidet diesen Prozess von den konventionellen STEP-Prozessoren, indem ein XML-basierendes Repräsentationsmodell verwendet wird.

Die Umsetzung eines Data-Sharing-Szenarios erfolgt dadurch, dass eine PDTnet-Clientkomponente eine in eine SOAP-Message verpackte PDTnet-Anfrage an eine PDTnet-Serverkomponente versendet. Die Serverkomponente ist in der Regel ein Webserver, der die PDTnet-Anfrage auf die proprietäre Schnittstelle eines PDM-Systems übersetzt [iVi12]. Die Serverkomponente verpackt die angefragten Produktdaten in eine XML-Struktur gemäß dem PDTnet-Schema und überträgt diese durch eine SOAP-Response an den Client zurück [Ehr04].

Die Ergebnisse des PDTnet-Projekts bilden eine technische Grundlage für die Integration von Zulieferern in die Entwicklungsprozesse von OEM. Dies hat zur Folge, dass die Anwendern

(Automobilhersteller und Systemlieferanten) sowohl finanziell als auch durch das Bereitstellen von Ergebnissen in Teilprojekten unterstützt werden [Pro02]. Jedoch ist der Ansatz aufgrund des komplexen Informationsmodellansatzes für flexible Produktentwicklungsprozesse ungeeignet [Ehr04].

3.3.3. SEVENPRO

Das von der EU geförderte Forschungsprojekt *SEVENPRO*⁸ hat zum Ziel den Produktentwicklungsprozess mithilfe von semantischen Technologien zu unterstützen [Com12]. Das Projekt trägt somit zu Verbesserung von Prozessen in der Produktentwicklung und in der Fertigung bei, indem eine semantisch gestützte Erfassung, Formalisierung und Verwaltung von Produktwissen erfolgt. Dabei wird die Prozessverbesserung anhand von Kriterien gemessen wie die Zeit, die für eine Produktinnovation erforderlich ist, dem Grad der Wiederverwendung von bereits vorhandenem Wissen in einem Unternehmen und Kostenreduzierung für die Entwicklung [Com12]. Das in *SEVENPRO* entwickelte System soll folgende Problemstellungen korrespondieren [Sch07]:

- heterogene Informationsquellen (z.B. CAD-Daten)
- Austausch von Informationen zwischen den Projektbeteiligten
- hohe Komplexität und hoher Individualisierungsgrad der Produkte
- zeitintensive Produktzyklen
- Generierung und Übertragung von neuem Wissen zu den beteiligten Bereichen

Die Basis von *SEVENPRO* stellt der semantische Wissensspeicher (engl. *repository*) dar, dessen Struktur in sogenannten *Engineering Ontologien* definiert ist. Diese Ontologien beschreiben und ermöglichen die Integration von verschiedenen Arten von Informationsressourcen, die Anwender während des Produkt-Design-Prozesses nutzen, beispielsweise Dokumente wie Patente, Normen, Berichte etc. [Sch06]. Die Daten werden hauptsächlich durch Annotation-Module in das semantische Repository eingetragen. Diese Module erlauben eine semiautomatische semantische Annotation von Informationsquellen. Die Annotationen sind Metadaten, die in einer hohen Anzahl an Dokumenten enthalten sind. Das semantische Repository erfasst Annotationen und andere semantisch repräsentierte Entwicklungsdaten, sodass diese mithilfe von semantischen Servern zu den höher angesiedelten Modulen weitergeleitet werden [Sch07]:

- *Semantic Engineering Module*: Ermöglicht die Erstellung von neuen Produkten, die Verwaltung von Produktwissen unter Berücksichtigung der Zeitachse (Wissen, Versionie-

⁸IST-027473

rung, Revisionen und Lebenszyklus) und die nahtlose Suche und den Abruf von Produktinformationen aus allen Arten von Wissensquellen.

- *Engineering Memory Module*: Ermöglicht die effiziente Wiederverwendung von Engineering-Wissen aus vergangenen Fällen.
- *Semantic Virtual Reality Module*: Ermöglicht den Zugriff auf alle Arten von Produktwissen zusätzlich zu den eigenen *Features*, wodurch eine virtuelle und interaktive Umgebung für den Zugriff und Austausch von Produktwissen unterstützt wird.
- *Virtual Reality Reasoning Module*: Unterstützt regelbasiertes Schließen bei Semantic Virtual Reality Module.
- *Relational Data Mining Module*: Durchsucht das gesamte semantische Produkt-Repository nach abstrakten Mustern, die das Wissen repräsentieren. Diese wertvolle „Informationen über Informationen“ wird dann im semantischen Repository gespeichert, sodass es für andere Module verfügbar ist.

Die entwickelte Architektur bietet die Interaktion zwischen den Modulen sowie den semantischen Serveragenten als ein zentrales Element des Frameworks an (siehe Abb. 3.13).

Gleichzeitig ist auf diese Weise ein Zugang zur semantischen Repository gegeben, in dem alle semantischen Produktdaten verwaltet werden. Alle anderen Module, die auf einer höheren Ebene angesiedelt sind, z. B. Endbenutzerwerkzeuge und das *Data-Mining*-Modul, nutzen den semantischen Serveragenten, um auf die Daten zuzugreifen.

Im Kontext der SEVENPRO-Projektarbeit gibt es zwei Jahresberichte 2006 und 2007 sowie mehrere Konferenzveröffentlichungen und die dazugehörige Internetpräsenz [Sch06], [Sch07]. Gegenwärtig sind jedoch keine Ergebnisse des Projektes in Form von Softwareprototyp, Frameworkgestaltung und konkreten Funktionalitäten veröffentlicht.

3.3.4. The Semaril

Die Vision des Forschungsprojekts *The Semaril* am Lehrstuhl für Konstruktionstechnik/CAD an der Universität des Saarlandes ist die Entwicklung einer Software zur Verwaltung von Produktdaten sowie Arbeitsabläufe auf Basis einer semantischen Netzstruktur [WWPS05]. Hierbei soll es möglich sein, den Überblick auch bei umfangreichen Projekten im Bereich der Produkt- und Softwareentwicklung zu gewährleisten, wobei die Projektdaten nur von einem Anwender und nicht von ganzen Projektteams verwaltet werden [CDK⁺07]. Eine Besonderheit des Semarils ist, dass es strukturierte und unstrukturierte Informationen miteinander verknüpft: Das Rückgrat wird durch ein Netz von Begriffen gebildet, während das „Fleisch“ kurze, schnell niedergeschriebene Notizen bilden, die mit den jeweiligen Begriffen verknüpft sind [WWPS05].

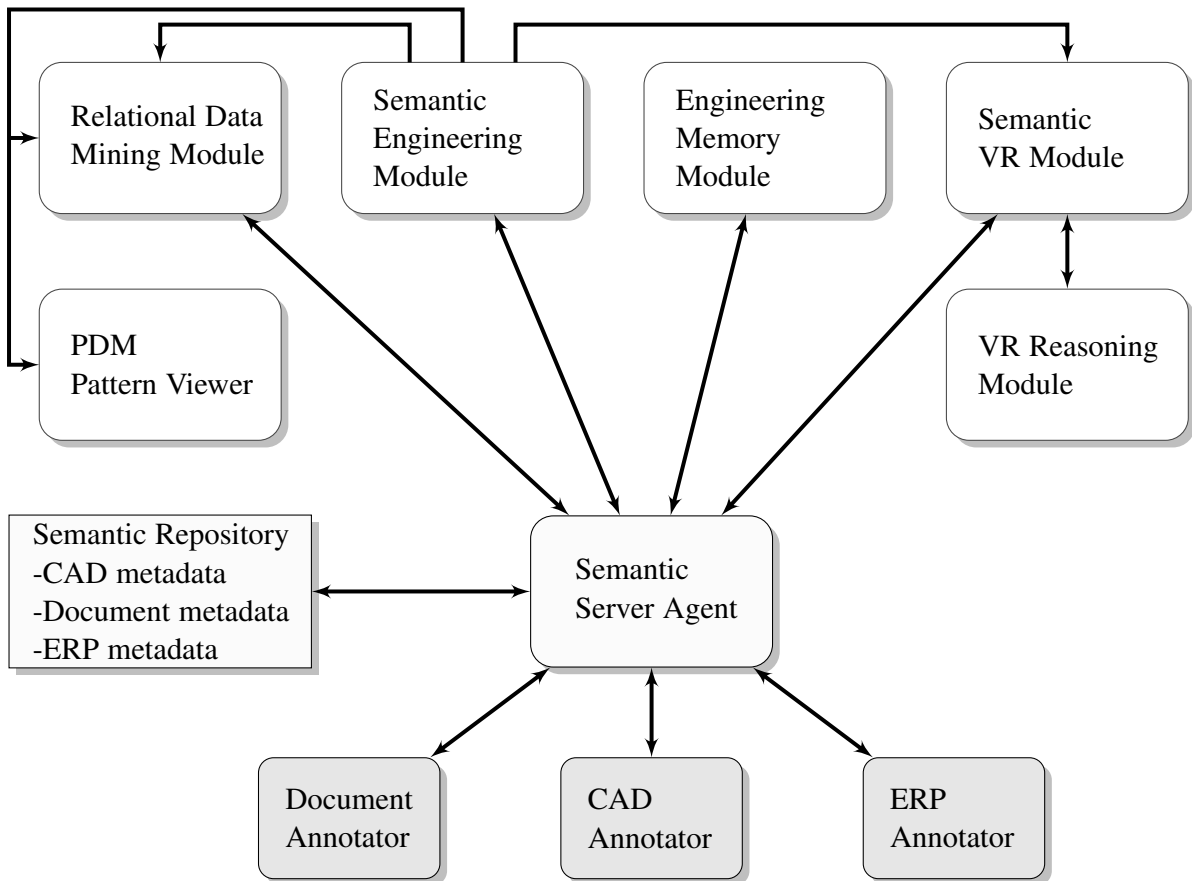


Abbildung 3.13.: Semantic Virtual Engineering Environment [Sch07]

WEBER ET AL. beschreibt die wesentlichen Funktionen sowie Anwendungsgebiete von Semaril wie folgt [WWPS05]:

- ein mächtiger Notizblock
- ein HTML-Editor
- ein Werkzeug zum Management von Dokumenten
- ein Werkzeug zum Dokumentieren von Projekten und Erstellen von Berichten
- ein Wissensmanagementtool
- ein Werkzeug, um verschiedene Excel-Tabellen zu synchronisieren
- ein Werkzeug, um semantische Netze zu bearbeiten und innerhalb dieser zu navigieren
- eine universelle Datenbank

Die Modellierungsfunktion von Semaril bietet die Möglichkeit einer prozessbegleitenden semantischen Erweiterung der Produktstruktur und der dazugehörigen Dokumente. Basierend auf

den bereits vorhandenen Beziehungen und Begriffen kann das System ebenfalls damit vernetzte Begriffe und Dokumente anzeigen [CDK⁺07].

Das semantische Netz kann zwar zur Laufzeit eines Prozesses modelliert werden, was wiederum die praktische Relevanz des Ansatzes erhöht, jedoch ist die Vernetzung von Begriffen statisch modelliert [PSWW05]. Eine weitere Schwachstelle von Semaril stellt die Benutzerschnittstelle dar. Da diese überwiegend mittels „Drag and Drop“-Technik zu bedienen ist, liefert sie keine bzw. nur marginale Information über die Änderungen seitens des Benutzers. Darüber hinaus unterstützt Semaril keine Vererbungsstruktur innerhalb des semantischen Netzes. Dies hat zur Folge, dass ein wesentliches Element der Semantik im System nicht realisiert ist.

Die Veröffentlichungen in Zusammenhang mit Semaril-Projekt zeigen interessante Einsatzmöglichkeiten von semantischen Netzen in der Produktentwicklung. Allerdings liegen diesem Ansatz keine semantisch formale Sprache und kein mathematisches Modell zugrunde.

3.4. Zusammenfassende Bewertung bestehender Ansätze

Die Untersuchung der beschriebenen Ansätze für Informationsintegration sowie für Informationsbereitstellung hat gezeigt, dass die Integration von Informationssystemen immer wieder Schwachstellen aufweist, die durch die einzelnen Verfahren nicht oder nur teilweise abgedeckt werden. Da die Verfügbarkeit konsistenter Informationen in interdisziplinärer und unternehmensübergreifender Produktentwicklung immer mehr an Bedeutung gewinnt, werden diese Problemstellungen noch zusätzlich verstärkt. Eine mangelnde Unterstützung liegt vor allem in den folgenden Punkten vor:

Mangelnde Berücksichtigung unternehmensinterner Anforderungen

Die aktive Mitwirkung von führenden Industrieunternehmen im Bereich der Entwicklung von Standard-Integrationssystemen wie STEP und PDM-Enabler hebt deutlich eine hohe praktische Relevanz solcher Informationsintegrationssysteme für die Produktentwicklung hervor. Aufgrund der unternehmensinternen Anforderungen und daraus resultierender Rahmenbedingungen für die Produktentwicklung sind existierende Integrationssysteme in der Regel nicht auf das Produktspektrum eines einzelnen Unternehmens harmonisiert. Dies hat zu Folge, dass die Handhabbarkeit und Instandhaltung der Informationsintegration sehr aufwendig ist, insbesondere wenn die PDM- oder ERP-basierte Integration – wie heute häufig – manuell erfolgt. Da die Integration in derzeit kommerziell verfügbare Systeme mit einem sehr hohen manuellen Anpassungsgrad verbunden ist, scheuen viele Produktionsunternehmen vor dem in diesem Kontext stehenden Zeit- und Kostenaufwand zurück.

Mangelnde Interoperabilität produktbeschreibender Datenbestände auf semantischer Ebene

Es existiert kein durchgängiger Ansatz, der die Informationsintegration auf eine formale, semantische und neutrale Basis in der Produktentwicklung sowohl konzeptionell als auch softwaretechnisch unterstützt. SOA-Ansätze bieten zwar unterschiedliche Konzepte mit individuellen Schwerpunkten für die Erfassung von heterogenen Datenbeständen, eine adäquate Softwareunterstützung zur Überbrückung schematischer und vor allem semantischer Heterogenität für den Praxiseinsatz fehlt jedoch völlig. Die Standard-Datenmodelle im Bereich Produktdatenintegration schaffen ebenfalls keine Abhilfe, da sie zum einen monodisziplinär und zum anderen auf Basis von STEP-Datenmodell sehr komplex sind. Eine Erweiterung dieser würde zur Folge haben, dass ohnehin die sehr hohe Komplexität nicht mehr handhabbar sein würde.

Kein Aufbau einer einheitlichen Begrifflichkeit

Die bereichsübergreifende Verständigung nimmt gerade in der Produktentwicklung einen hohen Stellenwert ein, weil immer mehr Interessensvertreter sich an den Produktentwicklungsprozessen beteiligen. Hierfür muss das Vorhandensein einer einheitlichen Begrifflichkeit sichergestellt werden, um auf der einen Seite die Vertraulichkeit in der Kommunikation und auf der anderen Seite ein gemeinsames Verständnis für Sachverhalte auf der semantischen Ebene zu gewährleisten. Ein eindeutig anerkanntes integriertes Vokabular für eine semantische Informationsintegration ist unumgänglich, um einerseits neue Begriffe kontinuierlich zu erfassen und andererseits die Aktualisierung bereits vorhandener Ausdrücke durchzuführen. Darüber hinaus liegt den bestehenden Ansätzen kein formales Wissensrepräsentationssystem zugrunde, das beschreibt, in welcher Form die Begriffe in Beziehung zueinander gesetzt werden können. Ein solches System stellt die Ausdrucksstärke und Komplexität in Bezug auf die algorithmische Berechenbarkeit der Sprache sicher.

Mangelnde Unterstützung in Bezug auf Inferenz von implizitem Wissen

Die dargestellten Forschungsansätze im Bereich wissensbasierter Systeme liefern für verschiedene Aspekte wie Wissensrepräsentation oder Wissensverarbeitung wichtige Beiträge für die Informationsintegration in der Produktentwicklung. Keiner dieser Ansätze befasst sich allerdings mit einer Wissensbasis, die über eine einheitliche TBox und verteilte ABox verfügt. Damit die Inferenzverfahren vollständig und korrekt sind, wird in den betrachteten Ansätzen angenommen, dass jedes Individuum in einer zentralen Ontologie vorliegt. Die Ausdrucksstärke der eingesetzten Sprache ist meistens sehr beschränkt, sodass dadurch nur eingeschränkte Inferenzmöglichkeiten erlaubt sind. Alle diese Ansätze beruhen außerdem auf dem Einsatz von

bestehenden Inferenz Engines wie Pellet⁹, FaCT++¹⁰ oder KAON2¹¹, die für das Reasoning über eine vollständige Wissensbasis im Sinne einer vorhandenen TBox und Abox spezifiziert sind. Die Anwendbarkeit der vorgestellten Ansätze auf die Rahmenbedingungen semantischer Informationsintegration in der interdisziplinären Produktentwicklung bleibt offen.

Mangelhafte Unterstützung des Problems der Bereitstellung von Lösungswegen

Eine integrierte Behandlung der Aufgabenstellung sowie der resultierenden Lösungsvorschläge ist erforderlich, um gezielt eine Lösung zu erreichen. So behandeln viele Ansätze zumeist nur die endgültige Ergebnisdarstellung – die Betrachtung, wie eine Problemstellung aufgenommen und zur Lösungserarbeitung überführt wird, wird häufig jedoch weggelassen. Dies hat zur Folge, dass a priori nur nach grundsätzlich bekannten Lösungen gesucht werden kann und diese mithilfe eigener Abstraktionsfähigkeit durch jeweils weitere Begriffe erweitert werden müssen. Das dafür erforderliche Wissen kann als Struktur bestehend aus Knoten und Kanten oder in Form einer an die natürliche Sprache angelehnte Darstellung repräsentiert werden. Im Gegensatz zu heute in der Produktentwicklung verbreiteter, hierarchisch strukturierter Beziehungsformen muss das semantische Integrationssystem in der Lage sein, dem Anwender sämtliche Beziehungen sichtbar zu machen.

All diesen Ansätzen ist gemein, dass sie keine oder nur zum Teil eine Unterstützung für ein Unternehmen bezüglich semantischer Integration von heterogenen Datenbeständen bieten. Die derzeit verfügbaren Verfahren und Methoden berücksichtigen nur einen Teil der komplexen Problemstellung, indem eine Lösung zur Überbrückung technischer und struktureller Heterogenität erarbeitet und operativ behandelt wird. Darüber hinaus ist es nicht möglich, eine Suchanfrage als Problemstellung mithilfe einer deklarativen Sprache zu erfassen und auf Basis einer gemeinsamen Begrifflichkeit zu erstellen. Ein formaler Ansatz zur Formulierung von semantischen Suchanfragen existiert nicht, d. h. der Anwender muss die Antwort bereits vor der eigentlichen Problemsuche formuliert haben, sodass er seine Anfrage lediglich anhand eines reduzierten Ausdrucks erfasst, um nach einer vorhandenen Antwort zu suchen.

Aufbauend auf den vorgestellten Schwachstellen sollen zunächst die Anforderungen an ein System zur semantischen Informationsintegration in Produktentwicklung abgeleitet werden.

⁹<http://clarkparsia.com/pellet/>

¹⁰<http://www.cs.man.ac.uk/~horrocks/FaCT/>

¹¹<http://kaon2.semanticweb.org/>

*Hat die Gesellschaft ein technisches
Bedürfnis, so hilft das der Wissenschaft
mehr voran als zehn Universitäten.*

FRIEDRICH ENGELS (1820–1895)

4. Anforderungen an ein System zur semantischen Informationsintegration

Ein semantisches Informationsintegrationssystem soll in der Lage sein, die erforderlichen Begriffe zur Beschreibung einer Anwendungsdomäne möglichst vollständig abzubilden sowie dem Anwender oder der Anwendung das Wissen in adäquater Form zur Verfügung zu stellen. Der Hauptgrund hierfür ist, dass immer mehr Experten aus verschiedenen Fachbereichen in den Produktentwicklungsprozess involviert sind. Dabei wird durch den Begriffswortschatz eines solchen Integrationssystems das tatsächliche Wissen ausgedrückt. Dieses muss in Form einer formalisierten konzeptionellen Struktur in einer Wissensbasis erfasst werden, um aus vorhandenen Beziehungen Informationen implizit abzuleiten, die es ermöglichen, eine Suchanfrage problemorientiert zu beschreiben und die Lösungsfindung mithilfe logischer Inferenzmechanismen zu erweitern. Damit sind einerseits ungenaue und vage Suchanfragen erlaubt, andererseits besteht auch die Möglichkeit, einige Hinweise zur Spezifizierung der Aufgabenstellungen zu erhalten oder sogar bereits vorhandene Lösungen aus einer Wissensbasis in Verbindung mit zu integrierenden Datenquellen zu gewinnen.

In den folgenden Abschnitten werden die Anforderungen an das System beschrieben, mit deren Hilfe semantische Informationsintegration in der Produktentwicklung durchgeführt werden kann. Diese Anforderungen bilden die Grundlage für das anschließend zu entwickelnde Lösungskonzept.

4.1. Allgemeine Anforderungen an ein System zur ontologiebasierten Informationsintegration

Die grundlegende Aufgabe eines Systems zur ontologiebasierten Informationsintegration ist die Schaffung eines einheitlichen Zugriffs auf zahlreiche heterogene Datenbestände. Diese voneinander unabhängigen Datenbestände liegen in der derzeitigen industriellen Praxis in der Regel verteilt in einer IT-Umgebung vor [GLW02]. Dabei werden die produktbeschreibenden Daten in unterschiedlichen Informationssystemen, z. B. in den EDM/PDM-Systemen, relationalen Datenbanken oder ERP-Systemen, akkumuliert und verwaltet. Aus diesem Grund ist ein geeignetes Informationsintegrationskonzept zu entwickeln, das einen einheitlichen, automatisierten Zugriff auf Produktdaten in einer heterogenen Systemlandschaft gewährleistet. Hierbei liegt das Hauptaugenmerk des Konzepts u. a. auf der Skalierbarkeit des Integrationssystems, damit eine schnelle und einfache Anbindung weiterer Softwaresysteme ermöglicht werden kann.

Ein in der industriellen Praxis verbreitetes Mittel zur Beherrschung rasant zunehmender Informationsmengen ist die Klassifikation. Jedoch erweist sich die Klassifikation in der heutigen Zeit als unzureichend. Das liegt darin begründet, dass durch Klassifikation die relevanten Informationen in der stetig ansteigenden Datenflut mit vertretbarem Aufwand nicht auffindbar sind. Darüber hinaus steigt der Aufwand unproportional zur Pflege der Klassifikation mit der Anzahl der zu integrierenden Datenquellen.

Ein weiterer Grund ist, dass die Klassifikationssysteme naturgemäß unternehmensspezifische Änderungen oder Anpassungen nicht optimal unterstützen [GLW02]. Eine Anpassung des Klassifikationssystems, beispielsweise an eine geänderte oder erweiterte Produktpalette, ist so gut wie ausgeschlossen, da dies bedeuten würde, den gesamten, bereits klassifizierten Datenbestand manuell nachklassifizieren zu müssen [GLW02]. Daher soll das System eine semantische Informationsintegration einerseits gemäß beliebiger, innerbetrieblicher Datenmanagementsysteme ermöglichen, andererseits die Änderungen innerhalb dieser Systeme unterstützen.

Somit ergeben sich für den Einsatz eines ontologiebasierten Informationsintegrationssystems die folgenden allgemeinen Anforderungen:

- *Unabhängigkeit der Anwendbarkeit:* Das Integrationssystem, insbesondere die semantische Informationsbereitstellung, sollte nicht nur für einen bestimmten Diskursbereich anwendbar sein. Ferner soll durch den Einsatz der formalen konzeptionellen Strukturen in Form einer Ontologie die Wissensrepräsentationsebene von der Wissensverarbeitungsebene entkoppelt werden, sodass die Anwendbarkeit des Systems auf unterschiedliche Anwendungsbereiche ermöglicht wird.
- *Modularität und Erweiterbarkeit:* Eine modulare Struktur des Integrationssystems soll den Einsatz einzelner Softwaremodule sowie die Skalierbarkeit des Gesamtsystems ermöglichen. Um heterogene Datenquellen auf semantischer Ebene in Beziehung setzen zu können, sind ein anwendungsneutrales und redundanzfreies Informationsmodell sowie die Erweiterbarkeit des Detaillierungsgrads der Beziehungen anzustreben.

4.2. Anforderungen an die produktbeschreibenden Datenbestände

Eine Vielzahl unterschiedlicher Softwaresysteme, beginnend mit MS-Office-Anwendungen wie Excel, über Datenbanksysteme wie MySQL bis hin zu PDM/PLM-Systemen wie Windchill bildet das informationstechnische Rückgrat heutiger Unternehmen.

Informationen zur Beschreibung eines Produkts sind in der Regel verstreut in all diesen Systemen verfügbar. Folglich muss ein System zur semantischen Informationsintegration alle produktbeschreibenden Datenbestände integrieren, um vorhandene Informationen zu einem solchen Produkt korrekt und vollständig bereitstellen zu können.

Ein möglicher Ansatz, den Zugriff auf all diese Produktinformationen zu gewährleisten, wäre

die Erfassung aller produktbeschreibenden Daten in einem einzigen, integrierten Produktmodell. Das integrierte Produktmodell umfasst dabei widerspruchs- und redundanzfreie Produktdaten zu einem beliebigen Zeitpunkt des Produktentwicklungsprozesses. Eine Übersicht über Historie und Methodik erarbeiteter integrierter Produktmodelle bietet der Forschungsbericht der Deutschen Forschungsgemeinschaft (DFG) [Nag03]. Die bedeutendsten Forschungsaktivitäten zur Entwicklung integrierter Produktmodelle trugen zur Entstehung der internationalen Norm ISO 10303 bei, die unter dem Namen STEP bekannt ist [AT00].

In der Praxis konnten sich integrierte Produktmodelle bislang noch nicht durchsetzen [ES09]. STEP-basierende Schnittstellen finden jedoch bei einem Datenaustausch zwischen verschiedenen CAD-Systemen Verwendung. Im Hinblick auf eine praxisorientierte Umsetzung soll das semantische Integrationssystem in der Lage sein, durch eine entsprechende Systemarchitektur und (Teil-)Funktionen produktbeschreibende Informationen aus einer verteilten, heterogenen IT-Landschaft zu integrieren und bereitzustellen.

Aus diesem Grund müssen einige Anforderungen in Bezug auf Struktur sowie auf den Inhalt heterogener Datenbestände erfüllt sein, um semantische Informationsintegrationen durchführen zu können, wenngleich kein integriertes Produktmodell von den zu integrierenden Datenbeständen vorhanden ist. Zum einen ist es nicht möglich, integrationsrelevante Produktdaten „aus dem Nichts“ zu beschaffen, das bedeutet, die produktbeschreibenden Daten müssen zumindest implizit in einer der Datenbestände vorkommen. Grundsätzlich, um eine Informationsintegration und darauf basierend zufriedenstellende Suchergebnisse zu erzielen, sollte die zu integrierende Datenquelle über eine möglichst hohe Datenqualität verfügen.

Diverse Softwareanwendungen setzen keine einheitliche Datenstruktur voraus, die der Benutzer einhalten muss. So wird z.B. MS-Excel in zahlreichen Unternehmensbereichen zur Erfassung der produktrelevanten Daten verwendet, beispielsweise für Konstruktionsstücklisten, Konstruktionsmitteilungen, Änderungsanträgen etc. Hierbei steht dem Anwender die Wahl der Struktur frei, beginnend mit Spaltenanzahl über Spaltenreihenfolge bis hin zu Spaltenbezeichnung. Folglich besteht eine weitere Anforderung in der Aufstellung einer – wenigstens semiformalen – lokalen Struktur der zu integrierenden Datenbestände, um die Informationsbereitstellung aus solchen Anwendungen zu ermöglichen.

4.3. Anforderungen an die Wissensrepräsentation

Um einen Diskursbereich formal beschreiben zu können, muss ein solcher Diskursbereich durch ein wohldefiniertes Vokabular darstellbar sein. Ein anerkannter und dem Benutzer vertrauter Begriffswortschatz bildet hierfür die Grundlage. Dabei soll dieser allgemeine, geschäftsbereichs- und unternehmensspezifische Begriffe enthalten, um einerseits ontologische Konzepte abzubilden und andererseits Verwaltung der konzeptionellen Struktur zu ermöglichen.

Konzeptionelle Struktur soll unter Verwendung der DIN-Normen 2330 und 2331 [DIN79], [DIN80] aufgestellt werden. Bei dem Aufbau konzeptioneller Strukturen in Form einer Ontologie sollen sprachliche Begriffe und Axiome durch die Betrachtung genau einer Domäne definiert werden, um subjektive, sprachliche Interpretationen zu verhindern. Hierbei soll die Ontologie durch diskursbereichsspezifische Eigenschaften sowie synonymbehaftete Definitionen unter der Berücksichtigung von Ein- und Mehrwortbenennungen angereichert werden. In diesem Zusammenhang ergeben sich für die Wissensrepräsentation einer Anwendungsdomäne folgende Anforderungen:

- Genauigkeit der Sprachkonstrukte
- Knappheit der Sprachkonstrukte
- anwenderorientierte Sprachkonstrukte

Die Genauigkeit der Sprachkonstrukte stellt eine möglichst eindeutige Beziehung sicher zwischen einem Konstrukt konzeptioneller Struktur und dessen beschreibungslogischer Bezeichnung. Diese Beziehung wird dabei durch die Knappheit der Sprachkonstrukte auf möglichst geringfügige Ausdrücke natürlicher Sprache reduziert. Ferner sollen Sprachkonstrukte an das dem Anwender vertraute Vokabular angelehnt sein und sogenannte lexikografische Hilfsmittel wie Nachschlagewerke, Normensammlungen etc. verwenden.

Durch das konzeptionelle Modell soll der Sachverhalt einer Anwendungsdomäne wiedergegeben werden. Dies erfordert eine Wissensrepräsentation, die redundanz- und widerspruchsfrei ist. Anderenfalls wird es zu sogenannten Schemaredundanzen auf der konzeptionellen Ebene führen und bei der Umsetzung sich zwangsläufig in Form sogenannter Instanzredundanzen auswirken. Sollen aus systemtechnischen Überlegungen Redundanzen bewusst in das zu implementierende Integrationssystem aufgenommen werden, dann soll dies in der Implementierungs- und Ausführungsphase mithilfe eines redundanzfreien konzeptionellen Modells aufgefangen werden.

Die Semantik produktbeschreibender Daten und der dazugehörigen Kontextinformationen soll am Rechner verarbeitbar, d. h. formalisiert sein, um diese automatisiert auswerten zu können. Die Formalisierung konzeptioneller Struktur soll mithilfe geeigneter ausdrucksmächtiger Ausdrucksmittel der Repräsentationssprache erfolgen. Die Wissensrepräsentation soll darüber hinaus ermöglichen, das konzeptionelle Modell um weitere Sach- und Situationszusammenhänge (z. B. Aufgabenbeschreibungen, Produkt- und Dienstleistungsbeschreibungen) mit vertretbarem Aufwand zu erweitern.

4.4. Anforderungen an die Inferenzmechanismen

Das System muss in der Lage sein, alle explizit in den heterogenen Produktdatenbeständen verfügbaren Produktinformationen zu ermitteln. Dabei liegen viele produktrelevante Informationen in den Datenbeständen nicht explizit vor, sondern können nur implizit mittels Inferenzmechanismen aus den verfügbaren Konzepten bzw. Axiomen abgeleitet werden. Ein Beispiel hierfür ist die Herleitung eines Zuliefererteils, wenn lediglich eine Beziehung explizit vorhanden ist, nämlich dass ein produzierendes Unternehmen einem Zuliefererverbund angehört. Das System soll in der Lage sein, solche impliziten Zusammenhänge zu erkennen und aus unterschiedlichen Quellen zu integrieren. Hinzu kommt, dass entsprechende Kontextinformationen in der Regel von unterschiedlichen Organisationseinheiten und mittels unterschiedlicher Softwaretools an verschiedenen Orten erzeugt werden. Um das Wissen, das in der Wissensbasis formalisiert ist, zur Integration von produktbeschreibenden Daten verwenden zu können, müssen beschreibungslogische Schlussfolgerungen unterstützt werden. Dementsprechend müssen nicht nur grundlegende Inferenzmechanismen zur Verfügung gestellt werden, sondern darüber hinaus die gelieferten Antworten im logischen Sinne korrekt und vollständig sein. Des Weiteren soll bei der ontologiebasierten Informationsintegration in der Produktentwicklung die semantische Suche in Form von konjunktiven Anfragen gewährleistet sein. Damit wird dem Anwender das Werkzeug zur Verfügung gestellt, mit dessen Hilfe komplexe Sachverhalte in Bezug auf Produktinformationen ermittelt werden können.

Produktbeschreibende Daten zeichnen sich durch eine lange Lebensdauer aus, sei es aus Gründen der Wiederverwendbarkeit oder wegen von den Gesetzgeber vorgegebener Rahmenbedingungen und Aufbewahrungsfristen [VDA05]. Aufgrund der Vielfalt der eingesetzten Systeme ist die Festlegung auf bestimmte Inferenzverfahren und -Implementierungen nicht zweckmäßig. Stattdessen sollen Systemkomponenten und ihre Schnittstellen spezifiziert werden, um die Umsetzung im Lauf der Zeit so anpassen zu können, dass keine anderen Komponenten des Systems davon betroffen werden.

4.5. Anforderungen an die Bereitstellung der Anfrageergebnisse

Zur Durchführung der Informationsintegration muss das System in der Lage sein, alle Ausprägungen der relevanten Produktdaten mehrerer zu integrierenden Schemata aus einer heterogenen, in einem Netzwerk verteilten Datenlandschaft zu ermitteln, und daraus einen produktbeschreibenden Datensatz zu bilden. Die definierten Axiome müssen für einen solchen Datensatz evaluiert werden können, um auf diese Weise weitere Produktdaten zu ermitteln und eine Einordnung in ein Antwortschema durchzuführen.

Die Anfrageergebnisse, d.h. die vollständigen, in ein Antwortschema einsortierten Datensätze aller integrierten Produktdaten, sollen für unterschiedliche Arten der Weiterverwendung bzw.

Weiterverarbeitung über eine anwendungsneutrale Schnittstelle bereitgestellt werden.

Die Steuerung des Systems zur semantischen Informationsintegration, d.h. die Festlegung des zu integrierenden Produktdatenbestandes, Abrufen der Anfrageergebnisse etc., soll für beliebige Anwendungsapplikationen erfolgen können, um eine vollständige Integration vorhandener Softwaresysteme in das Integrationssystem zu ermöglichen.

So sollen die Anfrageergebnisse dann beispielsweise dazu verwendet werden, die Entscheidungsfindung in der Produktentwicklung zu unterstützen oder für die Realisierung einer automatischen Warnung bei Datenredundanz während der Produktentstehungsphase zu sorgen. Hier ist eine Vielzahl unterschiedlicher Anwendungsszenarien denkbar.

4.6. Sicherheitsrelevante Anforderungen

Die Integration heterogener Informationssysteme führt dazu, dass produktbeschreibende Daten durch die Zugriffsmechanismen der jeweiligen Datenquellen nicht mehr geschützt sind. Das bedeutet, ein Anwender kann unter Umständen durch die systemübergreifende Informationsbereitstellung produktbeschreibende Daten einsehen, wenngleich dazugehörige Zugriffsrechte für den jeweiligen Datenbestand fehlen. Aus diesem Grund soll das System zur semantischen Informationsintegration über ein eigenes Rechtekonzept zur Benutzerverwaltung verfügen, um Identifizierung und Authentifizierung der Benutzer sicherzustellen.

Hierzu müssen Lese- und Schreibrechte für die Benutzer zur Verwaltung und Pflege der konzeptionellen Struktur geregelt werden. Leserechte für integrierte Datenquellen bzw. darin befindlichen Daten sowie Leserechte zur Erhebung relevanter Informationen.

4.7. Anforderungen an die Architektur des Systems

An das System zur semantischen Informationsintegration in der Produktentwicklung ergeben sich eine Reihe von Anforderungen an die Architektur bezüglich Modularität und Erweiterbarkeit. Ein Schnittstellenmodul bietet einen zentralen Zugriff auf das Integrationssystem mittels Middleware, wie ODBC, Webservice, SPARQL etc. oder API. Ein zentrales Integrationsmodul führt dann die Integration des verfügbaren Datenbestandes gemäß einer auf Beschreibungslogik basierenden Wissensbasis durch. Die Wissensbasis stellt somit eine Art *Metadata-Katalog* bzw. die globale Sicht auf die darunterliegenden Datenquellen dar. Darüber hinaus verfügt das Integrationsmodul über einen Abfrageprozessor, der für die Föderation, Optimierung und Ausführung von Anfragen an das System verantwortlich ist.

Da eine Integration beliebiger Datenquellen durchführbar sein soll, ist ein Modul zur Registrierung und Verwaltung dieser Softwaresysteme erforderlich. Ein Adaptermodul verwaltet verschiedene spezielle Schnittstellen für den lesenden Zugriff auf die unterschiedlichen Softwaresysteme. Im Fall eines mediatorbasierten Systems handelt es sich hierbei um sogenannte

Wrappers. Die einzelnen Adapter sollen hierbei einen möglichst geringen Funktionsumfang aufweisen, um eine einfache und schnelle Implementierung solcher Adapter für den Zugriff auf weitere Datenquellen zu gewährleisten (vgl. Abb. 4.1).

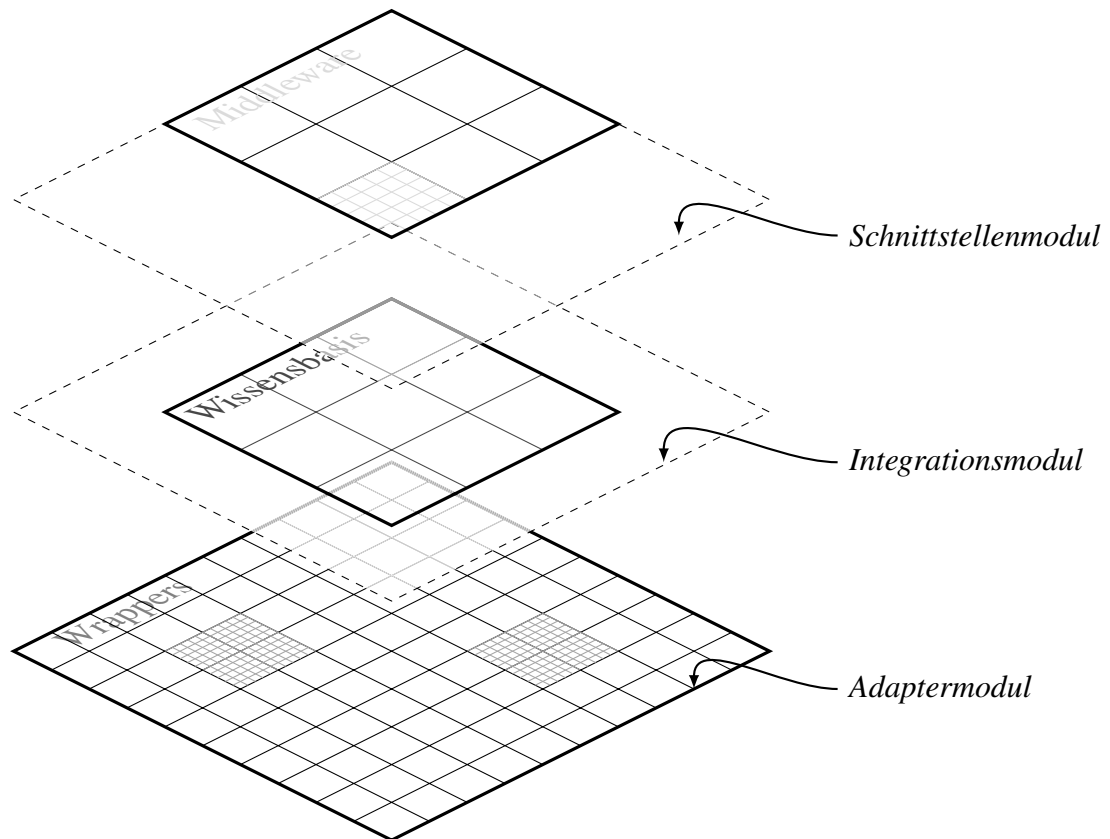


Abbildung 4.1.: Modularer Aufbau des Integrationssystems.

*Das Ergebnis habe ich schon, jetzt
brauche ich nur noch den Weg, der zu
ihm führt.*

CARL FRIEDRICH GAUSS (1777–1855)

5. Konzept zur semantischen Informationsintegration in der Produktentwicklung

Unter Berücksichtigung der aufgestellten und beschriebenen Anforderungen wird in den folgenden Abschnitten das Lösungskonzept für eine rechnergestützte semantische Informationsintegration hergeleitet. Vor dem Hintergrund, dass sich ein Wandel von der Daten- und Informations- zur Wissensgesellschaft vollzieht, gilt es, die Bewahrung von Wissen besonders zu beachten. Während die Halbwertszeit von Wissen zunehmend kürzer bzw. die Informationsmenge immer größer wird, nimmt die Wissensintensität sowohl im produzierenden Gewerbe als auch im Dienstleistungssektor stetig zu. Dabei umfasst der in diesem Kapitel beschriebene Lösungsansatz zur semantischen Produktdatenintegration in heterogenen Datenbeständen alle Phasen von der Auswahl der Datenbestände, deren Analyse, über die Konzept- und Rollengewinnung, den Aufbau einer angepassten, auf Beschreibungslogik basierenden Wissensbasis, bis hin zur automatischen Durchführung der Schlussfolgerung und Bereitstellung der Anfrageergebnisse.

Die ein Integrationssystem beschreibenden Informationen (z. B. Konzeptdefinitionen, Rollenspezifikationen) bilden die produktbeschreibenden Informationen in einer für die semantische Integration geeigneten Struktur und Ausdruckstärke ab. Die Summe dieser Informationen wird als *Metadaten* bezeichnet. Metadaten sind nach OVTCHAROVA:

[...] Informationen über Daten, die den intelligenten und effizienten Zugriff auf Daten ermöglichen. Metadaten sind demzufolge die Grundvoraussetzung für ein wirkungsvolles Datenmanagement. [OJ94]

Im Kontext der Informationsintegration umfassen Metadaten die in Abbildung 5.1 dargestellten Informationen:

- Metadaten bilden eine integrierte ontologische Sicht auf die zu integrierende Daten. Die Gesamtheit aller Datenquellen wird als *Nutzdaten* bezeichnet. Für jede einzelne Datenquelle, die über relevante Nutzdaten verfügt wird eine relationale Sicht gebildet.
- Mit einer Ontologiestruktur erfolgt die strukturierte graphbasierte Wissensrepräsentation mithilfe von spezifizierten Konzepten und Rollen. Konzepthierarchien stellen flexible, den jeweiligen Anwendungen anpassbare *Ontologieschemata* dar.
- Die Hauptprämisse zur Aufstellung eines semantischen Integrationssystems ist die Möglichkeit, Ontologiekonzepte zu identifizieren und zu erzeugen. Das wesentliche Charak-

teristikum eines Konzepts ist die *definierende Eigenschaft*, anhand deren Bewertung festgestellt wird, ob die zu integrierenden Nutzdaten zum Konzept der globalen Ontologie bzw. Wissensbasis zugeordnet werden oder nicht.

- Die Abbildung der in Nutzdaten gespeicherten Eigenschaften von Objekten (*Nutzdatenattribute*) erfolgt durch Abbildungsvorschriften, *Mappings* genannt. Im Rahmen des Mappings werden gleichwertige und vergleichbare Sachverhalte, Begriffe und Konzepte, die u. U. unterschiedlich beschrieben und benannt werden, miteinander in Beziehung gesetzt. Durch die Abbildung semantischer Beziehungen zwischen korrespondierenden Bereichen unterschiedlicher ontologischer Konzepte werden unterschiedliche Sichten miteinander in semantischen Bezug gebracht.

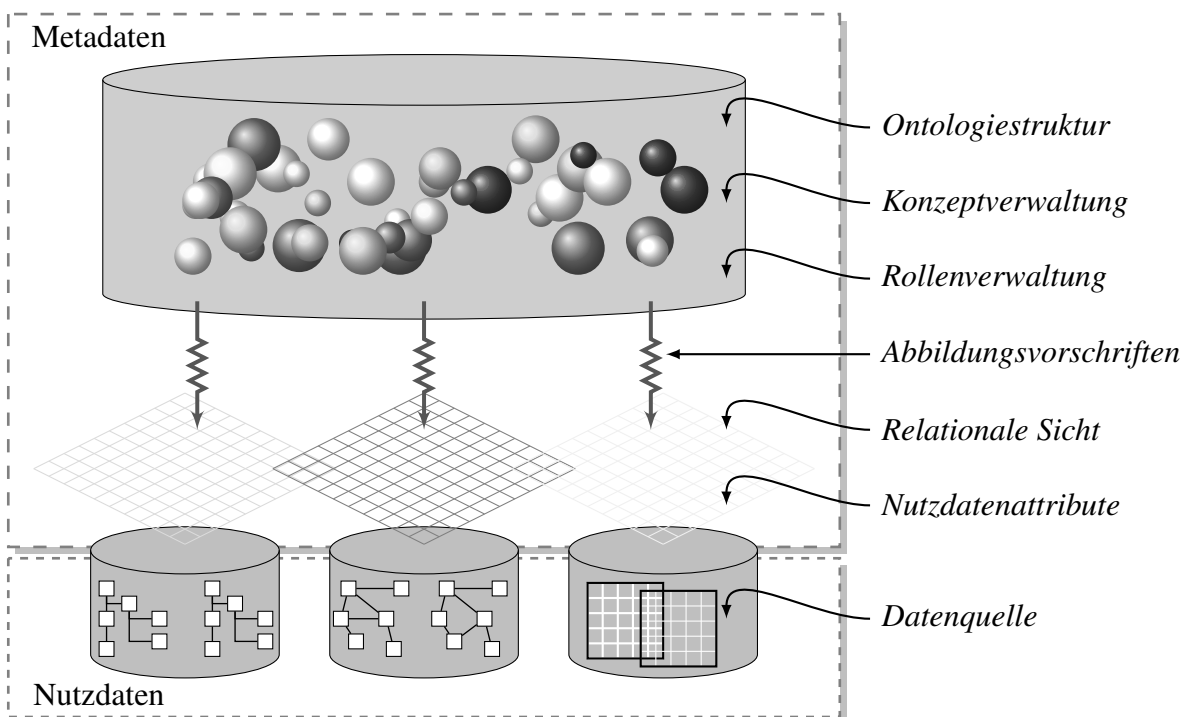


Abbildung 5.1.: Informationsinhalte der Metadaten

Im Lösungsansatz wird davon ausgegangen, dass Wissen in verschiedenen Formen explizit oder auch implizit vorliegen kann. Dabei muss implizites in explizites Wissen durch Externalisierung überführt werden, denn nur explizites Wissen lässt eine rechnerunterstützte Nutzung und Weiterverarbeitung zu. Hierbei können zwei unterschiedliche Verfahren zur Generierung von Metadaten aus gegebenen Datenbeständen angewandt werden: rechnerunterstützt oder manuell. Bei der automatischen Übertragung muss das Wissen in einem entsprechenden Format, z. B. in Form einer RDF/XML¹- oder Turtle²-Datei vorliegen, sodass es in das Integrationssystem ein-

¹<http://www.w3.org/TR/REC-rdf-syntax/>

²<http://www.w3.org/TeamSubmission/turtle/>

gefügt werden kann. Hierbei kann auf vorhandene Open-Source-Werkzeuge wie D2RQ³ oder Virtuoso⁴ zurückgegriffen werden. Bei der manuellen Generierung sieht der Ansatz vor, dass vorhandenes Wissen, das beispielsweise in Dokumenten oder Berichten erarbeitet bzw. zur Verfügung gestellt wird, zuerst überprüft und für gültig erklärt werden muss, bevor dieses in das Integrationssystem aufgenommen werden kann.

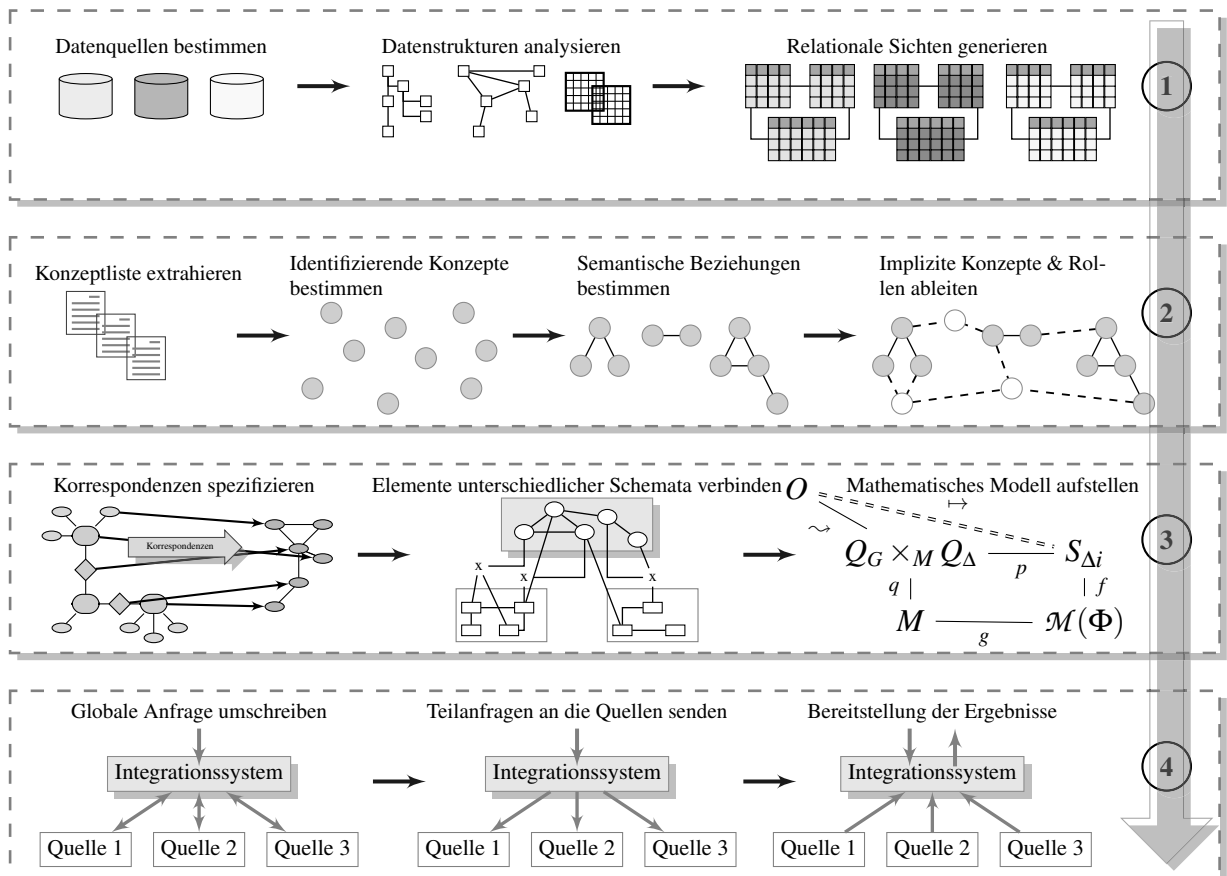


Abbildung 5.2.: Die vier Phasen des Lösungskonzepts

Es ist jedoch von essenzieller Bedeutung, dass jede einzelne Aktivität der automatischen Generierungsprozesse jederzeit lückenlos rückverfolgbar und bei Bedarf manuell änderbar ist. Damit können die Integrationsergebnisse einerseits besser nachvollzogen, andererseits einfacher verifiziert werden und somit der sogenannten Blackbox-Verhaltensweise entgegengewirkt werden. Gleichzeitig sind eingesetzte Algorithmen nicht imstande, implizit vorhandene Informationen aus den Datenquellen abzuleiten und daraufhin eine optimale Ontologiestruktur bezüglich der Ausdrucksstärke zu erzeugen. Darüber hinaus hängt die Struktur der Wissensbasis sehr stark von der subjektiven Sichtweise des Anwenders ab. Die automatisch generierten Elemente der

³<http://www4.wiwiss.fu-berlin.de/bizer/d2rq/>

⁴<http://virtuoso.openlinksw.com/>

Ontologie sind somit als Vorschläge des Systems zu betrachten, die die domänenspezifische Erstellung einer Ontologie zwar sehr gut unterstützen, die endgültige Entscheidung obliegt jedoch immer dem Anwender. Nach NONAKA und TAKEUCHI sind hierfür die sogenannten Wissenspraktiker verantwortlich, die sowohl implizites als auch explizites Wissen sammeln und erzeugen [NT97].

In Abbildung 5.2 ist eine Übersicht über den Lösungsansatz dargestellt. Es handelt sich um einen sequenziellen Ablauf, der jedoch auch Iterationen und Rücksprünge aufweisen kann. Der gesamte Informationsintegrationsprozess besteht aus den vier Phasen Datenanalyse, Konzept- und Rollenableitung, Aufbau des Integrationssystems sowie Anwendung des Integrationssystems.

Phase 1: Datenanalyse

Zu Beginn der Datenanalyse müssen die Datenquellen festgelegt werden, die die zu integrierenden Datenbestände umfassen. Hierbei kann es sich um zahlreiche heterogene informationstechnische Systeme handeln, die zwar Daten zum gleichen Produkt, jedoch aus unterschiedlichen Produktentwicklungsphasen erfassen. Sind die Datenquellen ermittelt, so muss als nächste Aktivität die Analyse der Datenmodelle der ausgewählten Datenquellen durchgeführt werden. Dabei können betrachtete Datenmodelle netzwerkartige, hierarchische, objektorientierte oder relationale Strukturen aufweisen.

Um einen systemunabhängigen Zugriff auf heterogene verteilte Datenmodelle sicherzustellen, ist die Generierung jeweils einer geeigneten lokalen Sicht auf die einzelnen Datenquellen notwendig. Die nachstehenden Punkte erläutern die grundlegende Motivation zur Verwendung eines relationalen Datenmodells, um geeignete Sichten auf heterogene Datenquellen bilden zu können:

- Die Mehrzahl der heute am Markt befindlichen Datenmanagement-, PDM/PLM- und ERP-Systeme verwendet ein relationales Modell.
- Das relationale Datenmodell stellt zahlreiche Werkzeuge zur Verfügung, um darauf relativ einfach zugreifen zu können. Darüber hinaus erlaubt es eine schnelle Entwicklung eigener Schnittstellen zum Datenaustausch.
- Alle anderen Datenmodellarten können aufgrund der generischen und flexiblen Struktur relationaler Datenmodelle relational beschrieben werden. Hierfür bietet das relationale Datenmodell hinreichend mächtige Konstrukte.
- Im Rahmen dieses Konzepts werden Schema-Mapping-Verfahren (oder Schema-Abbildung-Verfahren) zur automatischen Erzeugung der Metadaten angewandt.

Schema Mapping ist ein komplexer Prozess, der – ausgehend von der Menge von Korrespondenzen zwischen Nutzdatenattributen unterschiedlicher Schemata (Wertkorrespondenzen) – komplexere Schemakorrespondenzen und schließlich Datentransformationsvorschriften ableitet.

Die heterogene Datenlandschaft ist hierdurch als eine Menge relationaler Sichten auf die Datenquellen beschrieben.

Phase 2: Konzept- und Rollenermittlung

Die zweite Phase umfasst die Konzept- und Rollenermittlung für die Wissensbasis. Im ersten Schritt werden alle explizit in den lokalen Sichten verfügbaren Metadaten ermittelt. Anschließend werden diejenigen Konzepte ermittelt, die über eine identifizierende Eigenschaft verfügen, d. h. deren Individuen diesem Konzept semantisch zugeordnet werden können. Hierbei kann es sich auch um eine Kombination von Konzepten handeln.

Im nächsten Schritt werden die bislang ermittelten Konzepte mittels Rollen semantisch verknüpft. Anschließend werden logische Formeln der Beschreibungslogik spezifiziert, indem durch Axiome angegeben wird, in welcher Beziehung komplexe Konzepte und Rollen zueinander stehen.

Für integrationsrelevante Konzepte werden anschließend weitere Konzepte und Rollen ermittelt, die nicht explizit in den Datenquellen verfügbar sind. Diese impliziten Konzepte werden durch Verknüpfungsoperationen aus bereits verfügbaren Konzepten und Rollen definiert.

Die erstellte Ontologie wird hauptsächlich zur Überbrückung semantischer Heterogenität verwendet. Ihre Verwendung zur Überbrückung struktureller und semantischer Heterogenität oder zum Umgang mit beschränkten und unvollständigen Quellen wird mittels entwickelter Methoden sichergestellt. Diese Methoden weisen eine hohe Komplexität auf, da sie Verfahren der Datenbankwelt mit Verfahren aus der Ontologiewelt vereinen und selbstverständlich noch von Hand verbessert bzw. verfeinert werden müssen.

Phase 3: Aufbau eines Integrationssystems

In der dritten Phase wird in Anlehnung an die Definition 4 das Integrationssystem für heterogene und verteilte Datenbestände in der Produktentwicklung aufgestellt. Hierzu werden Abbildungsvorschriften anhand von LaV- und GaV-Verfahren (siehe Abschnitt 2.2.2) definiert, wodurch die beschreibungslogische Wissensbasis mit den lokalen Sichten der Datenquellen verknüpft wird. Darauf basierend wird ein mathematisches Modell aufgestellt, das eine Grundlage für eine formale Spezifikation der in den vorangegangenen Phasen bzw. Abschnitten erarbeiteten Teilkonzepte bildet. Mithilfe dieses Modells ist eine systematische und strukturierte Beschreibung von Entitäten der realen Welt möglich.

Phase 4: Anwendung eines Integrationssystems

Die letzte Phase umfasst den Einsatz und die Informationsbereitstellung eines Informationsintegrationssystems. Dieses enthält vorderhand eine initiale Integration, bei der die bestehenden Datenquellen entsprechend der in den vorangegangenen Phasen definierten Ontologie integriert werden. Anschließend erfolgt die Bereitstellung der Ergebnisse auf eine konjunktive Anfrage an das Integrationssystem. Hierzu müssen logische Inferenzmechanismen über die beschreibungslogische Ontologie ausgeführt werden, sodass sichere Antworten seitens des Systems ermittelt werden. Des Weiteren soll bei einer Modifikation in den lokalen Quellen eine erneute Integration stattfinden, bei der ausschließlich die geänderten bzw. neuen Datenbestände integriert werden. Neue oder geänderte Datenbestände beanspruchen eine erneute Überprüfung und gegebenenfalls eine Adaptierung des Integrationssystems. Das Lösungskonzept sieht vor, dass Änderungen des Inhalts von Datenbeständen – also existenziale Änderungen – keine Auswirkung auf das Integrationssystem haben, weil nur Metadaten bzw. terminologisches Wissen in der Wissensbasis repräsentiert sind. Liegt jedoch eine Änderung der Struktur der Datenquelle vor (z. B. wegen der Ausweitung des Produktportfolios oder der Erweiterung der Systemlandschaft durch die Einführung eines neuen informationstechnischen Systems), dann muss in die erste Phase zur Durchführung der Datenanalyse zurückgesprungen werden.

In den folgenden Abschnitten werden die einzelnen Phasen des Lösungsansatzes zur semantischen Integration heterogener und verteilter Datenbestände in der Produktentwicklung detailliert betrachtet.

5.1. Datenanalyse und -aufbereitung

Der erste Schritt besteht in der Identifikation der *Datenquellen*, die die zu integrierende Daten enthalten. Hierbei muss vor allem der Aspekt in Betracht gezogen werden, dass produktbeschreibende Daten häufig mehr als in einer Datenquelle präsent sind. Beispielsweise werden in einem Produktdatenmanagementsystem Stammdaten von Komponenten sowie von gesamten Produktstrukturen verwaltet. Auf diese Weise kann das PDM-System die ersten wichtigen produktbeschreibenden Daten und Merkmale liefern. Des Weiteren werden in der Entwicklungsabteilung eines Unternehmens die Einzelteile und Baugruppen als CAD-Modelle und 2D-Zeichnungen entwickelt, anschließend mit einem Stammsatz angereichert und ebenfalls in einem PDM-System abgelegt und verwaltet. Ein ERP-System kann eine Menge weiterer produkt- und fertigungsbeschreibender Unterlagen sowie anderweitiger Berichte und Dokumente wie Testberichte, Prüfpläne, Fertigungspläne etc. erfassen und verwalten, die u. a. unterschiedliche Formate aufweisen. Erst die Gesamtheit aller Unterlagen stellt alle Informationen bereit, die benötigt werden, um ein Erzeugnis möglichst vollständig zu beschreiben. Eine weitere Schwierigkeit ist das Vorhandensein von redundanten oder gar inkonsistenten Informationen.

Hierfür schafft die semantische Informationsintegration Abhilfe, indem Duplikate und Inkonsistenzen leichter identifiziert werden können.

Auf dem heutigen IT-Markt können zahlreiche vertretene Systeme anhand der Anwendungsbereiche in sieben Gruppen eingeteilt werden: CAD-Systeme, EDM/PDM-Systeme, ERP-Systeme, relationale Datenbanksysteme, dateibasierte relationale Datenquellen, hierarchische Datenquellen und eine weitere Gruppe, die schwach strukturierte Datenquellen repräsentiert. Die Tabelle 5.1 stellt exemplarisch einige Datenquellen dar.

Gruppe	Beispiel
CAD-Systeme	Pro/ENGINEER, Unigraphics NX
EDM/PDM-Systeme	Windchill PDMLink, Teamcenter
ERP-Systeme	Infor ERP LN, SAP R/3
Relationale Datenbanksysteme	Oracle, mySQL, MS-Access
Dateibasierte relationale Datenquellen	MS-Excel, DIN 82045-2
Hierarchische Datenquellen	XML, CAEX, AutomationML
Schwach strukturierte Datenquellen	MS-Word, ASCII, Unicode

Tabelle 5.1.: Gruppenzuordnung der Datenquellen

An dieser Stelle ist zu beachten, dass mehrere Datenquellen durch ein einziges Softwaresystem erzeugt werden können. Beispielsweise kann eine Datenquelle eine Vielzahl von Angeboten enthalten, die mithilfe eines Textverarbeitungsprogramms erzeugt wurden. Würde man das gleiche Softwareprogramm zur Erstellung der Stücklisten einsetzen, so müsste eine solche Dateigruppe ebenfalls als eigenständige Datenquelle betrachtet werden.

5.1.1. Datenstrukturen und Sichtenbildung

Datenmodelle sind ein wichtiges Hilfsmittel zur Gestaltung von Datenbanken und zur Entwicklung von Software [Har94]. Nach HARS können damit grundlegende Zusammenhänge erkannt, strukturiert und erfasst werden [Har94]. Die wichtigsten logischen Datenmodelle, die heute angewandt werden, sind relationale Datenmodelle, hierarchische Modelle, objektorientierte Datenmodelle sowie Netzwerkmodelle. Als Beispiel sind in Abbildung 5.3 vier Schemata zu sehen, die in unterschiedlichen Datenmodellen vorliegen, aber in etwa dieselben Informationen darstellen können.

Das objektorientierte Modell verwendet eine Spezialisierungsbeziehung, um gemeinsame Attribute von Produkten, die zum einen für den deutschen Binnenmarkt, zum anderen für den internationalen Markt bestimmt sind, in einer Klasse zu vereinen. Das relationale Modell speichert

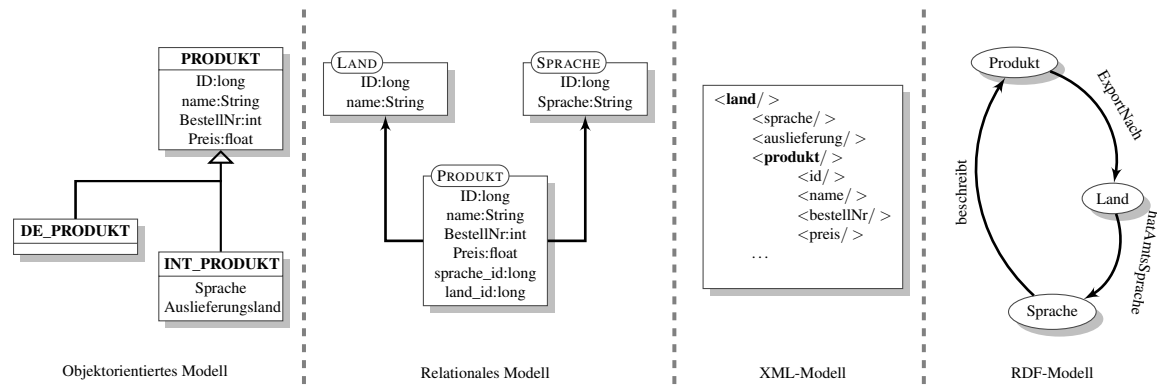


Abbildung 5.3.: Vier nahezu äquivalente Schemata in unterschiedlichen Datenmodellen

alle Produkte in einer Tabelle produkt, verwendet zur Speicherung des Auslieferungslandes und der Sprache eines Produkts aber eigene Tabellen, die über Fremdschlüsselbeziehungen mit der Tabelle produkt verbunden sind. Das XML-Modell verwaltet Produkte geschachtelt unter ihrem Auslieferungsland und der dazugehörigen Sprache des Produkts. Schließlich sind Konzepte (Produkt, Sprache, Land) im Netzwerkmodell graphartig miteinander verbunden und die Entitäten sind ganzheitlich dargestellt.

Zwar nehmen objektorientierte Datenmodelle beim Entwurf von Informationssystemen und Software eine zunehmend größere Rolle ein, doch haben sie in Bezug auf die Produktdatenerfassung in gegenwärtig auf dem informationstechnischen Markt vorhandenen Systemen einen unwesentlichen Stellenwert. Heute eingesetzte objektorientierte Datenbanksysteme sind größtenteils als Spezialexemplare zu betrachten, die in der industriellen Praxis nicht weitverbreitet sind. Im Vergleich zur Anfragesprache SQL für relationale Datenbanken, fehlt für objektorientierte Datenbanksysteme eine allgemein anerkannte Anfragesprache. Im Rahmen der vorliegenden Arbeit wird aufgrund der genannten Defizite auf objektorientierte Datenbankmanagementsysteme nicht näher eingegangen.

Die im vorangegangenen Abschnitt aufgeführten Datenquellengruppen haben entweder relationale Strukturen (SQL-Datenbanksysteme, EDM/PDM/PLM- und ERP-Systeme, anderweitige relationale Datenquellen), netzartige Datenstrukturen (CAx-Systeme), hierarchische Datenstrukturen (z. B. SGML, XML) oder sie weisen keine wohldefinierte Datenstruktur auf (Textdateien, ASCII). Die Auswahl einer Datenquelle für semantische Informationsintegration setzt deshalb eine Strukturermittlung voraus, um die nachfolgende Aufstellung einer relationalen Sicht auf diese Datenquelle durchzuführen. Der Grund dafür liegt in der Gewährleistung eines einheitlichen Zugriffs auf unterschiedliche Datenquellen mithilfe einer allgemeingültigen konzeptionellen Schicht. Das relationale Datenmodell ist die gegenwärtig am weitesten verbreitete Datenstruktur und bildet daher die Grundlage für die meisten gängigen Datenbanksysteme, wie EDM/PDM-Systeme und ERP-Systeme. Informationstechnische Systeme, die über kein

relationales Datenmodell verfügen, können dennoch durch eine relationale Datenstruktur ausgedrückt werden. Dem in dieser Arbeit dargestellten Lösungskonzept für die semantische Informationsintegration liegt ferner eine relationale Sicht für jeweils eine zu integrierende Datenquelle zugrunde. Ergänzend deckt der entwickelte Lösungsansatz ebenfalls die netzwerkartige ontologische Struktur als eine Sicht für die zu integrierende Datenquellen ab.

Beim relationalen Datenmodell stehen als Strukturelemente ausschließlich Relationen zu Verfügung, die sich durch Tabellen darstellen lassen [LLS05]. Die Datensätze bilden die Zeilen bzw. die Tupel und die Eigenschaften des Objekts bzw. die Datenfelder entsprechenden Spalten der Tabelle [Kar12]. Der Zugriff auf bestimmte Datensätze wird über die Feldinhalte ermöglicht [LLS05].

Für eine relationale Beschreibung verschiedener Datenquellen werden somit drei Strukturelemente benötigt: *System*, *Entität* und *Attribut*. Unter dem Parameter „System“ ist die eindeutige Bezeichnung einer Datenquelle zu verstehen, die einen Teil der zu integrierenden Daten enthält. Innerhalb eines Systems existieren Objekte, in denen Informationen zu einer logischen Einheit zusammengefasst und persistent abgelegt werden. Ein derartiges Objekt wird als Entität bezeichnet und besteht aus einer endlichen Menge an Eigenschaften, die dieser Entität zugeordnet sind. Ein Attribut beschreibt eine einzige Eigenschaft einer Entität. Im Fall einer SQL-Datenbank, die von sich aus bereits eine relationale Struktur besitzt, wäre das System die Datenbank selbst, die Entitäten, die in der Datenbank enthaltenen Tabellen und die Attribute die Spaltenüberschriften der einzelnen Tabellen.

5.1.2. Relationale Sicht auf hierarchische Datenstrukturen

Jedes in hierarchischen Datenstrukturen vorkommende Element verfügt genau über ein Vorfahrenelement. Einen Ausnahmefall stellt logischerweise das Wurzelement dar, da es keinen Vorfahren hat. Zugleich kann jedes Datenelement keine, eine oder mehrere Nachfolge- oder Kindelemente haben. Diese Form einer hierarchischen Struktur wird auch *Baumstruktur* genannt. In Abbildung 5.4 ist eine Baumstruktur beispielhaft repräsentiert. Ein typischer Vertreter der Baumstruktur stellt das XML-Format dar.

Um die Transformation von jedem in der Baumstruktur vorkommenden Knoten durchführen zu können, wird hierfür eine entsprechende Entität des relationalen Modells erstellt. Dabei werden nur solche Knoten in Betracht gezogen, die keine Werte besitzen. Die mehrfach auftretenden Knotenbezeichnungen werden einer gemeinsamen Entität zugeordnet. Dadurch wird die Eindeutigkeit von Entitätsbeziehungen sichergestellt. Bezug nehmend auf das vorangegangene Beispiel sind das die Entitäten *Katalog*, *Hersteller*, *Produkt* und *Preis*.

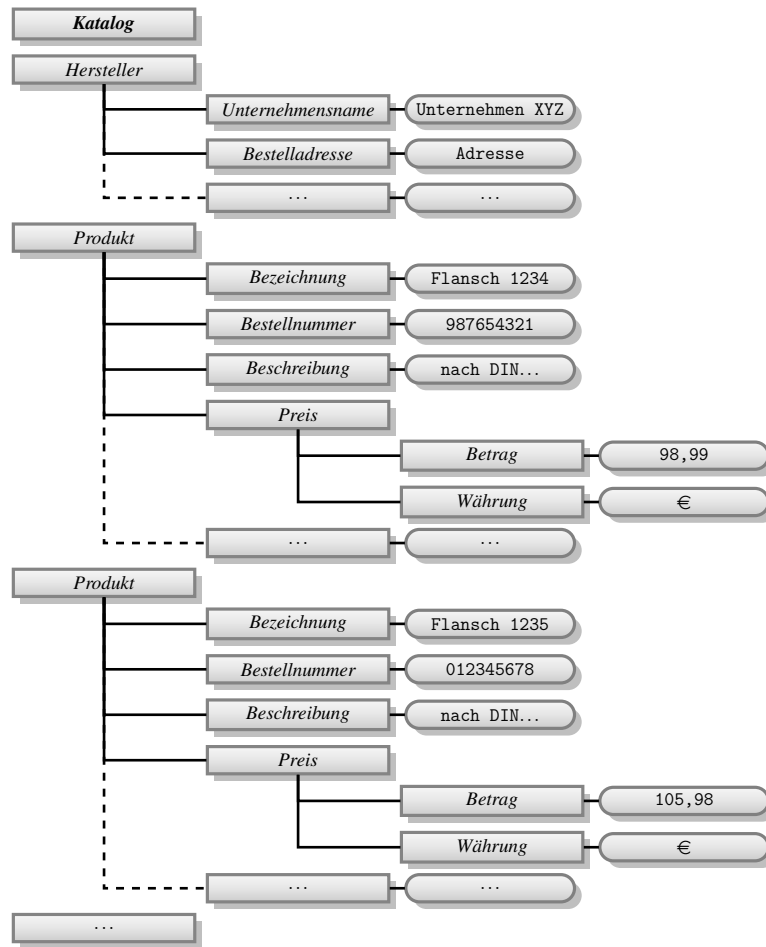


Abbildung 5.4.: Beispiel einer Baumstruktur [Ehr95]

Analog dazu erfolgt die Überführung der Attribute aus der Baumstruktur in das relationale Datenmodell – mit dem Unterschied, dass nur diejenigen Knoten zu Attributen werden, die über einen Wert verfügen. Konkret bedeutet das: Die Entität *Katalog* aus dem vorhergehenden Beispiel bekommt vorerst kein Attribut, dagegen werden der Entität *Preis* zwei Attribute, nämlich *Betrag* und *Währung* zugewiesen. Grundsätzlich ist die Erzeugung weiterer Beziehungsentitäten für hierarchische Datenstrukturen nicht notwendig. Sollte dennoch eine 1 : n-Beziehung abgebildet werden, z. B. in dem Fall, dass einem Katalog mehrere Produkte zugewiesen werden, erfolgt dies über ein zusätzliches ID-Attribut. Dabei ist zu achten, dass ID-Attribute mit einem eindeutigen Wert belegt werden.

5.1.3. Relationale Sicht auf schwach strukturierte Datenstrukturen

Häufig weisen integrationsrelevante Daten nur sehr schwach strukturierte Datenkonstrukte auf. Dies kommt z. B. dann vor, wenn produktbeschreibende Informationen in Form eines formatlosen Textdokuments zur Verfügung stehen. Um ebenfalls diese schwach strukturierte Datenquellen bei der Integration zu berücksichtigen, ist es notwendig, schwach strukturierte oder gar

strukturlose Daten wie ASCII- oder UTF8-Textdokumente durch ein relationales Datenmodell abzubilden.

In den zu integrierenden Datenquellen vorkommende Textdokumente können eine Struktur z. B. in Form einer Unterkapitelgliederung aufweisen; dennoch kann dies bei der Integration nicht vorausgesetzt werden. Die (weitgehend) generelle relationale Darstellung von Dokumenten dieser Art kann somit anhand einer einzigen Entität *Dokument* mit den entsprechenden Attributen *Dateibezeichnung* und *Textinhalt* erfolgen. Ferner ist für jedes Dokument ein vollständiger Dateiname (Dateibezeichnung) mit dem sämtlichen Inhalt des Dokuments (Textinhalt) hinterlegt. Damit ist gewährleistet, dass der Inhalt dieser schwach strukturierten Datenquellen durch nachgeschaltete Methoden und Ansätze (z. B. Schlüsselwortsuche) erhoben und bei der Integration mitberücksichtigt und verwertet werden kann. Einige dieser Ansätze zur Ermittlung von produktbeschreibenden Eigenschaften aus textuellen Dokumenten basieren auf Text-Mining-Technologien, wie *Probabilistic Ontology Model* oder *Structural Semantic Interconnections* und können beispielsweise mithilfe der Werkzeuge Text2Onto⁵, OntoLT⁶, OntoLearn⁷ erfolgen.

5.2. Konzept- und Rollenermittlung

Die Grundlage für ein System zur semantischen Informationsintegration in der Produktentwicklung ist die Bereitstellung eines allgemein verfügbaren und anerkannten Begriffswortschatzes. Um den im Abschnitt 4.3 aufgestellten Anforderungen gerecht zu werden und Wissen in Form von beschreibungslogischer Ontologie darstellen zu können, ist der strukturierte Einsatz von semantischen Konzepten und dazugehörigen Rollen bzw. Relationen erforderlich. Dies setzt jedoch eine gezielte Auswahl von semantisch korrekten Begriffen und Relationen voraus, um eine offene, flexible und strukturierte Erstellung einer Wissensbasis zu gewährleisten, die sicherstellt, dass das Wissen konsistent und eindeutig gespeichert ist. Hierbei soll auf die Ausdrucksstärke von semantischen Konzepten und Rollen geachtet werden, sodass die erstellte Ontologie in der Praxis eingesetzt werden kann.

Die datenquellenübergreifende Konzept- und Rollenerfassung ist eine der Hauptaufgaben in der Ermittlungsphase. Dabei kommt der Verwendung von Begriffen eine wesentliche Bedeutung zu, denn eine effiziente Wissenserschließung führt nur über die Verwendung von Wörtern (z. B. allgemeine Wörter, Fachausdrücke) und über eine gemeinsame Begriffswelt zu einem einheitlichen Verständnis. Je genauer die Konzepte beschrieben werden können und je genauer ein Anwender ausdrücken kann, an welchen Dingen er interessiert ist, desto leichter kann das

⁵<http://ontology-learning.net/wiki/TextToOnto>

⁶<http://olp.dfki.de/OntoLT/OntoLT.htm>

⁷<http://lcl2.uniroma1.it/home.jsp>

relevante Wissen gefunden werden. Die gemeinsame Basis hierzu sind Konzepte, die benannt und definiert werden können und die durch Rollen/Relationen mit weiteren Begriffen zu einem Netzwerk verbunden werden.

Die Extrahierung relevanter Konzepte für die Ontologie ist ein Prozess, der nicht vollständig automatisch ablaufen kann und eine manuelle Verifikation und ggf. Korrektur bzw. Vervollständigung erfordert. Dies liegt darin begründet, dass nicht explizit aus den Datenquellen ermittelbare Beziehungen zwischen Entitäten aus den verfügbaren Datenbeständen abgeleitet werden müssen. Um alle Konzepte automatisch ermitteln zu können, die eine Datenquelle vollständig beschreiben, ist die explizite Abbildung jeder Entität aller Datenquellen erforderlich. Hierbei soll eine Abgrenzung zwischen den Konzepten erfolgen. Der Konzeptermittlung sollen Merkmale zugrunde gelegt werden, die eine klare und eindeutige Begriffsabgrenzung ermöglichen. Des Weiteren soll auf der linguistischen Ebene auch auf die klare Begriffsbenennung geachtet werden. Ausdruck von Konzepten zweiter Ordnung (Wissen über Wissen = Metawissen) soll aufgrund algorithmischer Komplexität ausgeschlossen werden. Gängige Normen und Richtlinien, wie beispielsweise „Begriffe und Benennungen; Allgemeine Grundsätze“ [DIN2320] sollen bei der Synonymdefinition berücksichtigt werden. Ferner sind lexikalische Grundlagen, in diesem Fall *Homonyme*, einzubeziehen.

Eine automatische Ermittlung aller Daten, die eine zu integrierende Datenquelle beschreiben, erfordert eine explizite Abbildung aller Beziehungen zwischen den Entitäten aller Datenquellen. Im Folgenden Abschnitt wird zunächst die Semantik der Beschreibung dieser Beziehungen untersucht. Anschließend wird ein Verfahren zur automatischen Ermittlung der expliziten Konzepte und Rollen aus den verfügbaren Datenquellen beschrieben.

5.2.1. Ansatz zur automatischen Konzept- und Rollenermittlung

Zur Ableitung von Konzepten und Rollen wird das im Folgenden beschriebene allgemeingültige Verfahren herangezogen, mithilfe dessen relationale Datenbankstrukturen durch Ontologieformalismen repräsentiert werden. Zuvor müssen jedoch einige grundsätzliche Definitionen aufgestellt werden.

Definition 21: Ein relationales Datenbankschema D ist eine endliche Menge von Entitäten $D = \{E_1, \dots, E_n\}$, wobei jede Entität durch eine endliche Menge von Attributen $\{A_{i1}, \dots, A_{im}\}$ charakterisiert ist.

Definition 22: Eine Abbildungsvorschrift $pkey$ weist jeder Entität ihren Primärschlüssel (engl. primary key) zu. Dabei ist ein Primärschlüssel durch eine endliche Menge von Attributen $K \subseteq E$ beschrieben.

Definition 23: Eine Instanz t über der Entität E ist eine Menge von Tupeln, die die Werte von $|E|$ enthält. Ferner ist die Datenbank d über D definiert als eine Menge von Instanzen $d = \{t_1, \dots, t_n\}$.

Zur Erfassung von Korrelationen zwischen den Entitäten werden Inklusionsabhängigkeiten benutzt. Hierzu ist eine Inklusionsabhängigkeit ein Ausdruck folgender Form: $R[X] \subseteq S[Y]$, wobei X und Y entsprechende Attributlisten der Entitäten R und S sind. Darüber hinaus soll bei der Inklusionsabhängigkeit die Restriktion $|X| = |Y|$ erfüllt sein. Es besteht eine Abhängigkeit zwischen zwei Instanzen r und s eines relationalen Schemas, wenn für jedes Tupel u in r ein Tupel v in s existiert, sodass $u[X] = v[Y]$ gilt. Mit anderen Worten, eine Inklusionsabhängigkeit ist eine geeignete Art, um auszusagen, dass Datensätze lediglich von einer anderen Entität kopiert sind. Die Fremdschlüsselbeziehung kann als Inklusionsbeziehung definiert werden, indem eine zusätzliche Eigenschaft hinzugefügt wird: $Y = \text{pkeys}(S)$. Wir verwenden die Schreibweise $R[X] \subseteq S[\text{pkeys}(S)]$, um diese spezifische Abhängigkeit zu beschreiben.

Das Hauptaugenmerk liegt hierbei auf der Untersuchung der Beziehungen zwischen den Entitäten einer Datenbank, um diese in die ontologische Struktur zu überführen. Das Datenbankschema ist die erste Informationsquelle, die es zu analysieren gilt, sodass durch die Anwendung von priorisierten Abbildungsvorschriften eine Transformation von den Elementen der DB-Schemata in die Elemente der Ontologie in Form von Konzepten und Rollen sichergestellt wird.

Hierzu sind in der Tabelle 5.2 drei bewährte Abbildungsvorschriften angegeben, die unter anderem bei mehreren bestehenden Ansätzen eingesetzt werden. Die erste triviale Vorschrift besagt, dass jede Entität eventuell in ein Konzept überführt werden kann. Die zweite Abbildung ist ebenfalls eine einfache Zuordnung von einer Fremdschlüsselbeziehung zu einer funktionalen Rolle der Ontologie. Die dritte Abbildung sorgt für die Herstellung einer Beziehung zwischen einer Entität mit einem zusammengesetzten Primärschlüssel und zwei schlüsselbasierten Attributen. Diese Art der Beziehungen wird in Datenbanken eingesetzt, um zwei Entitäten anhand der Schlüsselbeziehung miteinander zu verknüpfen. Sie ist besser bekannt als $n : m$ -Beziehung.

Um diesen Ansatz besser nachvollziehen zu können, wird er nachfolgend an einem Beispiel veranschaulicht. Darin wird angenommen, dass ein Produktportfolio im semantischen Integrationssystem in Form einer Ontologie zur Verfügung gestellt werden soll, die aktuellen Produktdaten befinden sich jedoch in relationalen Tabellen einer Datenbank.

1. Entität zum Konzept		
Quelle	Vorbedingung	Ziel
$E \in D$	$\neg C \mid E = quelleVon(C)$	$concept(C_E)$

2. Fremdschlüsselbeziehung zur Rolle		
Quelle	Vorbedingung	Ziel
$E_0[A] \subseteq E_1[pkey(E_1)]$	$E_0 = quelleVon(C_0)$ $E_1 = quelleVon(C_1)$	$Rolle(R_A$ $domain(C_0)$ $range(C_1))$

3. Zusammengesetzte Schlüsselbeziehung zur Rolle		
Quelle	Vorbedingung	Ziel
$E_0 \in D$ $ E_0 = 2$ $pkey(E_0) = \{K_1, K_2\}$ $E_0[K_1] \subseteq E_1[pkey(E_1)]$ $E_0[K_2] \subseteq E_2[pkey(E_2)]$	$E_1 = quelleVon(C_1)$ $E_2 = quelleVon(C_2)$	$Rolle(R_E$ $domain(C_1)$ $range(C_2))$

Tabelle 5.2.: Drei Vorschriften zur Abbildung der Datenbankschemata

In Abbildung 5.5 ist exemplarisch ein Schemaauszug aus einer Datenquelle abgebildet, der die Ermittlung von Konzepten und Rollen einer Ontologie erfordert. Das Datenbankschema enthält fünf Entitäten einer Datenquelle, in der Kunde, Produkt, Bestellungen und Zulieferer verwaltet werden. Über zwei Beziehungen werden Bestellungen mit einem oder mehreren Produkten verknüpft, Kunden werden mit einer Bestellung assoziiert. Zudem kann ein Produkt über das Attribut *Zulieferer* einem Zulieferer zugeordnet werden.

Aus dem vorangegangenen relationalen Schemaauszug wird folgendes Ontologiemodell automatisch abgeleitet (siehe Abb. 5.6): Der erste Schritt des Ansatzes umfasst Ableitungen, die durch die Identifikation von Mustern aus dem Datenbankschema motiviert sind. Dabei steht jede Entität- (oder Tabellen)-Definition aus dem Datenbankschema für die Quelle eines Konzepts in der Ontologie. Solche einfachen Zuordnungen von Entitäten auf Konzepte sind häufig relevant, obwohl mehrere Ausnahmen behandelt werden müssen (z. B., wenn einige Entitäten eher als Konzept-zu-Konzept-Assoziationen zu überführen sind). Die Fremdschlüsselbeziehungen sind die zuverlässigste Quelle für die Verknüpfung von Konzepten. In diesem Beispiel wurde jede der vier Beziehungen in eine Rolle übersetzt.

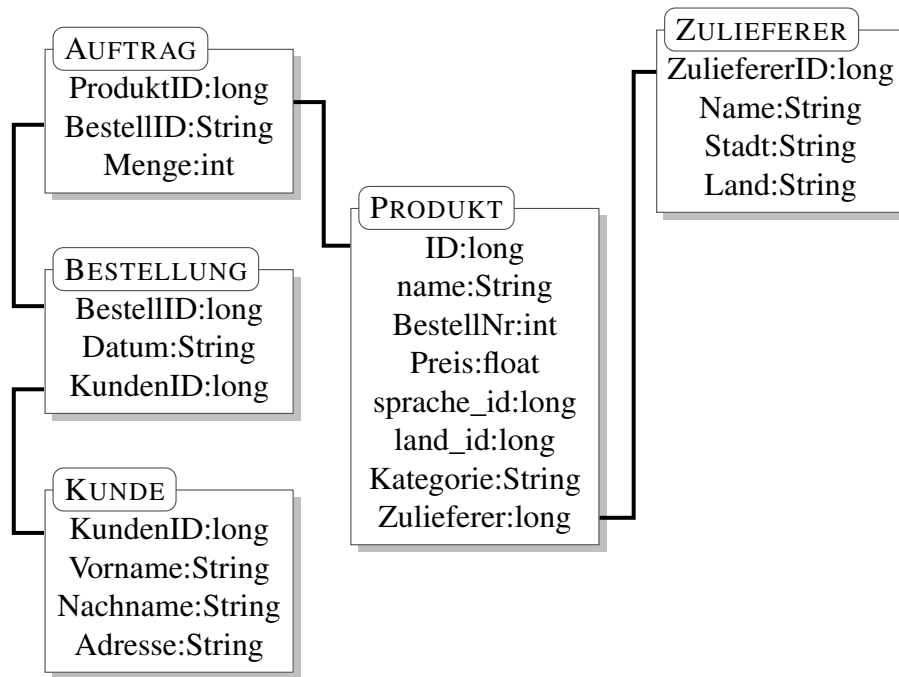


Abbildung 5.5.: Schemaauszug aus einer relationalen Datenbank

Aktuelle Methoden decken die beschriebene Ableitungsweise ab, sodass bei ihrer Anwendung auf den Schemaauszug die meisten der Methoden die dargestellte ontologische Struktur als endgültige Ausgabe liefern würden.

Wie bereits gezeigt werden konnte, werden in der ontologiebasierten Integration die Datenquellen als Konzepte der globalen Ontologie modelliert. Hierzu werden die Entitäten ihrer Datenbankschemata durch die Begriffe der Ontologie beschrieben. Dabei kann es durchaus notwendig sein, die globale Ontologie manuell zu erweitern bzw. zu verfeinern. Dies ist darin begründet, dass die globale Ontologie ein Modell eines Diskursbereichs, das zur Kommunikation zwischen unterschiedlichen Anwendern erstellt und genutzt wird, zwei Bedingungen erfüllt: Zum einen ist es derart formal und exakt, dass es zur Inferenz, also zur automatischen Ableitung von neuem Wissen durch logisches Folgern, eingesetzt werden kann. Zum anderen sind die im Modell erfassten Konzepte und Rollen innerhalb der Zielgruppe eindeutig und unumstritten. Ein anderer Aspekt, der den manuellen Eingriff bei der Ontologierstellung erfordert, ist die logische Ausdrucksstärke der Ontologie. Hierfür steht der Wissensingenieur vor einem Dilemma: Die Erstellung einer genauen und detaillierten Ontologie erfordert eine ausdrucksmächtige Wissensrepräsentationssprache. Gleichzeitig erreicht die Inferenz zur Anfragebearbeitung für solche Sprachen sehr schnell hohe Komplexität. Sehr ausdrucksstarke Wissensrepräsentationssprachen sind bei auftretenden Entscheidungsproblemen unentscheidbar.

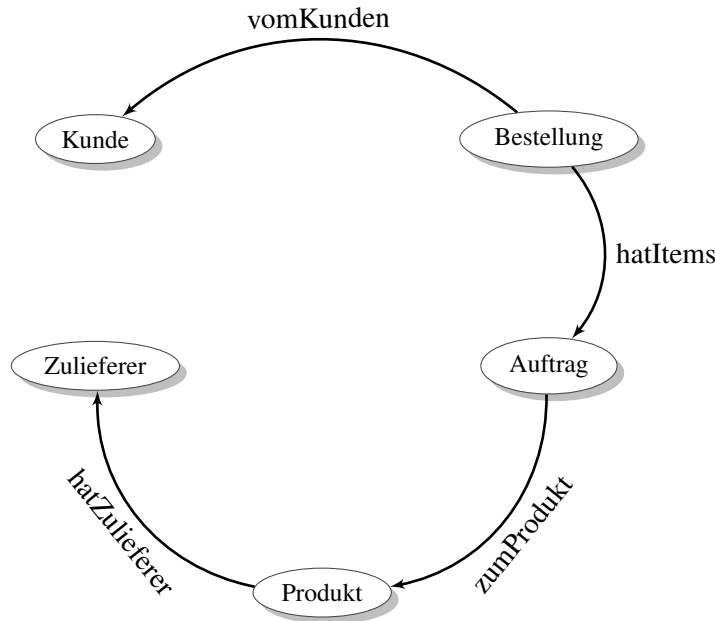


Abbildung 5.6.: Ontologiemodell zum relationalen Schemaauszug

In diesem Zusammenhang wird im nächsten Abschnitt ein Wissensrepräsentationsformalismus vorgestellt, um einen Konsens zwischen einer ausdrucksstarken globalen Ontologie und der Komplexität der Anfragebearbeitung erzielen zu können.

5.2.2. Der Wissensrepräsentationsformalismus \mathcal{L}_{tripod}

In diesem Abschnitt wird ein ausdrucksstarker Wissensrepräsentationsformalismus betrachtet, der bei dem Einsatz in semantischen Integrationssystemen eine Umschreibung konjunktiver Anfrage in relationale Algebra unterstützt. Der vorgestellte Formalismus – nachfolgend als \mathcal{L}_{tripod} ⁸ bezeichnet – verfügt über die Eigenschaft, in polynomieller Zeit eine Anfrage zu verarbeiten. Des Weiteren soll gezeigt werden, dass die Erweiterung des Wissensrepräsentationsformalismus \mathcal{L}_{tripod} zum Verlust dieser Eigenschaft führt.

Der Hauptunterschied von \mathcal{L}_{tripod} zu der in Kapitel 2.4.2 vorgestellten Sprache \mathcal{ALC} besteht einerseits darin, dass eine Reihe beschreibungslogischer Konstruktoren erlaubt sind. Andererseits führen wir bestimmte Einschränkungen auf die *Subsumptionaxiome* ein. Hierbei definiert WOODS [Woo91] die *Subsumptionsrelation* wie folgt:

$$S_2 \sqsubseteq S_1 \text{ iff } \forall x : S_2(x) \rightarrow S_1(x)$$

⁸Das Akronym kommt von: **t**ractable **i**ntegration for **p**roduct **d**evelopment, d. h. eine Sprache zur Informationsintegration in polynomieller Zeit.

d. h. S_1 subsumiert S_2 , wenn alle Elemente von S_2 auch Elemente von S_1 sind.

Die Sprache \mathcal{L}_{tripod} wird entsprechend der im Kapitel 2.4.2 betrachteten Definitionen für Beschreibungslogiken definiert.

Definition 24: Es seien N_C, N_R, N_I abzählbare, unendliche und paarweise disjunkte Mengen von Konzept-, Rollen- und Konstantennamen. Eine atomare Rolle ist ein Element von $N_R \cup \{P \mid P^- \in N_R\}$, wobei Rollen der Form P^- als inverse Rollen bezeichnet werden. Des Weiteren werden in \mathcal{L}_{tripod} Rollen nach folgenden Syntaxregeln definiert, indem R_L als eine einfache Rolle und R als allgemeine Rolle bezeichnet werden:

$$R_L ::= P \mid P^- \quad R ::= R_L \mid \neg R_L$$

Ein atomares Konzept ist ein Element $A \in N_C$. Für die Sprache \mathcal{L}_{tripod} werden die Mengen von einfachen Konzepten C_L und allgemeinen Konzepten C entsprechend folgender syntaktischer Notation definiert:

$C_L \rightarrow$	A	\mid	$\exists R$	\mid	$C_{L1} \sqcap C_{L2}$
$C \rightarrow$	\top	\mid	\perp	\mid	A
	$\neg A$	\mid	$C_1 \sqcap C_2$	\mid	$\exists R$
	$\neg \exists R$	\mid	$\exists R.C$		

Eine *allgemeine Konzeptinklusion* hat die Form $C \sqsubseteq D$ mit den Konzeptbeschreibungen C und D . Der Wissensrepräsentationsformalismus \mathcal{L}_{tripod} betrachtet eine eingeschränkte Form der *Konzeptinklusion*, indem ein einfaches Konzept in die ontologische Struktur eingefügt wird. In dieser Hinsicht kann ein einfaches Konzept durch die Verwendung einer eingeschränkten Menge von beschreibungslogischen Konstruktoren definiert werden. Im Gegensatz zur Konzeptdefinition C wird bei einfachen C_L -Konzepten eine syntaktische Untermenge verwendet.

Definition 25: Eine \mathcal{L}_{tripod} TBox ist eine endliche Menge von Konzept- und Rollenaxiomen der Form $C_L \sqsubseteq C$ und $R_L \sqsubseteq R$, wobei C_L, R_L einfaches Konzept bzw. einfache Rolle und C, R allgemeines Konzept bzw. allgemeine Rolle sind. Als Untermenge enthält die Menge der Rollenamen zudem ein Axiom folgender Gestalt $\rho(R) \sqsubseteq d$, das die Menge von Datentypen für R bestimmt (d – ist der zugelassene Datentyp).

Um die Allgemeinheit nicht einzuschränken, richtet sich der Fokus der folgenden Ausführungen

auf \mathcal{L}_{tripod} TBox \mathcal{T} in normaler Form, in der alle Axiome von der Form $A_1 \sqsubseteq (\neg)A_2$, $A_1 \sqsubseteq \exists R_{L1}$, $A_1 \sqsubseteq \exists R_{L1}.A_2$, $\exists R_{L1} \sqsubseteq A_1$, oder $R_{L1} \sqsubseteq (\neg)R_{L2}$ sind, wobei A_1 und $A_2 \in \mathcal{N}_C$, und R_{L1} und R_{L2} einfache Rollen sind. Dabei kann jede \mathcal{L}_{tripod} TBox \mathcal{T} in eine äquivalente \mathcal{L}_{tripod} TBox \mathcal{T}_{NF} in Normalform transformiert werden. Hierzu werden komplexe Konzepte durch atomare in Anlehnung an BAADER ET AL. systematisch substituiert [BBL05]. Die entsprechende Transformation wird im nächsten Abschnitt eingehender thematisiert.

Syntax	Semantik	Beschreibung
P	$P^I \subseteq \Delta^I \times \Delta^I$	Rollenname
P^-	$\{(b, a) \in \Delta^I \times \Delta^I \mid (a, b) \in P^I\}$	inverse Rolle
A	$A^I \subseteq \Delta^I$	Konzeptname
d	$d^I \subseteq \Delta^I$	Datentypname
\top	Δ^I	universelles Konzept
\perp	\emptyset	leeres Konzept
$\neg C$	$(\neg C)^I = \Delta^I \setminus C^I$	Negation von C^I in Δ^I
$\neg R$	$(\neg R)^I = (\Delta^I \times \Delta^I) \setminus R^I$	Negation von R^I in Δ^I
$\neg \exists R$	$(\neg \exists R)^I = \Delta^I \setminus R^I$	Negation von R^I in Δ^I
$C_1 \sqcap C_2$	$(C_1 \sqcap C_2)^I = C_1^I \cap C_2^I$	Schnittmenge
$\exists R$	$(\exists R)^I = \{a \in \Delta^I \mid \exists b \in \Delta^I : (a, b) \in R^I\}$	allgemeine Existenzquantifizierung
$\exists R.C$	$(\exists R.C)^I = \{a \in \Delta^I \mid \exists b \in C^I : (a, b) \in R^I\}$	Existenzquantifizierung
$\rho(R)$	$(\rho(R))^I = \{b \in \Delta^I \mid (a, b) \in R^I\}$	Datentypbereich

 Tabelle 5.3.: Semantik der Sprache \mathcal{L}_{tripod}

Die Semantik der Sprache \mathcal{L}_{tripod} ist mittels der Interpretationsfunktion I definiert. Eine Interpretation I ist ein *Modell* von \mathcal{T} , falls sie jedes Modell in \mathcal{T} erfüllt. Die Interpretationsfunktion I für die Sprache \mathcal{L}_{tripod} wird induktiv mittels Regeln definiert (siehe Tab. 5.3). Der Wissensrepräsentationsformalismus \mathcal{L}_{tripod} lässt keine Nominale $\{a_1, \dots, a_n\}$ zu und trifft die Annahme über die Einzigartigkeit der Namen in der Ontologie.

Ausgehend von der Definition der Sprache \mathcal{L}_{tripod} werden zunächst deren Eigenschaften be-

trachtet. Anschließend soll gezeigt werden, dass die Sprache \mathcal{L}_{tripod} maximal bezüglich ihrer Ausdrucksstärke ist und dass jegliche Erweiterung des Wissensrepräsentationsformalismus um andere Konstrukte oder Axiome dazu führen wird, dass es keine genauere Umformulierung konjunktiver Anfragen in die relationale Algebra gibt. Diese Überlegungen basieren dabei auf einer Reihe von bereits bekannten Teilergebnissen aus dem Bereich der Beschreibungslogiken.

Lemma 1: Sei die Sprache \mathcal{L}_{tripod} um folgende Konstruktoren erweitert:

$$C_L \rightarrow \exists R.A \quad \text{oder} \quad C \rightarrow \forall R.A$$

Dann ist die Umformulierung einer konjunktiven Anfrage in die Vereinigung der konjunktiven Anfrage bzgl. der Sprache \mathcal{L}_{tripod} im allgemeinen Fall nicht möglich.

Beweis 1: Zunächst soll die Erweiterung der Sprache durch den Konstruktor $C_L \rightarrow \exists R.A$ betrachtet werden. Hierfür soll ein Beispiel angegeben werden, bei dem die Umschreibung einer konjunktiven Anfrage nicht möglich ist. Es seien in der globalen Ontologie (TBox \mathcal{T}) ein Konzept *Mensch* und eine Rolle *istElternTeil* vorhanden, die ein Kind und Elternteil in Beziehung setzt. Des Weiteren ist durch ein Axiom $\exists \text{istElternTeil.Mensch} \sqsubseteq \text{Mensch}$ ausgedrückt, dass das Kind eines Menschen selbst ein Mensch ist. Die lokalen Datenstrukturen (ABox $\mathcal{A}_{1\dots n}$) enthalten die Klasse Mensch_L und die Rolle istKind_L und Abbildungen folgender Form:

- $\text{istKind}(a,b)_L \subseteq \text{istKind}$
- $\text{Mensch}_L(a) \subset \text{Mensch}(a)$

Ferner sei eine Anfrage $Q(a) \leftarrow \text{Mensch}(a)$ gegeben. Dann ist für beliebige $n \in \mathbb{N}$ folgende Anfrage Q'_n eine Umformulierung der Anfrage Q :

$$Q'_n \leftarrow \text{istElternTeil}_L(a,b_1) \wedge \text{istElternTeil}_L(b_1,b_2) \wedge \dots \wedge \text{istElternTeil}_L(b_{n-1},b_n) \wedge \text{Mensch}_L(b_n)$$

Es ist offensichtlich, dass für beliebige $n \in \mathbb{N}$ eine Interpretation hergestellt werden kann, sodass die Vereinigung (Disjunktion) aller Q'_1, \dots, Q'_n keine maximale Umformulierung der Anfrage Q darstellt. Folglich kann eine maximale (und sichere) Umformulierung in Form einer Vereinigung von konjunktiven Anfragen nicht ausgedrückt werden. Hierzu sei angemerkt, dass eine maximale Umformulierung der betrachteten Anfrage durch eine rekursive Datalog-Anfrage sichergestellt werden kann:

$$Q'(a) \leftarrow \text{Mensch}_L(a) \quad Q'(a) \leftarrow \text{istElternTeil}(a,b) \wedge Q'(b)$$

Ein Gegenbeispiel des Falls $C \rightarrow \forall R.A$ erfolgt analog am Beispiel des Axioms: $\text{Mensch} \sqsubseteq \forall \text{istKind.Mensch} \square$

Lemma 2: Sei die Sprache \mathcal{L}_{tripod} um folgende Konstruktoren erweitert:

$$C_L \rightarrow \geq nR \quad \text{oder} \quad C \rightarrow \geq nR$$

Dann ist die Umschreibung einer konjunktiven Anfrage in die Vereinigung der konjunktiven Anfrage bzgl. der Sprache \mathcal{L}_{tripod} im allgemeinen Fall nicht möglich.

Beweis 2: Dieses Lemma wird analog zu der Arbeit [BLR97] bewiesen. Es besteht die folgende Problemstellung: Es sei in der globalen Ontologie (TBox \mathcal{T}) eine Rolle *istKind* definiert, die den Elternteil mit dem Kind in Beziehung setzt. In lokalen Datenstrukturen (ABox $\mathcal{A}_{1\dots n}$) ist eine Rolle R_1 und ein Konzept C_1 definiert. Ferner sind folgende Abbildungen beschrieben:

$$R_1(a, b) \subseteq \text{istKind}(a, b) \wedge \text{istKind}(a, z) \wedge (\leq 1 \text{ istKind})(z)$$

$$C_1(a) \subseteq (\geq 3 \text{ istKind})(a)$$

Die Benutzeranfrage besteht in der Ermittlung aller Eltern, die mindestens zwei Kinder haben:

$$Q(a) \leftarrow (\geq 2 \text{ istKind})(a)$$

Man kann feststellen, dass für beliebige $n \in \mathbb{N}$ folgende Anfrage Q'_n eine Umschreibung der Anfrage Q ist:

$$Q'_n(a) \leftarrow R_1(a, b_1) \wedge R_1(b_1, b_2) \wedge \dots \wedge R_1(b_n, w) \wedge C_1(w)$$

Um diesen Sachverhalt überprüfen zu können, wird zunächst die Variable b_n betrachtet. Laut der Definition für $R_1(b_n, w)$ hat einer der Kinder von b_n , in der Formel mit der Variablen z ausgezeichnet, höchstens ein eigenes Kind. Gemäß der Abbildung $R_1(b_n, w)$ referenzieren beide Variablen w und z die Kinder von b_n . Wenn aber diese Variablen ungleich sind, d. h. sie referenzieren unterschiedliche Elemente, dann hat b_n zwei Kinder. Gleichzeitig – laut $C_1(w)$ – verfügt w über mindestens drei Kinder und ist demzufolge ungleich z . Infolgedessen hat b_n zwei Kinder. Die Verwendung dieser Argumentation auf b_{n-1} führt zur Erkenntnis, dass b_{n-1} ebenfalls zwei Kinder hat usw. Letztendlich kann festgehalten werden, dass a ebenfalls zwei Kinder hat, womit $Q(a)$ wahr ist.

Es ist offensichtlich, dass für beliebige $n \in \mathbb{N}$ eine Interpretation hergestellt werden kann, sodass die Vereinigung (Disjunktion) aller Q'_1, \dots, Q'_n keine maximale Umformulierung der Anfrage Q darstellt. Folglich kann eine maximale (und sichere) Umformulierung in Form einer Vereinigung von konjunktiven Anfragen nicht ausgedrückt werden. Hierzu sei angemerkt, dass eine maximale Umformulierung der betrachteten Anfrage durch rekursive Datalog-Anfrage sicher-

gestellt werden kann:

$$Q'(a) \leftarrow C_1(a)$$

$$Q'(a) \leftarrow R_1(a,b) \wedge Q'(b)$$

Ein Gegenbeispiel für den Fall $C_L \rightarrow \leq nR, C \rightarrow \leq nR$ erfolgt analog. \square

Lemma 3: Sei die Sprache \mathcal{L}_{tripod} um folgende Konstruktoren erweitert:

$$C_L \rightarrow \neg A \mid \forall R.A \quad \text{oder} \quad C \rightarrow A_1 \sqcup A_2$$

Dann ist die Umschreibung einer konjunktiven Anfrage in die Vereinigung der konjunktiven Anfrage bzgl. der Sprache \mathcal{L}_{tripod} im allgemeinen Fall nicht möglich.

Beweis 3: Der Beweis dieses Lemmas folgt aus dem Satz 5 in der Arbeit [CGL⁺05]. Es wird gezeigt, dass die Berechnung einer Antwort auf eine konjunktive Anfrage für einfache Beschreibungslogiken, die in der Lemma angenommene Konstruktoren und Axiome der Konzeptinklusion aufweisen, grundsätzlich *co-NP*-schwer sind. \square

Lemma 4: Sei die Sprache \mathcal{L}_{tripod} um folgenden Konstruktor erweitert:

$$C \rightarrow \neg C$$

Dann ist die Umschreibung einer konjunktiven Anfrage in die Vereinigung der konjunktiven Anfrage bzgl. der Sprache \mathcal{L}_{tripod} im allgemeinen Fall nicht möglich.

Beweis 4: Der Beweis dieses Lemmas folgt teilweise aus dem Lemma 3, weil man aus der Negation des komplexen Konzepts relativ leicht eine Disjunktion zweier Konzepte auf der rechten Seite des Inklusionsaxioms erhalten kann: $B \sqsubseteq \neg(\neg A_1 \sqcap \neg A_2) \Leftrightarrow B \sqsubseteq A_1 \sqcup A_2$. Ferner führt die Negation eines komplexen Konzepts zu wesentlich größerem Rechenaufwand der Sprache als eine Disjunktion. So wurde in der Arbeit [DLNS94] gezeigt, dass die Überprüfung auf Konzeptzugehörigkeit für die Sprache \mathcal{ALC} ein *PSPACE*-vollständiges Problem ist. \square

Lemma 5: Sei die Sprache \mathcal{L}_{tripod} um folgende Konstruktoren erweitert:

$$C_L \rightarrow \neg \exists R.C \quad \text{oder} \quad C \rightarrow \neg \exists R.C$$

Dann ist die Umschreibung einer konjunktiven Anfrage in die Vereinigung der konjunktiven Anfrage bzgl. der Sprache \mathcal{L}_{tripod} im allgemeinen Fall nicht möglich.

Beweis 5: Der Beweis folgt aus der Äquivalenz entsprechender Konstruktoren (gemäß der De-

inition einer Interpretationsfunktion) $\neg \exists R.C$ und $\forall R.(\neg C)$ und der Lemmatas 1 und 3. \square

Lemma 6: Sei die Sprache \mathcal{L}_{tripod} um das Axiom der Rollentransitivität erweitert: $(Trans R)$. Dann ist die Umschreibung einer konjunktiven Anfrage in die Vereinigung der konjunktiven Anfrage bzgl. der Sprache \mathcal{L}_{tripod} im allgemeinen Fall nicht möglich.

Beweis 6: Der Beweis folgt aus der axiomatischen Semantik. Zur Umformulierung der Anfrage in relationale Algebra, die eine Anweisung in Form einer transitiven Rolle enthält, benötigt man zwingend eine Rekursion. \square

Lemma 7: Sei die Sprache \mathcal{L}_{tripod} um funktionale Rollen erweitert: $(Funct R)$ Dann ist die Umschreibung einer konjunktiven Anfrage in die Vereinigung der konjunktiven Anfrage bzgl. der Sprache \mathcal{L}_{tripod} im allgemeinen Fall nicht möglich.

Beweis 7: Der Beweis folgt aus dem Lemma 2 und dem Satz 14 in der Arbeit [CGLR08]. Die Semantik der funktionalen Rollen ist äquivalent zu $\top \sqsubseteq \leq 1 R$. Wir beschreiben ein solches Axiom in Form von $(Funct R)$, da der Konstruktor ≤ 1 im Wissensrepräsentationsformalismus \mathcal{L}_{tripod} nicht erlaubt ist. \square

Auf diese Weise zeigten wir, dass die Sprache \mathcal{L}_{tripod} maximal ist – und zwar insofern, als dass die Zulassung von Konstruktoren in anderer Form als dieser in der Syntax der Sprache \mathcal{L}_{tripod} definiert ist, im allgemeinen Fall zu keiner Umformulierung der konjunktiven Anfragen in relationale Algebra führt.

5.2.3. Normalisierung der TBox

Wie im Abschnitt 5.2.2 diskutiert, soll jede TBox \mathcal{T} in Normalform vorliegen. Tatsächlich erfordert die Anfrageumschreibung, dass die in der Sprache \mathcal{L}_{tripod} beschriebene TBox \mathcal{T} in die Normalform überführt werden soll. Um \mathcal{T} normalisieren zu können, kann es notwendig sein, einige *Hilfskonzepte* einzuführen. Ausgangspunkt soll folgendes Beispiel sein:

Beispiel

Eine TBox \mathcal{T} enthält folgendes Axiom:

$$\text{Ingenieur} \sqsubseteq \exists \text{entwickelt} . \exists \text{hatCADModell}$$

Das Axiom besagt, dass alle Ingenieure etwas entwickeln, das ein CAD-Modell hat. An dieser Stelle kann gezeigt werden, dass die normalisierte Version der TBox $\mathcal{T}_{\mathcal{N}(\mathcal{F})}$ folgende Axiome enthält:

$$\text{Ingenieur} \sqsubseteq \exists \text{entwickelt} . A_1$$

$$A_1 \sqsubseteq \exists \text{hatCADModell}$$

Es kann festgestellt werden, dass ein Hilfskonzept A_1 während des Normalisierungsprozesses hinzugefügt worden ist.

Während der Normalisierung der TBox können Prädikate, die den Hilfskonzepten entsprechen, eingeführt werden und damit bei der Umformulierung auftreten. Ohne Beschränkung der Allgemeinheit können wir annehmen, dass für jede ABox die extensionale Repräsentation eines solchen Hilfsprädikats leer ist. Folglich können wir jede Klausel mit Hilfsprädikaten bei der Umformulierung eliminieren.

5.3. Aufbau eines Integrationssystems

Dieser Abschnitt stellt zunächst den konzeptionellen Entwurf eines Mapping-Verfahrens dar, das als Bindeglied zwischen Elementen der globalen Ontologie und Elementen der Schemata der Datenquellen dient. Hierzu müssen wir den Begriff der *Anfragekorrespondenz* (engl. *mappings*) einführen. Eine Korrespondenz ist eine deklarative Beschreibung, die vom Integrationssystem benutzt wird, um eine globale Anfrage in eine oder mehrere lokale Anfragen zu übersetzen.

Durch Korrespondenzen (oder Mapping-Beziehung) wird semantische Heterogenität zwischen den Schemata überwunden, denn auf diese Weise werden thematisch und semantisch korrespondierende Elemente miteinander gekoppelt. Der Vorgang, um solche Mapping-Beziehungen zu definieren, ist der Mapping-Prozess. Der Mapping-Prozess wird durch eine Mapping-Gruppe von Experten durchgeführt – denn Mapping-Spezifikationen in Form heutiger Produktdatenmodelle sowie Terminologie haben mittlerweile einen Umfang erreicht, der durch eine einzelne Person nicht bearbeitet werden kann. Darüber hinaus sind die Sichten der in einem produzierenden Unternehmen vertretenen Systeme kontextspezifisch. Daher ist die Mitarbeit von denjenigen Vertretern, die über detaillierte Kenntnisse der konkreten Datenquellen und ihnen zugrundeliegende Schemata verfügen unabdingbar, um beispielsweise die Intention eines Begriffs aus der Anwendung heraus besser zu erklären.

Ein weiterer Schritt in Richtung Aufbau eines semantischen Integrationssystem ist das mathematische Modell des Ansatzes. Hierin werden die im Ansatz verwendeten Begriffe eingeführt. Dies ist wichtig, um die Entwurfsentscheidungen und die Architektur des zu entwickelnden semantischen Informationsintegrationssystem auf eine definierte und nachvollziehbare Basis zu stellen. Als Grundlage für das Modell dienen die Arbeiten von CALVANESE [CDGLR11, CDGL⁺09], HORROCKS [PUMH10, JRCHB11], LUTZ [LM07, GLHS08], CHOMICKI [Cho07, Cho08] und des W3C [W3C].

5.3.1. Verfahren zum Mapping

Der Mapping-Prozess zwischen globaler Ontologie und heterogenen, verteilten Datenquellen ist von iterativer Natur und zielt auf die Bestimmung von Anfragekorrespondenzen (oder Abbildungsvorschriften) zwischen semantisch gleichwertigen Begrifflichkeiten je nach Granulierung auf Schema-, Entitäts- oder Attributebene. Die Abbildung 5.7 zeigt schematisch den Mapping-Prozess bei dem manuell oder (halb-)automatisch definierte Korrespondenzen als logische Abbildung von der Quelle zum Zielmodell durch das Runtime-System interpretiert werden.

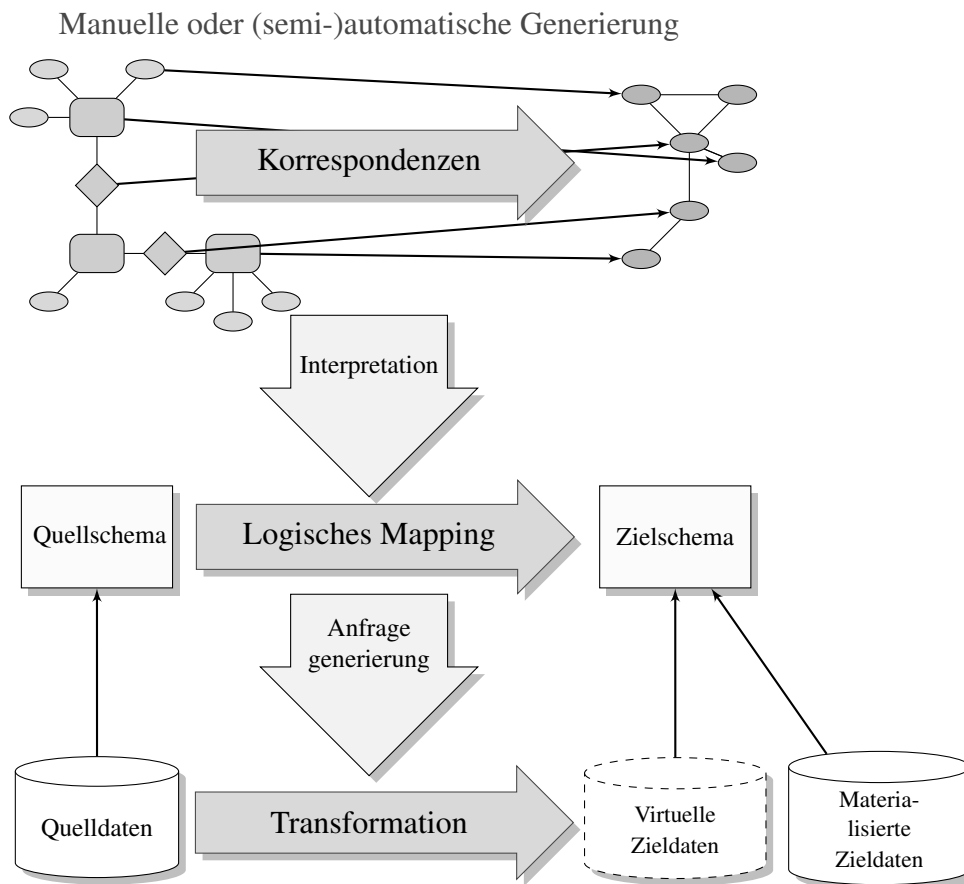


Abbildung 5.7.: Mapping-Prozess im Überblick in Anlehnung an LESER UND NAUMANN [LN07]

Sobald eine Anfrage an das Zielmodell – in diesem Fall ein virtuelles globales Ontologiemodell – aufgestellt ist, verwendet das Integrationssystem die logische Zuordnung, um lokale Anfragen zu generieren und damit alle relevanten Daten aus den Quellen entsprechend zu erheben.

Der gesamte Mappingprozess wird in vier Teilprozesse untergliedert und ist wie folgt beschrieben:

1. Vorbereitungsschritt

Aufstellung von Anforderungen und des Hauptleitbildes des Mapping-Prozesses. Die zu mappenden Datenquellen werden in Bezug auf ihre Spezifikation verifiziert. Der Aufbau von Datenquellenstrukturen (Schema, Art der Beziehungen etc.), die Eigenschaften, die der Korrespondenzbildung zugrunde liegen und die angewandten Begrifflichkeiten werden erläutert. Ergebnis dieses Schritts ist die Auswahl von Personen, die der Mapping-Gruppe hinzugefügt werden sowie die Zusammenstellung aller für den Mapping-Vorgang relevanter Unterlagen, z. B. Spezifikationen über zu integrierende Datenquellen, Produktdatenmodelle, systemspezifische Sach- und/oder Dienstleistungsklassifikationsstrukturen, unternehmensspezifische Templates und Bibliotheken, notwendige Standards und Normen.

2. Semantische Analyse der Datenquellen

Im zweiten Schritt führt die Mapping-Gruppe eine Analyse von Architekturen und Anwendungskontexten der Datenquellen durch. Hierbei wird der Fokus auf die Untersuchung syntaktischer, semantischer Gemeinsamkeiten und Unterschiede sowie thematischer Gültigkeitsbereiche gelegt. Das Resultat dieses Schritts ist die Aggregation von thematisch und semantisch korrelierenden Anwendungsbereichen mehrerer Datenquellen. Ferner wird die Detaillierungsstufe des Mapping-Prozesses festgelegt, d. h. ob das Mapping auf Entitäts-, auf Attributebene oder ggf. auf einer Mischform aus beiden erfolgen soll.

Ein weiteres Resultat semantischer Analyse besteht darin, dass die Art der Mapping-Beziehungen zwischen zusammengehörenden Bereichen festgelegt wird. Das heißt, ob uni- oder bidirektionale Korrespondenzen spezifiziert und realisiert werden. Aus der Sicht des Mapping-Prozesses zwischen globaler Ontologie und lokaler Schemata der Datenquellen werden bidirektionale Korrespondenzen bevorzugt. An dieser Stelle werden semantisch korrespondierende Bereiche mehrerer Datenquellen – im Gegensatz zum Mapping-Vorgang bei Produktmodellen – nicht priorisiert behandelt, sondern ausgehend von den Ergebnissen dieses Schritts als gleichwertig betrachtet.

3. Anfragekorrespondenzen

Die Mapping-Ansätze zur Informationsintegration sind zunächst charakterisiert durch ein globales Schema und eine Menge strukturell heterogener Datenquellen. Es sind hauptsächlich

zwei Ansätze zur Vermittlung zwischen den verschiedenen lokalen Schemata und dem globalen Schema des Integrationssystems: Global-as-View (GAV) und Local-as View (LAV), die im Abschnitt 2.2.2 erläutert wurden. Darauf aufbauend soll im Folgenden der hybride Ansatz Global-Local-as-View (GLaV) zur semantischen Integration für die Produktentwicklung vorgestellt werden.

Global-Local-as-View

Der GLaV-Ansatz kombiniert die Vorteile beider Basisansätze, indem er die Erhaltung von neu hinzugefügten Quellen sowie die Unterstützung von komplexen Anfragetransformation und die Unabhängigkeit des globalen Schemas gewährleistet.

In einem Informationsintegrationssystem $IIS = \langle G, S, M \rangle$ basierend auf GLaV-Ansatz verbinden die Korrespondenzen M (Mappingbeziehungen) jede Anfrage Q_S eines Quellschemas S mit einer Anfrage Q_G über das globale Schema G . Infolgedessen stellen LaV-Korrespondenzen eine zusammengesetzte Menge aus Abbildungsvorschriften folgender Form dar: $Q_S \rightsquigarrow Q_G$. Zum besseren Verständnis soll im Folgenden eine globale Ontologie O_G und relationale Schemata der Datenquellen betrachtet werden.

Die Quellen s_1 und s_2 enthalten die Auflistung von Autos und Motorrädern mit dazugehöriger Preiskategorie und in diesem Zusammenhang stehendem Preisbereich. Dann können GLaV-Korrespondenzen durch folgende Abbildungsvorschriften zusammengestellt werden:

Vor diesem Hintergrund ergeben sich folgende Möglichkeiten für den Typ einer Korrespondenz zu jeder Anfrage Q_S von M . Formal wird definiert:

$$Typ : G \rightarrow \{\text{korrekt, vollständig, äquivalent}\}$$

Mit Typ kann angegeben werden, wie genau die Anfrage Q_S der Quelle S mit der entsprechenden Anfrage Q_G in Beziehung steht. Hierzu werden die verschiedenen Typen wie folgt spezifiziert:

- Wenn $Typ(Q_S) = \text{korrekt}$, dann bedeutet das, dass die Extension von Q_S eine Untermenge von Tupeln ist, die Q_G erfüllen. Somit ist die Extension von Q_S in der Extension von Q_G enthalten und das durch Q_S symbolisierte Konzept ist folglich spezifischer als das von Q_G . Mit anderen Worten: Die Extension von Q_S enthält *nur* korrekte Antworten auf die Anfrage Q_G , jedoch nicht notwendigerweise alle Antworten. In diesem Fall gilt, dass das globale Schema O_G ein M -Modell in Bezug auf die Daten der Quelle D ist, wenn für jedes Element $Q_S \in M$ gilt: $Q_S^D \subseteq Q_G^{O_G}$. Das bedeutet also: $\forall \vec{x} (Q_S(\vec{x}) \rightarrow Q_G(\vec{x}))$.
- Wenn $Typ(Q_S) = \text{vollständig}$, dann bedeutet das, dass die Extension von Q_S eine Obermenge von Tupeln ist, die Q_G erfüllen. Somit enthält die Extension von Q_S die Extension

von Q_G ; das durch Q_S symbolisierte Konzept ist folglich genereller als das von Q_G . Mit anderen Worten: Die Extension von Q_S enthält *alle* korrekten Antworten auf die Anfrage Q_G . Jedoch können darunter auch inkorrekte Antworten oder gar das sogenannte Rauschen vorkommen. In diesem Fall gilt, dass das globale Schema Q_G ein \mathcal{M} -Modell in Bezug auf die Daten der Quelle D ist, wenn für jedes Element $Q_S \in M$ gilt: $Q_S^D \supseteq Q_G^{O_G}$. Das bedeutet also: $\forall \vec{x}(Q_G(\vec{x}) \rightarrow Q_S(\vec{x}))$.

- Wenn $Typ(Q_S) = \text{äquivalent}$, dann bedeutet das, dass die Extension von Q_S eine exakte Menge von Tupeln ist, die O_G erfüllen. Somit ist die Extension von Q_S identisch mit der Extension von Q_G , und das durch Q_S symbolisierte Konzept stimmt mit dem von O_G überein. Mit anderen Worten: Die Extension von Q_S enthält *alle* und *nur* korrekte Antworten auf die Anfrage Q_G . In diesem Fall gilt, dass das globale Schema Q_G ein \mathcal{M} -Modell in Bezug auf die Daten der Quelle D ist, wenn für jedes Element $Q_S \in M$ gilt: $Q_S^D \equiv Q_G^{O_G}$. Das bedeutet also: $\forall \vec{x}(Q_S(\vec{x}) \leftrightarrow Q_G(\vec{x}))$.

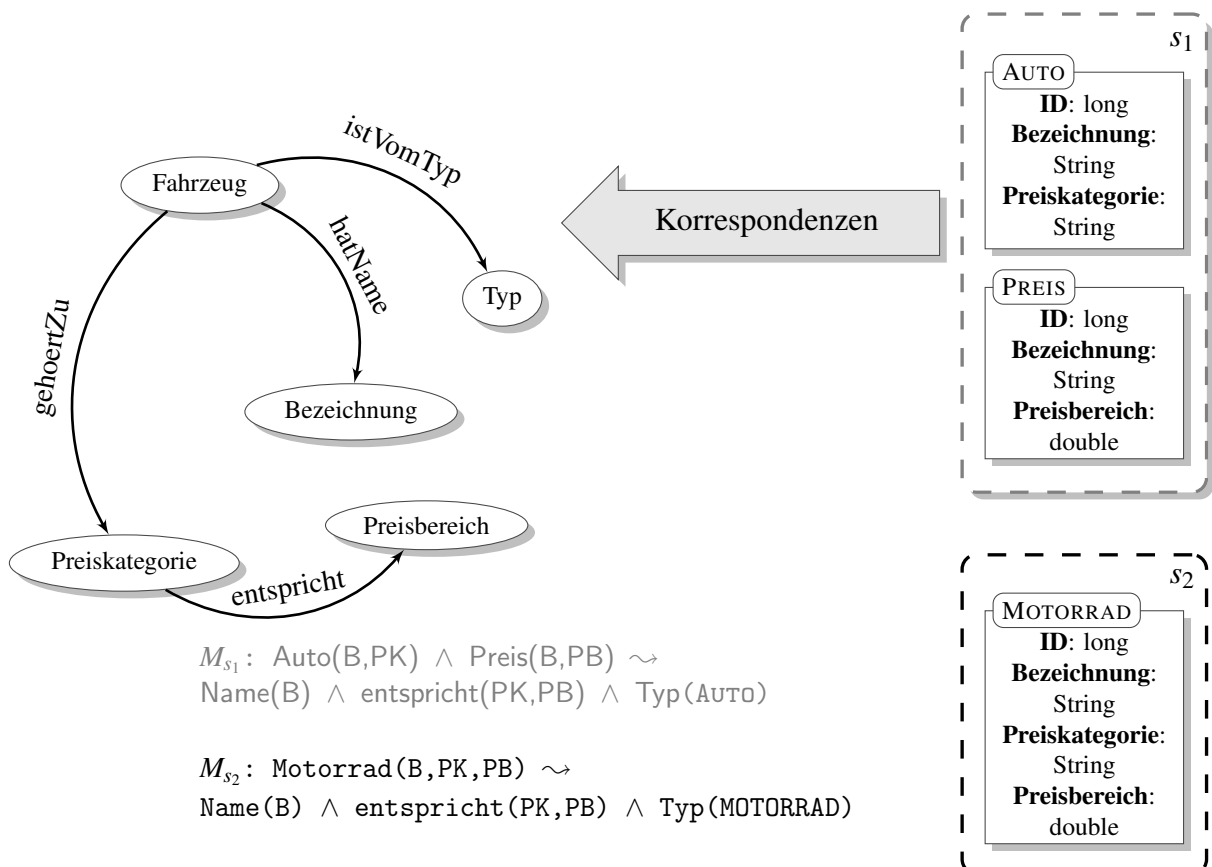


Abbildung 5.8.: Beispiel für ein GLaV-Mappingprozess

Mit der Mappingfunktionalität soll sichergestellt werden, dass Endanwender ständig ihre ihnen vertrauten Begrifflichkeiten verwenden können, aber dennoch Zugriff auf relevante Infor-

mationen haben, die auf Basis unterschiedlicher Datenquellen bereitgestellt werden. Prinzipiell hat ANDREA CALÌ ET AL. gezeigt, dass jedes GLaV-System in ein GaV-System transformiert werden kann [CCGL02]. Hierzu besteht die Idee darin, dass eine GLaV-Korrespondenz in eine GaV-Korrespondenz mit zusätzlicher Inklusionsabhängigkeit in G überführt werden kann. Das bedeutet: Um ein GLaV-System in GaV transformieren zu können, wird jeder GLaV-Korrespondenz der Form $Q_S \rightsquigarrow Q_G$ ein neues Element r in globales Schema G eingefügt. Darüber hinaus wird r mit der GaV-Korrespondenz $r \rightsquigarrow Q_S$ und der Inklusionsabhängigkeit $r \sqsubseteq Q_G$ in Verbindung gesetzt. Gemäß dem Grundprinzip der Transformation eines GLaV-Systems in ein GaV-System werden Mappings mittels einer Menge von Abbildungsvorschriften der folgenden Form definiert:

$$G(w) \mapsto D(v), \text{ wobei}$$

- $G(w)$ der sogenannte Kopf der Mappingbeziehung ist, wobei G ein Konzept oder eine Rolle sein kann, die in der TBox der globalen Ontologie vorkommen kann, und w eine Reihe von Termen repräsentiert;
- $D(v)$ der sogenannte Körper der Mappingbeziehung ist, wobei D eine in Prädikatenlogik erster Stufe ausgedrückte Anfrage mit der Stelligkeit $n > 0$ ist. Dabei repräsentiert v eine Menge von gebundenen Variablen über den Quellschemata S . Diese Art der Anfragen werden hierbei in Form von SQL-Anfragen ausgedrückt.

Des Weiteren werden in diesem Zusammenhang zwei Arten von direkten Mappingbeziehungen definiert:

- *Konzeptkorrespondenz*, in der der Kopf ein unäres Atom der Form: $K(f(w))$ ist. Hier ist K ein Konzept und f ein Funktionssymbol der Stelligkeit n .
- *Rollenkorrespondenz*, in der der Kopf ein binäres Atom der Form $R(f_1(v'), f_2(v''))$ ist. Hierzu sind R eine Rolle, f_1, f_2 Funktionssymbole der Stelligkeit $n_1, n_2 > 0$ und v', v'' Variablen, die in v vorkommen.

In Abbildung 5.9 sind noch einmal direkte Mappingbeziehungen aufgeführt. Auf diese Weise kann das Integrationssystem vorhandene Indexstrukturen und Eigenschaften der ihnen zugrunde liegenden Informationssysteme aus der Produktentwicklung besser nutzen.

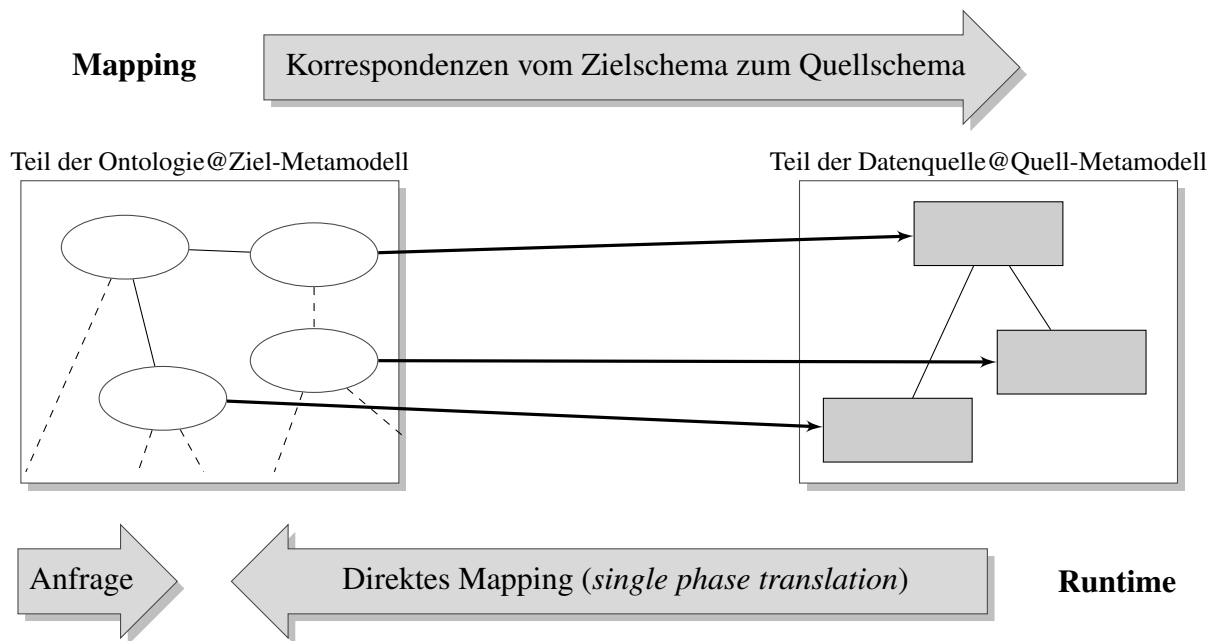


Abbildung 5.9.: Direktes Mapping basierend auf G(L)aV-Ansatz nach [Biz03], [LN07], [Lan09]

4. Mapping-Prozessbegleitende Dokumentation

Die Ergebnisse vorangegangener Teilaktivitäten des Mapping-Prozesses müssen durchgängig dokumentiert werden. Dies liegt im Wesentlichen darin begründet, dass einerseits eine begleitend erstellte Dokumentation im weiteren Ablauf als Umsetzungsgrundlage verwendet wird, andererseits bekommen dadurch auch die Beteiligten die Übersicht von einzelnen Aktivitäten vermittelt, die der Mapping-Gruppe nicht direkt zugeordnet waren.

Mit dem entwickelten Verfahren zum Mapping ist die fundamentale Voraussetzung für ein mathematisches Modell zur semantischen Informationsintegration für die Produktentwicklung geschaffen, auf das im Folgenden eingegangen wird.

5.3.2. Mathematisches Modell zur semantischen Informationsintegration

Grundlage für eine formale Spezifikation der in den vorangegangenen Abschnitten erarbeiteten Teilkonzepte ist das mathematische Modell. Das Modell erlaubt eine systematische und strukturierte Beschreibung von Entitäten der realen Welt. STACHOWIAK kennzeichnet Modelle durch drei Hauptmerkmale [Sta73]:

- *Abbildungsmerkmal*: Modelle sind grundsätzlich Modelle von etwas, nämlich Abbildungen, Repräsentationen natürlicher oder künstlicher Objekte, die selbst wieder Modelle sein können.
- *Verkürzungsmerkmal*: Modelle enthalten in der Regel nicht alle Attribute des durch sie repräsentierten Originals, sondern nur solche, die den jeweiligen Modellerschaffern und/oder Modellbenutzern relevant erscheinen.
- *Pragmatisches Merkmal*: Modelle sind ihren Originalen nicht per se eindeutig zugeordnet. Sie erfüllen ihre Ersetzungsfunktion (1) für bestimmte – erkennende und/oder handelnde, modellbenutzende – Subjekte, (2) innerhalb bestimmter Zeitintervalle und (3) unter Einschränkung auf bestimmte gedankliche oder tatsächliche Operationen.

Infolgedessen kann grundsätzlich jede berechenbare Menge mathematischer Vorschriften, Gleichungen etc. als ein Modell bezeichnet werden, die einen Aspekt eines realen Anwendungsbereichs zum Ausdruck bringt. Der Ausgangspunkt dafür liegt in der Erstellung eines formalen Modells, in dem die als relevant bewerteten Bestandteile und ihre Beziehungen untereinander explizit beschrieben werden. Hierfür stellt das mathematische Modell ein allgemeingültiges, methodisches und begriffliches Rahmenkonzept dar, um langfristig die Beschreibung und Realisierung eines semantischen Integrationssystems für die Produktentwicklung zu ermöglichen.

Definition 26: *Es seien m Datenquellen $\Delta_1, \dots, \Delta_m$ gegeben. Jede Quelle ist durch ein relationales Schema S_{Δ_i} über eine abstrakte Beschreibungssprache \mathcal{L}_{Δ} repräsentiert. Ein so beschriebenes Schema drückt eine mögliche Datenquelle als Instanz sowie die Möglichkeit, auf dieser Instanz zu operieren, aus. Ferner ist ein ontologiebasiertes Informationsintegrationssystem folgendermaßen definiert: $\Phi \stackrel{\text{def}}{=} \{O_G, \{S_{\Delta_i}\}_{i=1..m}, M\}$, wobei*

- $O_G = \{\mathcal{T}_G, \mathcal{A}_G\}$ eine globale Ontologie über dem Alphabet N_G durch den Wissensrepräsentationsformalismus $\mathcal{L}_{\text{tripod}}$ dargestellt ist. Es wird vorausgesetzt, dass die Ontologie O_G konsistent ist, d. h. $\mathcal{M}(O_G) \neq \emptyset$. Ohne Beschränkung der Allgemeinheit wird die Annahme getroffen, dass die Faktenmenge (ABox) in der globalen Ontologie O_G leer ist⁹ ($\mathcal{A}_G = \emptyset$)

⁹Das System $\Phi \stackrel{\text{def}}{=} \{O_G, \{S_{\Delta_i}\}_{i=1..m}, M\}$ mit $\mathcal{A}_G \neq \emptyset$ wird zu einem System mit zusätzlicher Quelle reduziert $\Phi' = \{\{\mathcal{T}_G, \emptyset\}, O_G \cup \{S_{\Delta_i}\}_{i=1..m}, M \cup M_g\}$, wobei M_g eine Eins-zu-eins-Abbildung aller Konzepte der Ontologie O_G ist.

- $\{S_{\Delta_i}\}_{i=1,\dots,m}$ eine endliche Menge von Schemata der Datenquellen Δ_i ist. Hierzu ist jedes Quellschema über dem Alphabet N_{Δ_i} durch ein Fragment der Sprache Prädikatenlogik erster Stufe \mathcal{L}_{Δ} repräsentiert. Es wird vorausgesetzt, dass alle Quellschemata S_{Δ_i} konsistent sind ($\mathcal{M}(S_{\Delta_i}) \neq \emptyset$). Es wird ohne Beschränkung der Allgemeinheit angenommen, dass alle Alphabete paarweise disjunkt¹⁰ sind und sich von dem Alphabet der globalen Ontologie N_G unterscheiden
- M ist eine endliche Menge von Korrespondenzen (engl. mappings) zwischen globaler Ontologie O_G und der Quellschemata $\{S_{\Delta_i}\}_{i=1,\dots,m}$.

Definition 27: Eine Zuordnungsvorschrift der Form

$$Q_{\Delta} \rightsquigarrow Q_G$$

wird als Korrespondenz bezeichnet, wobei Q_G und Q_{Δ} vergleichbare Anfragen über dem Alphabet N_G und $\bigcup_{i=1,\dots,m} N_{\Delta_i}$ sind. Die Anfragen sind durch die Sprachen $Q_G \in \mathcal{QL}_{MG}$ und $Q_{\Delta} \in \mathcal{QL}_{M\Delta}$ beschrieben. Das Zeichen \rightsquigarrow steht dabei für eine der folgenden Beziehungen $\{\subseteq, \supseteq, \equiv\}$. Somit wird die Abbildung der Form $Q_{\Delta} \subseteq Q_G$ als korrekt, der Form $Q_{\Delta} \supseteq Q_G$ als vollständig und der Form $Q_{\Delta} \equiv Q_G$ als äquivalent bezeichnet.

Die Wahl der Anfragesprachen \mathcal{QL}_{MG} und $\mathcal{QL}_{M\Delta}$ steht in direktem Zusammenhang mit der Berechenbarkeit der Antwortmenge auf die Anfrage an das IIS. Zu diesem Zweck wird eine informelle Restriktion an die Anfragesprache $\mathcal{QL}_{M\Delta}$ aufgestellt. Sie soll nämlich über die Eigenschaft verfügen, effizient die Antworten auf Anfragen bezüglich der Menge verteilter Datenquellen $\{\Delta_i\}_{i=1,\dots,m}$ zu berechnen. Hierbei muss die Anfrage in einzelne Anfragefragmente aufgeteilt und an entsprechende Quellen delegiert werden. Die deklarative Sprache *Structured Query Language* (SQL) erfüllt diese Anforderung, da ihre theoretische Grundlage die relationale Algebra sowie den Relationenkalkül bilden und SQL von praktisch allen relationalen Datenbanksystemen zur Verfügung gestellt wird. Die in der vorliegenden Arbeit betrachtete Klasse von Korrespondenzen, für die ein effektiver Algorithmus zur Berechnung von Antworten vorgestellt wird, ist die Klasse korrekter konjunktiver G(L)aV-Korrespondenzen: $Q_{\Delta} \subseteq Q_G$.

Die Semantik eines ontologiebasierten Informationsintegrationssystems für die Produktentwicklung wird mittels einer Interpretationsvorschrift beschrieben. Es wird festgelegt, dass eine globale Interpretation I_G korrekt bezüglich Korrespondenzen M ist, wenn I_G alle Korrespondenzen in M erfüllt. Die Menge aller globalen Interpretationen, die hinsichtlich M korrekt sind, wird mit $\mathcal{M}(M)$ bezeichnet.

¹⁰Die Disjunktheit kann offensichtlich durch das Hinzufügen einer Datenquellenummer zu jedem Symbol dieser Datenquelle erreicht werden.

Definition 28: Als ein globales Modell von IIS $\Phi = \{O_G, \{S_{\Delta_i}\}_{i=1\dots m}, M\}$ wird eine globale Interpretation I_G bezeichnet, die einerseits korrekt bezüglich M und andererseits ein Modell der globalen Ontologie O_G ist. Eine Menge globaler Modelle eines IIS Φ wird beschrieben mit $\mathcal{M}(\Phi)$:

$$I_G \in \mathcal{M}(\Phi) \Leftrightarrow I_G \in \mathcal{M}(M) \wedge I_G \in \mathcal{M}(O_G)$$

Definition 29: Das Integrationssystem Φ ist konsistent, wenn es über globale Modelle verfügt oder formal ausgedrückt: $\mathcal{M}(\Phi) \neq \emptyset$.

Anfragen an das Integrationssystem Φ sind in einer Anfragesprache $Q_{\mathcal{L}_G}$ über dem Alphabet \mathbb{N}_G formuliert. Dabei liegt der Anfragesprache $Q_{\mathcal{L}_G}$ die Sprache der Beschreibungslogik, nämlich der Wissensrepräsentationsformalismus \mathcal{L}_{tripod} zugrunde.

Definition 30: Es sei eine konjunktive Anfrage Q in der Sprache \mathcal{L}_{tripod} und ein Informationsintegrationssystem Φ gegeben. Dann ist die Menge semantisch sicherer Antworten auf die Anfrage Q durch eine Menge von Tupeln definiert. Dabei besteht die Tupelmenge aus den Konstanten \vec{t} , die die Antwortmenge auf die Anfrage Q bezüglich eines beliebigen globalen Modells I_G für Φ erfüllen:

$$ans(Q, \Phi) \stackrel{def}{=} \{ \vec{t} \mid I_G \models Q(\vec{t}), \forall I_G \in \mathcal{M}(\Phi) \}$$

Eine Antwort ist eine logische Schlussfolgerung, wenn sie einer beliebigen Interpretation angehört. Dabei ist jede Interpretation zum einen mit Korrespondenzen zwischen den Datenquellen und globaler Ontologie und zum anderen mit den Axiomen globaler Ontologie kompatibel. Grundsätzlich bedeutet das, dass solche Antworten aus Fakten lokaler Datenquellen, aus Korrespondenzen und aus Aussagen globaler Ontologie logisch hergeleitet werden können.

Lemma 8: Eine logisch hergeleitete Antwortmenge auf die Anfrage Q in Bezug auf ein Informationsintegrationssystem Φ ist die Schnittmenge der Antworten auf die Anfrage Q bezüglich aller globalen Modelle von Φ :

$$ans(Q, \Phi) = \bigcap_{I_G \in \mathcal{M}(\Phi)} Q(I_G)$$

Beweis 8: Der Beweis geht unmittelbar aus der Definition 30 hervor. \square

Bei der Antwortberechnung sollen die Axiome globaler Ontologie mitberücksichtigt werden; sie können im gewissen Sinne analog zu den Integritätsbedingungen einer relationalen Datenbank betrachtet werden. Gemäß der Definition 30 bedarf es einer Berechnung von Antworten auf die Anfrage hinsichtlich der Modelle der globalen Ontologie, also globaler Interpretationen,

die allen Axiomen der Terminologie \mathcal{T}_G entsprechen. Ferner können die zugrunde liegenden Datenquellen unvollständig sein, sodass die Durchführung der logischen Inferenz erforderlich ist, um implizite Fakten basierend auf den Axiomen der TBox \mathcal{T}_G und der konjunktiven Anfrage Q zu ermitteln.

5.4. Anwendung eines Integrationssystems

Neben der Ermittlung integrationsrelevanter Konzepte und Rollen, der Definition von Korrespondenzen und deren Zuordnung zu Datenquellenschemata ist die Anfragebeantwortung, d. h. die automatische Erhebung der relevanten Informationen aus heterogener Systemlandschaft eine der Hauptaufgaben des Systems zur semantischen Informationsintegration für die Produktentwicklung. In diesem Abschnitt wird der konzeptionelle Ablauf der auf Beschreibungslogik basierenden Integration von in heterogenen Datenquellen gespeicherten Produktdaten dargestellt. Hierzu sind vom System die folgenden Einzelschritte durchzuführen, die anschließend detailliert erläutert werden:

1. Umschreiben einer Anfrage
2. Analyse des sicheren Umschreibens
3. Antwortberechnung auf Anfragen über \mathcal{L}_{tripod}

Bevor die einzelnen Schritte zur Integration durchgeführt werden können, hat eine Auswahl des Integrationssystems bzw. der Wissensrepräsentation zu erfolgen, gemäß derer die Integration durchgeführt werden soll. Darüber hinaus muss festgelegt werden, welche der angebotenen Datenquellen integriert werden sollen, wobei auch innerhalb einer Datenquelle der zu integrierende Datenbestand eingeschränkt werden kann.

5.4.1. Umschreiben einer Anfrage bezüglich des Informationsintegrationssystems

In vielen Fällen ist die Berechnung der sicheren Antworten mit Hinzunahme der logischen Inferenz unmittelbar über die Faktenmenge der Datenquellen nicht möglich. Grund hierfür ist zum einen die hohe Netzwerkauslastung, zum anderen die niedrige Performanz, wenn der Datenbestand in den Quellen sehr groß ist. Dieser Sachverhalt führt dazu, dass eine „direkte“ Antwortberechnung bezüglich eines Integrationssystems für die Praxis nicht brauchbar ist. In diesem Abschnitt steht die Frage im Mittelpunkt, wie eine effektive Methode zur Berechnung von Antworten in einem ontologiebasierten Integrationssystem in der Praxis umgesetzt werden kann. Hierbei liegt der Fokus auf der Integration von Datenquellen mit sehr großen Datenbeständen, die vorwiegend in relationalen Datenbanken, also der am weitesten verbreiteten Datenverwaltungsform in der Produktentwicklung, vorliegen.

Die Problemstellung der Antwortberechnung wird in zwei Phasen aufgeteilt:

1. In der ersten Phase – sie basiert auf der Wissensbasis eines Informationsintegrationssystems IIS – erfolgt das Umschreiben der Benutzeranfrage Q , die durch Begriffe der

globalen Ontologie erfasst ist, in die Anfrage Q' . Die umgeschriebene Anfrage Q' ist so aufgebaut, dass für eine beliebige ABox \mathcal{A} die Gleichung

$$\text{ans}(Q, \mathcal{WB} = \{TBox, ABox\}) = \text{ans}(Q', \mathcal{WB} = \{\emptyset, ABox\})$$

gilt und Q' effektiv durch das IIS verarbeitet werden kann. Als „effektiv“ wird die Möglichkeit verstanden, die Anfrage so zu zerteilen, dass resultierende einzelne, isolierte Teilanfragen von entsprechenden Adoptoren (engl. *wrapper*) der jeweiligen Datenquellen ausgeführt werden können. Daraufaufgehend liefern die Adoptoren ermittelte Einzelantworten zurück, die durch das IIS (Mediatorsystem) zusammengeführt werden.

2. In der zweiten Phase wird die erhaltene Anfrage Q' ausgeführt. Hierzu wird die umgeschriebene Anfrage Q' , die als Ergebnis aus der ersten Phase hervorgeht, als *logischer Anfrageplan* zur Ausführung der Anfrage Q bezeichnet. Basierend darauf und auf den Korrespondenzen M wird der *physische Anfrageplan* zur Ausführung der Anfrage aufgestellt. Der physische Anfrageplan legt die Reihenfolge der Operationen (Anweisungen) zur Erhebung der Daten aus den Quellen und deren Verarbeitung fest.

Ausgehend von der ontologiebasierten Informationsintegration werden formale Definitionen zum Umschreiben von Anfragen eingeführt.

Definition 31: Die Anfrage Q' ist ein perfektes Umschreiben der Anfrage Q basierend auf der TBox \mathcal{T} einer Wissensbasis \mathcal{WB} des Integrationssystems Φ , wenn:

- alle in der Anfrage Q' vorkommenden Konzepte und Rollen sich vollständig auf die Begrifflichkeiten lokaler Schemata von Quellen $\{S_{\Delta_i}\}_{i=1\dots m}$ des Integrationssystems Φ referenzieren lassen
- die Antwortmenge auf die umgeschriebene Anfrage Q' bezüglich aller Quellschemata $\{S_{\Delta_i}\}$ mit der Menge von Antworten auf die ursprüngliche Anfrage Q in Bezug auf das Informationsintegrationssystem übereinstimmt $\Phi : Q'(S_{\Delta}) = Q(\Phi)$.

Das bedeutet also, dass ein perfektes Umschreiben der Anfrage alle sicheren (und nur diese) Antworten auf die Anfrage unter Berücksichtigung aller Definitionen des Integrationssystems Φ berechnet. Folglich, kann so ein Umschreiben als das beste unter den möglichen betrachtet werden.

In diesem Zusammenhang spielt die Anfragesprache, die zum Umschreiben herangezogen wird, eine wesentliche Rolle. Offensichtlich wirkt sich die Wahl der Sprache unmittelbar auf das Umschreiben der Anfrage aus. So kann beispielsweise für eine eingeschränkte Sprache kein Um-

schreiben geben, sogleich aber für eine ausdrucksstärkere Sprache existieren. Mit Ausdrucksstärke der Sprache hängt die Fähigkeit zusammen, ob alle notwendigen Daten aus den Quellen zur Formulierung einer sicheren Antwort hinsichtlich des Integrationssystems verwendet werden können.

Um eine Menge sicherer Antworten mittels umgeschriebener Anfragen berechnen zu können, wird das Anfrageumschreiben unmittelbar mit einer logischen Inferenz versehen. Hierbei richtet sich die logische Inferenz an die Axiome globaler Ontologie und an die Korrespondenzen eines Integrationssystems. Infolgedessen ist ein Umschreiben der Anfrage nur dann sichergestellt, wenn eine gewisse Kongruenz zwischen der Ausdrucksstärke einer Umschreibsprache und der Ausdrucksmächtigkeit der beschreibungslogischen Sprache globaler Ontologie, Abbildungen und der ursprünglichen Anfragesprache besteht.

Ein perfektes Umschreiben der Anfrage ist prinzipiell immer möglich unter der Voraussetzung, dass:

- die Sprache einer umgeschriebenen Anfrage keine Restriktionen erleidet,
- die Berechnung der Antwort auf die ursprüngliche Anfrage hinsichtlich einer einzelnen Datenquelle für die verwendete Sprache der Beschreibungslogik grundsätzlich entscheidbar ist.

In diesem Kontext wird der Umschreibvorgang der ursprünglichen Anfrage Q betrachtet und geprüft, in welchen Fällen das Umschreiben sicher ist. Von Interesse sind hierbei Konstruktionen von denjenigen Umschreibungen, die:

- *korrekt* sind; es werden nur Antworten mit der Einbeziehung von TBox-Axiomen, die der Antwortmenge auf die ursprüngliche Anfrage angehören berechnet: $Q'(S_\Delta) \subseteq Q(\Phi)$ (inkorrekte Antworten werden nicht berechnet),
- mittels einer Anfragesprache ausgedrückt sind $Q_{LPR} : Q' \in Q_{LPR}$,
- maximal für (im gewissen Sinne) eine Klasse von Anfragen über die Sprache Q_{LPR} .

Hierzu werden notwendige formale Definitionen eingeführt.

Definition 32: *Es seien ein Integrationssystem $\Phi = \{O_G, \{S_{\Delta_i}\}_{i=1..m}, M\}$ und eine Anfrage Q' gegeben. Dabei ist die Anfrage Q' anhand einer Kombination von Schemaelementen repräsentiert, die im rechten Teil der Abbildung $Q_\Delta \rightsquigarrow Q_G$ in der Menge M (genau bis auf die Variablenumbenennung) vorkommen. Als „Auffalten“ (engl. *unfolding*) der Anfrage Q' basierend auf Korrespondenzen eines Integrationssystems Φ wird eine Anfrage der Form $(Q')^-$ bezeichnet.*

Die Anfrage $(Q')^-$ ergibt sich durch die Substituierung jedes Schemaelementes Q_G mit dem Schemaelement Q_Δ gemäß allen Abbildungen der Form $Q_\Delta \rightsquigarrow Q_G$ in der Menge M . Hierzu werden alle durch den Existenzquantor gebundenen Variablen in Q_G durch neue, noch nicht in der Anfrage $(Q')^-$ enthaltenen Variablen ersetzt.

Es wird angenommen, dass die Anfrage Q in der Sprache $Q_{\mathcal{L}_G}$ über die globale Ontologie O_G verfasst ist. Dann ist die Anfrage Q' eine *partielle Umschreibung* der Anfrage Q hinsichtlich des Integrationssystems Φ , wenn:

1. alle verwendeten Konzepte und Rollen in der Anfrage Q' Begrifflichkeiten der Quellschemata $\{S_{\Delta_i}\}$ eines Integrationssystems Φ sind;
2. die Auffaltung einer Anfrage Q' basierend auf den Korrespondenzen eines Integrationssystems Φ in der ursprünglichen Anfrage Q enthalten ist. Dabei werden terminologische Axiome einer globalen Ontologie \mathcal{T}_G mitberücksichtigt, also $(Q')^- \sqsubseteq_{\mathcal{T}_G} Q$.

Wird die Annahme getroffen, dass die Anfrage Q in der Sprache $Q_{\mathcal{L}_G}$ über die globale Ontologie O_G verfasst ist, dann ist die Anfrage Q' eine *maximale Umschreibung* der Anfrage Q in der Sprache $Q_{\mathcal{L}_{PR}}$ hinsichtlich des Integrationssystems Φ , wenn:

1. die Anfrage Q' eine Umschreibung der Anfrage Q in der Sprache $Q_{\mathcal{L}_{PR}}$ bezüglich des Integrationssystems Φ ist,
2. keine Umschreibung U von der Anfrage Q existiert, sodass $Q' \sqsubseteq_{\cup \Delta_i} U$, aber $Q' \not\equiv_{\cup \Delta_i} U$ gilt.

Informell liefert die maximale Umschreibung alle Antworten zurück, die mithilfe von Adaptoren aus den Datenquellen extrahiert werden können.

Des Weiteren sei die Anfrage Q in der Sprache $Q_{\mathcal{L}_G}$ über die globale Ontologie O_G verfasst ist. Dann ist die Anfrage Q' eine *äquivalente Umschreibung* der Anfrage Q in der Sprache $Q_{\mathcal{L}_{PR}}$ hinsichtlich des Integrationssystems Φ , wenn:

1. die Anfrage Q' eine Umschreibung der Anfrage Q in der Sprache $Q_{\mathcal{L}_{PR}}$ basierend auf einem Integrationssystem Φ ist,
2. die Auffaltung einer Anfrage Q' basierend auf Korrespondenzen eines Integrationssystems Φ der ursprünglichen Anfrage Q äquivalent ist. Dabei werden terminologische Axiome einer globalen Ontologie \mathcal{T}_G mitberücksichtigt, also $(Q')^- \equiv_{\mathcal{T}_G} Q$.

Nichtformal betrachtet, ist eine äquivalente Umschreibung einer ursprünglichen Anfrage logisch gleich. Es ist offensichtlich, dass eine äquivalente Umschreibung nicht existieren kann, wenn die Datenquellen über ungenügend Daten bzw. Informationen verfügen, um basierend auf den Korrespondenzen ein logisches Äquivalent aufstellen zu können.

Beispiel

Auf die Anfrage nach dem Kundenfeedback zu einem Sportwagen, kann aus den Datenquellen lediglich darauf zurückgeschlossen werden, dass es sich grundsätzlich um einen Kundenfeedback zu einem PKW handelt.

5.4.2. Algorithmus zum Umschreiben von Anfragen für ontologiebasierte Informationsintegration

Bevor der Algorithmus zum Umschreiben von Anfragen näher erläutert wird, wird zunächst die Problemstellung in Zusammenhang mit der Anfrageumschreibung hinsichtlich einer auf \mathcal{L}_{tripod} basierenden Ontologie betrachtet.

Satz 1: Sei das System zur ontologiebasierten Informationsintegration $\Phi = \{O_G, \{S_{\Delta_i}\}_{i=1\dots m}, M\}$ so definiert, dass:

- die globale Ontologie O_G in beschreibungslogischer Sprache $\mathcal{L}_G = \mathcal{L}_{tripod}$ beschrieben ist;
- die Schemata der Datenquellen $\{S_{\Delta_i}\}$ in der Sprache relationaler Algebra (also ein Fragment der Prädikatenlogik erster Stufe) ausgedrückt sind;
- Korrespondenzen in Form von $Q_{\Delta} \subseteq Q_G$ angegeben sind, wobei $Q_{\Delta}, Q_G \in CQ - \mathcal{L}_{tripod}$ konjunktive Anfragen über den Wissensrepräsentationsformalismus \mathcal{L}_{tripod} sind. Es wird also ein GLaV-System mit korrekten konjunktiven Korrespondenzen betrachtet;
- das Integrationssystem Φ konsistent ist, also $\mathcal{M}(\Phi) \neq \emptyset$.

□

Des Weiteren sei die Benutzeranfrage Q an das Integrationssystem Φ in Form von Vereinigung konjunktiver Anfragen $Q_{\mathcal{L}_G} = UCQ - \mathcal{L}_{tripod}$ anzugeben. Dann existiert ein korrektes Umschreiben der Anfrage Q basierend auf dem Integrationssystem Φ über die Sprache $Q_{\mathcal{L}_{PR}} = UCQ$ – also eine Anfrage, dargestellt durch die Vereinigung relationaler, konjunktiver Anfragen in der Terminologie der Datenquellen.

Aufbau eines Umschreibalgorithmus

In diesem Abschnitt werden die Grundsätze des vorgeschlagenen Algorithmus zur Konstruktion einer umgeschriebenen Anfrage für das im Satz 1 betrachtete Informationsintegrationssystem betrachtet.

Der Kerngedanke des Algorithmus ist die Aufteilung des Umschreibprozess einer Anfrage in

einzelne Schritte, nämlich in Bezug auf die Axiome globaler Ontologie und bezüglich aufgestellter Korrespondenzen. Für den vorgeschlagenen Algorithmus müssen a priori folgende Bedingungen erfüllt sein:

1. Die globale Ontologie über die Sprache \mathcal{L}_{tripod} soll in normalisierter Form vorliegen.
2. Die Konsistenzprüfung des Integrationssystems Φ soll durchgeführt sein.

Die zweite Vorbedingung wird nur deshalb eingeführt, weil die Korrektheit des vorgeschlagenen Algorithmus nur dann sichergestellt ist, wenn das System Φ konsistent ist. Das bedeutet, dass die Ontologie des Integrationssystem über globale Modelle verfügt. Im Fall, dass die Daten in den Quellen in gewissem Maße in Widerspruch zu den Axiomen globaler Ontologie stehen, ist die Antwort auf die Anfrage bezüglich des Informationsintegrationssystems per Definition ungültig bzw. sinnfrei. Diese Vorbedingung ist eher theoretischer Natur, da die Überprüfung bei jeder Anfrage, ob in den Quellen Daten existieren, die widersprüchlich zur Ontologie sind, in der Praxis nicht angemessen ist.

Im Folgenden wird der Algorithmus betrachtet, der den Aufbau einer umgeschriebenen Anfrage unter der Hinzunahme von Bedingungen aus Satz 1 gewährleistet. Der vorgeschlagenen Algorithmus besteht aus folgenden Hauptschritten:

1. Umschreiben der Anfrage bezüglich Axiome globaler Ontologie.
2. Umschreiben der Anfrage bezüglich Korrespondenzen des Integrationssystems.

Des Weiteren wird angenommen, dass das Informationsintegrationssystem konsistent und die globale Ontologie normalisiert ist. Ohne Beschränkung der Allgemeinheit kann nun die Aufmerksamkeit ausschließlich auf die \mathcal{L}_{tripod} -TBox in Normalform gelegt werden, in der alle Axiome in der folgenden Formen vorliegen: $A_1 \sqsubseteq (\neg)A_2$, $A_1 \sqsubseteq \exists R_{L1}$, $A_1 \sqsubseteq \exists R_{L1}.A_2$, $\exists R_{L1} \sqsubseteq A_1$, $R_{L1} \sqsubseteq (\neg)R_{L2}$, wobei $A_1, A_2 \in N_C$ und R_{L1}, R_{L2} einfache Rollen sind. Jede \mathcal{L}_{tripod} -TBox \mathcal{T} kann in eine gleichwertige \mathcal{L}_{tripod} -TBox \mathcal{T}' in Normalform transformiert werden, indem komplexe Konzepte systematisch durch atomare Konzepte in Anlehnung an BAADER ET AL. ersetzt werden [BBL05].

1. Umschreiben der Anfrage bezüglich Axiome globaler Ontologie

Im ersten Schritt baut der Algorithmus eine Zwischenumschreibung der Benutzeranfrage $Q \in UCQ - \mathcal{L}_{tripod}$ unter der Einbeziehung von Axiomen der globalen Ontologie \mathcal{T}_G auf:

$$Q^{(1)} = \text{rewrite}(Q, \mathcal{T}_G)$$

Das Hauptmerkmal des ersten Schritts ist die „Kodierung“ bzw. die Hinzunahme von notwendigen Axiomen der Terminologie \mathcal{T}_G in die Anfrage Q . Eine direkte Berechnung der Antwort

auf die Anfrage Q hinsichtlich eines Integrationssystems Φ sieht vor, dass die logische Inferenz auf Basis von Fakten und Axiomen einer Ontologie durchgeführt wird, um implizites Wissen erheben zu können. Der vorgeschlagene Algorithmus schließt die Notwendigkeit solcher Inferenz aus. Stattdessen wird die ursprüngliche Anfrage in solcher Form umgeschrieben, sodass diese Anfrage das implizite Wissen mitberücksichtigt. Im Prinzip bedeutet die Konstruktion einer umgeschriebenen Anfrage eine Art logische Schlussfolgerung bezüglich der ursprünglichen Anfrage Q und Axiome globaler Ontologie.

Der Pseudocode des vorgeschlagenen Algorithmus $\text{rewrite}_{\mathcal{L}}$ zum Umschreiben einer Anfrage hinsichtlich der $\mathcal{L}_{\text{tripod}}$ -Ontologie ist weiter unten aufgeführt. Hierbei stellt die UCQ -Anfrage (eine Vereinigung aller konjunktiven Anfragen) eine Menge von Konjunktionen und jede Konjunktion eine Menge von Atomen dar. Es sei angemerkt, dass die Mengen keine identische Konjunktionen bzw. Atomen enthalten. Der Algorithmus erzeugt alternative Umschreibungen basierend auf den positiven Axiomen der Terminologie, indem auf die Anfrage eine Reihe von Regeln zur Substitution von Prädikaten angewandt wird (siehe Tab. 5.4). Darüber hinaus werden negative Axiome der Terminologie nicht berücksichtigt. In diesem Zusammenhang ist zu beachten, dass falls eine gegebene Wissensbasis (\mathcal{WB}) konsistent bzw. erfüllbar ist, werden die negativen Inklusionsaxiome zur Beantwortung konjunktiver Anfrage CQ nicht benötigt. Diese Behauptung wurde von CALVANESE ET AL. bewiesen [CGL⁺07]. Um die Funktionsweise des

Atom g	Axiom $\tau \in \mathcal{T}$	Substitutionsfunktion $\text{rew}(g, \tau)$
$A(u)$	$C_1 \sqcap \dots \sqcap C_n \sqsubseteq A$	$\text{rew}(g, \tau) = \bigwedge_{i=1 \dots n} C_i^{\text{rew}}$, wobei C_i^{rew} abhängig von der Form von C_i $i = 1 \dots n$ gemäß folgender Regeln bestimmt wird:
$P(u, \chi)$, wobei χ eine ungebundene Variable ist	$C_1 \sqcap \dots \sqcap C_n \sqsubseteq \exists P$	- wenn $C_i = A_i$, dann $C_i^{\text{rew}} = A_i(u)$ - wenn $C_i = \exists P_i$, dann $C_i^{\text{rew}} = P_i(u, \chi)$ - wenn $C_i = \exists P_i^-$, dann $C_i^{\text{rew}} = P_i(\chi, u)$
$P(\chi, u)$, wobei χ eine ungebundene Variable ist	$C_1 \sqcap \dots \sqcap C_n \sqsubseteq \exists P^-$	wobei χ eine neue ungebundene Variable ist.
$P(u, v)$	$P \sqsubseteq P'$	$\text{rew}(g, \tau) = P'(u, v)$
$P(u, v)$	$P^- \sqsubseteq P'$	$\text{rew}(g, \tau) = P'(v, u)$

Tabelle 5.4.: Die Regeln zum Umschreiben von Anfragen hinsichtlich $\mathcal{L}_{\text{tripod}}$ -Terminologie

Algorithmus besser erklären zu können, werden im Folgenden einige Definitionen eingeführt (analog zur Theorie der Datenbanksysteme [AHV95]).

Definition 33: Als Variablenzuordnung (Substitution) μ wird eine Abbildung von einer endlichen Menge von Variablen und Konstanten in eine endliche Menge von Variablen und Kon-

stanten bezeichnet. Hierbei weist die Abbildungsvorschrift jeder Variablen eine Konstante oder Variable und jeder Konstante dieselbe Konstante zu.

Sei eine Formel F gegeben, dann wird mit $\mu(F)$ eine Formel bezeichnet, die durch Substitution jeder Variable (oder Konstante) X in der Formel F durch $\mu(X)$ entsteht.

Definition 34: Zwei Atome (atomare Formeln) $g_1 = p(Z_1)$ und $g_2 = p(Z_2)$ sind unifizierbar, wenn es eine Variablenzuordnung (Substitution) μ gibt, die die beiden Ausdrücke unifiziert (gleich macht), d. h. es ist $\mu(g_1) = \mu(g_2)$. Eine solche Variablenzuordnung μ wird als ein Unifikator beider Atome bezeichnet.

Definition 35: μ ist ein allgemeinster Unifikator (engl. most general unifier) $\text{mgu}(g_1, g_2)$ zweier Atome $g_1 = p(Z_1)$ und $g_2 = p(Z_2)$, wenn sich alle anderen Unifikatoren durch Instantiierung der Variablen in μ ergeben, d. h.: Zu jedem Unifikator μ' gibt es eine Variablenzuordnung (Substitution) γ mit $\mu'(g_1) = \gamma(\mu(g_1)) = \gamma(\mu(g_2)) = \mu'(g_2)$.

Wird zur konjunktiven Anfrage der Unifikator $\text{mgu}(g_1, g_2)$ angewandt, wobei beide Atome g_1 und g_2 durch den gleichen Prädikat beschrieben sind, dann werden beide Variablen so umbenannt, dass beide Atome identisch werden. Ein Algorithmus zur Berechnung von mgu für eine Menge von Atomen ist aus der Datenbanktheorie bekannt [AHV95].

Definition 36: Eine Variable, die im Körper einer Anfrage vorkommt, wird als ungebunden bezeichnet, wenn diese Variable im Anfragekörper nur einmal auftritt und im Anfragekopf nicht erwähnt ist.

Eine ungebundene Variable im Algorithmus wird mit χ bezeichnet. Dabei wird verstanden, dass alle solchen Variablen sich trotz der einheitlichen Schreibweise voneinander unterscheiden. Bei der Berechnung des mgu der Atome g_1 und g_2 ersetzt der Unifikator je nach Möglichkeit jede ungebundene Variable χ in g_1 durch entsprechende gebundene Variable in g_2 . Analog verläuft die Berechnung in umgekehrte Richtung. Tretten auf bestimmter Stelle in beiden Atomen ungebundene Variablen auf, so bleibt nach der Unifikation an dieser Stelle das Symbol χ stehen. Eine Funktion zur Umbenennung aller ungebundenen Variablen im Anfragekörper durch das Symbol χ wird mit $\text{ren}(Q)$ bezeichnet.

```

Input : Konjunktive Anfrage  $Q$ ,  $\mathcal{L}_{tripod}$  - TBox  $\mathcal{T}$ 
Result:  $Q'$  – Anfrage über die Sprache  $UCQ$  (Vereinigung konjunktiver Anfragen)
 $Q' = ren(Q')$ ; // Umbenennung durch Symbol  $\chi$ 
// (1) Hauptiteration zur Bestimmung der Umschreibungen
repeat
   $Q_{temp} = Q'$  // Anfrage zwischenspeichern
  foreach Konjunktion  $q \in Q_{temp}$  do
    // (a) Umschreibung pos. Axiome
    foreach Atom  $g \in q$  do
      foreach Axiom  $\tau \in \mathcal{T}$  do
        if für  $g$  und  $\tau$  gilt:  $rew(g, \tau) \neq \emptyset$  then
           $q' = add((q/g), rew(g, \tau))$ ; // Substitution durchführen
           $Q' = add(Q', q')$ ; // umg. Anfrage hinzufügen
        end
      end
    end
    foreach Axiom  $\tau \in \mathcal{T}$  folgender Form:  $P \sqsubseteq P'$  do
      if Atom  $g \in q \wedge P \xleftarrow{predicate} g$  then
         $Q' = Q'/q$ ; // Disjunktion entfernen
      end
    end
    foreach Axiom  $\tau \in \mathcal{T}$  folgender Form:  $P^- \sqsubseteq P'$  do
      if Atome  $g_1, g_2 \in q \wedge g_1 \wedge P^- \xleftarrow{predicate} g_1 \wedge P' \xleftarrow{predicate} g_2 \wedge g_1, g_2 \neq \chi$  then
         $Q' = Q'/q$ ; // Disjunktion entfernen
      end
    end
    foreach Axiom  $\tau \in \mathcal{T}$  folgender Form:  $C_1 \sqcap \dots \sqcap C_n \sqsubseteq \perp, n \geq 1$  do
      if Atome  $g_1 \dots g_n \in q$  then
        if  $C_i = A_i$  then
           $g_i = A_i(u)$ 
        end
        else if  $C_i = \exists P_i$  then
           $g_i = P_i(u, v_i)$ 
        end
        else if  $C_i = \exists P_i^-$  then
           $g_i = P_i(v_i, u)$ 
        end
         $Q' = Q'/q$ ; // Disjunktion entfernen
      end
    end
    // (b) Anfrageminimierung
    foreach für Atompaar  $g_1, g_2 \in q$  mit gleicher Stelligkeit do
      if  $g_1$  und  $g_2$  unifizierbar then
         $q' = q/g_2$ 
         $q' = ren(mgu(q'))$ ; // allg. Unifikator anwenden
         $Q' = add(Q', q')$ ; // unif. Anfrage in  $Q'$  hinzufügen
      end
    end
  end
  if  $Q' = Q_{temp}$  then return; // Abbruchbedingung
until keine weitere Konjunktion hinzugefügt werden kann;
Algorithm 1:  $rewrite_{\mathcal{L}}$  - Algorithmus für  $\mathcal{L}_{tripod}$ -Terminologie

```

Der Algorithmus $\text{rewrite}_{\mathcal{L}}$ ist in zwei Phasen aufgeteilt. In der ersten Phase basierend auf der Anfrage Q vervollständigt der Algorithmus die Anfrage mittels neuer Disjunktionen, die einen Beitrag zu gesuchter Antwort leisten können. Dies erfolgt durch die umfassende Anwendung folgender Regeln:

- (i) Wenn ein Axiom $\tau \in \mathcal{T}$ und eine Konjunktion $q \in Q$ existieren, sodass auf ein Atom $g \in q$ die Umschreibregel, gemäß der Funktion $\text{rew}(g, \tau)$ aus der Tabelle 5.4 angewandt werden kann, dann wird das Ergebnis Q' durch eine alternative Konjunktion erweitert. Diese ergibt sich aus der Substitution des Atoms g mit $\text{rew}(g, \tau)$ entsprechend einer solchen Regel. Dabei wird jede Konjunktion als eine Menge betrachtet, d. h. eine Konjunktion verfügt über keine identische Atome und identische Atome können nicht in einer Konjunktion durch die Anwendung der Umschreibregel auftreten.
- (ii) Wenn eine Konjunktion $q \in Q$ existiert, sodass zwei Atome g_1 und g_2 in q unifizierbar sind, dann wird Q' durch eine Konjunktion vervollständigt. In einer solchen Konjunktion sind beide Atome durch ein Atom ersetzt, das mithilfe von $\text{mgu}(g_1, g_2)$ erhältlich ist. Es sei angemerkt, dass die Funktion $\text{mgu}(g_1, g_2)$ auf die gesamte Konjunktion angewandt wird. Letztendlich, da während des Unifikationsprozesses eine der Variablen ungebunden werden kann, werden alle in resultierender Konjunktion ungebundenen Variablen durch das Symbol χ umbenannt.

In der zweiten Phase werden alle a priori unerfüllbaren Konjunktionen entsorgt, die im Widerspruch zu negativen Axiomen der Terminologie \mathcal{T} stehen.

Beispiel

Es wird eine $\mathcal{L}_{\text{tripod}}$ -Terminologie mit folgendem Axiom betrachtet: Dienstleistung $\sqsubseteq \neg$ Sachleistung. Intuitiv besagt das Axiom, dass wenn ein Individuum eine Instanz der Dienstleistung ist, dann ist es keine Instanz von Sachleistung. Angenommen wird: Nach der Durchführung der ersten Phase wurde die folgende Konjunktion $\text{Dienstleistung}(x) \wedge \text{Sachleistung}(x)$ erzeugt. Es ist offensichtlich, dass das Axiom in \mathcal{T} widersprüchlich zur erweiterten Konjunktion ist.

Zu beachten ist, dass die Vereinigung konjunktiver Anfragen, die auf diese Weise berechnet werden, nicht minimal ist. Es können Paare von Anfragen vorkommen, bei denen eine Anfrage in einer anderen Anfrage enthalten sein kann. In diesem Zusammenhang soll aus praktischer Sicht eine Minimierung erhaltener Anfrage zur Erhöhung der Effizienz des Algorithmus durchgeführt werden. Dafür können bekannte Minimierungsmethoden aus der Welt relationaler Anfragen eingesetzt werden [AHV95].

Lemma 9: Es sei die Anfrage Q über die Sprache $UCQ - \mathcal{L}_{tripod}$ und Terminologie \mathcal{T} über die Sprache \mathcal{L}_{tripod} gegeben. Dann terminiert der Algorithmus $rewrite_{\mathcal{L}}(Q, \mathcal{T})$.

Beweis 9: Der Algorithmus $rewrite_{\mathcal{L}}(Q, \mathcal{T})$ terminiert zuverlässig, weil:

1. Die Anzahl möglicher Atome, die in der Anfrage verwendet werden können, ist endlich, denn:
 - a) Es gibt nur eine endliche Zahl möglicher Prädikate, die in den Atomen einer Anfrage benutzt werden können. Es sind zum einen Begriffe, die in der Anfrage erwähnt werden, zum anderen in der Terminologie \mathcal{T} selbst. Es werden keine neuen Prädikate während des Berechnungsprozesses des Algorithmus hinzugefügt.
 - b) Die Variablen-/Konstantenanzahl, die beim Umschreiben verwendet werden, ist endlich: Nach der Definition des Algorithmus, sind es entweder Variablen und Konstanten der ursprünglichen Anfrage oder das spezielle Symbol χ .
2. Die Länge einer beliebigen Konjunktion in der Anfrage kann nicht größer als die maximale Anzahl von Atomen max_atoms sein. Dies liegt daran, dass gemäß der Konstruktion alternativer Umschreibungen keine Konjunktion identische Atome beinhaltet.
3. Laut der Definition, ist die Hauptiteration dann beendet, wenn alle alternative Umschreibungskonjunktionen berechnet sind.
Die Anzahl dieser ist endlich und nicht höher als $max_atoms^{max_atoms}$. Sofern die Iteration (1) keine Konjunktion aus der Anfrage entfernt, terminiert die Iteration (1). Die restlichen Iterationen im Algorithmus terminieren offensichtlich, da diese über Elemente endlicher Mengen iterieren.

□

Lemma 10: Sei Q eine Anfrage über die Sprache $UCQ - \mathcal{L}_{tripod}$ und \mathcal{T} normalisierte Terminologie über die Sprache \mathcal{L}_{tripod} . Dann beträgt die Laufzeit vom $rewrite_{\mathcal{L}}(Q, \mathcal{T})$ -Algorithmus $O(|\mathcal{T}| \cdot |Q|^2)^{|Q|+|\mathcal{T}|}$.

Beweis 10: Im Folgenden wird die Iteration (1-a) betrachtet, da die restlichen Schleifen über die Mengen iterieren, deren Größe bzw. Elementenanzahl polinomial mit $|Q|$ und $|\mathcal{T}|$ zusammenhängt. Um die Anzahl der Durchläufe dieser Iteration abzuschätzen, wird die maximale Anzahl möglicher Konjunktionen geschätzt, die durch Iteration generiert werden können.

Es kann festgehalten werden, dass die Länge beliebiger Konjunktion in einer Anfrage $O(|Q| + |\mathcal{T}|)$ ist. Dabei ist $|Q|$ die Länge einer Anfrage und $|\mathcal{T}|$ der Umfang der Terminologie. Dies geht aus dem Sachverhalt hervor, dass einerseits alle Atome in Konjunktionen unterschiedlich sein müssen (Mengensemantik) und andererseits durch die Anwendung von Umschreiberegeln auf

ein Atom eine Menge von Atomen als Ergebnis herauskommt, deren Mächtigkeit nicht höher als $\sum_{C_L \sqsubseteq C \in \mathcal{T}} (|C_L|)$ ist. Das bedeutet, es kann kein Atom durch eine Konjunktionenfolge ersetzt werden, die länger als die Gesamtlänge aller Konjunktionen auf der linken Seite der Konzeptinklusion ist. Die Gesamtlänge beträgt hingegen $O(|\mathcal{T}|)$.

Die Anzahl möglicher Prädikate, die in den Atomen beliebiger Konjunktionen auftreten können, ist $O(|\mathcal{T}|)$. Die Zahl der Variablen und Konstanten, die in einer Umschreibung vorkommen können, beträgt $O(|Q|)$. Der Grund dafür ist, dass gemäß der Algorithmusdefinition es nur Variablen und Konstanten der ursprünglichen Anfrage oder das spezielle Symbol χ sein können. Da die Länge der Argumente eines Prädikats höchstens 2 sein kann, beträgt die Anzahl möglicher Atome $O(|\mathcal{T}| \cdot |Q|^2)$.

□

Folglich ergibt sich die maximale Anzahl möglicher Konjunktionen, die durch den Algorithmus erzeugt werden können und somit die Algorithmuslaufzeit von $O(|\mathcal{T}| \cdot |Q|^2)^{|Q|+|\mathcal{T}|}$.

2. Umschreiben der Anfrage bezüglich Korrespondenzen des Integrationssystems

Im zweiten Schritt schreibt der Algorithmus die aus dem ersten Schritt resultierende Anfrage $Q^{(1)}$ hinsichtlich der Korrespondenzen des Integrationssystems um. Hierzu besteht kein Bedarf, die Axiome globaler Ontologie einzubeziehen:

$$Q^{(2)} = \text{rewrite}_M(Q^{(1)}, M)$$

Die Aufgabenstellung zum Umschreiben der Anfrage bezüglich Korrespondenzen eines ontologiebasierten Integrationssystems kann auf die Berechnung adäquater Problemstellung für relationale Datenmodelle reduziert werden. Aus diesem Grund können im zweiten Schritt bekannte Ansätze zur Anfrageumschreibung verwendet werden, die für relationale Integrationssysteme entwickelt wurden.

Dazu wird ein relationales Informationsintegrationssystem $\Psi_{rel}(D_i, M)$ betrachtet, dessen Datenquellen relationale Datenbanksysteme sind. Die Korrespondenzen sind in so einem System in Form von $q_D \subseteq q_G$ angegeben. Zunächst wird die erhaltene Anfrage $Q^{(1)} \in UCQ$ bezüglich des Integrationssystems Ψ_{rel} umgeschrieben. Es ist bekannt, dass für ein solches System eine maximale Umschreibung der Anfrage in Form einer Vereinigung konjunktiver Anfragen aufgestellt werden kann [Hal01]. Wird die reduzierte Aufgabenstellung zurückgestellt und wird stattdessen die ursprüngliche Problemstellung fokussiert, kann die Anfrage $Q^{(2)}$ als $UCQ - \mathcal{L}_{tripod}$ -Anfrage betrachtet werden.

Der Ansatz zum Umschreiben einer Anfrage hinsichtlich GLaV-Korrespondenzen des Integrationssystem Ψ_{rel} sieht wie folgt aus:

1. Berechnung von Umschreibungen für jede konjunktive Anfrage in $Q^{(1)}$ in Bezug auf die Korrespondenzen des Integrationssystem. Hierbei bestehen Korrespondenzen aus Abbildungen der Art $\alpha_i \subseteq q_{Gi}$ für beliebige $q_{Di} \subseteq q_{Gi} \in M$. Eine solche Berechnung kann auf die Aufgabenstellung zur sichtenbasierten Anfragebearbeitung reduziert werden [Hal01].
2. Abhandlung aller α_i gemäß der Abbildungen $q_{Di} \subseteq \alpha_i$ für beliebige $q_{Di} \subseteq q_{Gi} \in M$.

Zur Lösung der Aufgabenstellung hinsichtlich sichtenbasierter Anfragebearbeitung für relationale Datenmodelle existieren einige Algorithmen, z. B.:

1. Bucket-Algorithmus [Hal01, LRO96]
2. Inverse-Rules-Algorithmus [Hal01]
3. U-join/query folding-Algorithmus [Qia96]
4. Shared-variable-bucket-Algorithmus [Mit01]
5. MiniCon-Algorithmus [Hal01, PH01]
6. Destination-based-Algorithmus [WMT02]

Die drei letzten von den oben aufgelisteten Algorithmen weisen eine höhere Performanz bei praxisrelevanten Problemen auf. Dennoch ist die asymptotische Komplexität bei allen Algorithmen vergleichsweise gleich [PH01]. Die Wahl eines bestimmten Algorithmus ist aus theoretischer Sicht nicht grundlegend; die einzige Bedingung ist, dass der Algorithmus eine maximale Anfrageumschreibung für die betrachtete Aufgabenstellung sicherstellt. In der Praxis ist es jedoch sinnvoll bzw. zweckmäßig, leistungsfähigere Algorithmen wie beispielsweise MiniCon einzusetzen.

5.4.3. Bereitstellung und Nutzung der Integrationsergebnisse

Die Integrationsergebnisse können in vielgestaltiger Ausdrucksform verwendet werden. Hierzu können die erzielten Ergebnisse zum einen in tabellarischer Form repräsentiert, zum anderen durch die Navigation und Traversierung lokalisiert werden. Darüber hinaus können Ergebnisse durch andere Anwendungen als Eingangsdaten aufgenommen und weiterverarbeitet werden. So können die Resultate z. B. zur Durchführung einer Portfolioanalyse oder zum Abgleich und automatischer Vervollständigung der Metadaten in einem nachgeschalteten System eingesetzt werden. Ferner sind weitere zahlreiche Szenarien vorstellbar.

Folglich ist eine applikationsneutrale Schnittstelle unumgänglich, die einerseits die Handhabung von Integrationsprozessen ermöglicht, andererseits die Bereitstellung von Integrationsergebnissen sicherstellt. Daher soll die Schnittstelle über folgende funktionale Merkmale verfügen:

- *Systemverwaltung*: Die Systemverwaltung stellt den Vorgang zur Authentifizierung und Autorisierung eines Anwenders sicher sowie die Möglichkeit der Prozessteuerung. Hierzu soll dem Anwender erlaubt sein beliebig viele Anfrageprozesse auszulösen.
- *Metadatenbereitstellung*: Alle in der globalen Ontologie spezifizierten Begrifflichkeiten sind verfügbar: Konzept- und Rollenspezifikation, Korrespondenzspezifikation, konzeptioneller Taxonomiestruktur sowie die verfügbaren Datenquellen.
- *Prozessspezifikation*: Einem Integrationsprozess liegt dessen Spezifikation zugrunde. Hierzu muss der Diskursbereich festgelegt werden, der durch die globale Ontologie beschrieben ist und nach dem die Integration erfolgen soll sowie Bestimmung der zu integrierenden Datenquellen (u.U. kann das relationale Schema reduziert werden). Durch die manuelle Verifikation wird der Sachverhalt geklärt, ob die Datenquellen der ausgewählten Integrationssicht (bzw. Diskursbereich) entsprechen und folglich integriert werden können.
- *Prozesssteuerung*: Zur Prozesssteuerung gehört neben dem Starten und Abbrechen eines Anfrageprozesses auch das Monitoring des aktuellen Prozessstatus sowie eine Fehlerbehandlung. Darüber hinaus sollen beim Erreichen eines Zustands, bestimmte Aktivitäten ausgelöst werden, die beispielsweise eine endlose Abarbeitung der Anfrage verhindern.
- *Ergebnisabruf*: Die Ergebnisse einer Anfrage können als ein einziges Gesamtergebnis sowie quellenweise abgerufen werden, d. h., die Individuen werden bereitgestellt, die einer Datenquelle angehören, deren Schemaelemente mit Konzepten und Rollen der globalen Ontologie in Beziehung gesetzt sind.

Einerseits können die Anfrageergebnisse direkt von einem Anwender abgerufen und weiterverwendet werden, andererseits kann eine Anwendung ggf. durch einige Transformationsvorgänge die Anfrageergebnisse zur Weiterverarbeitung verwerten. Darüber hinaus können Ergebnisse zur Verbesserung bzw. Anreicherung der bestehenden Ontologie herangezogen werden, indem sie als Eingabedaten für die in den Abschnitten 5.2.1 und 5.3.1 beschriebenen Ansätzen der Konzept- und Rollenermittlung und Korrespondenzspezifikation dienen. Basierend darauf wird die Terminologie der Wissensbasis – nach einer entsprechenden manuellen Überprüfung – erweitert.

5.4.4. Systemarchitektur

In Abbildung 5.10 ist die Architektur des Systems zur ontologiebasierten Informationsintegration dargestellt. Die Architektur ist durch einen streng modularen Aufbau gekennzeichnet, was eine leichte Wartbarkeit und Erweiterbarkeit des Systems gewährleistet.

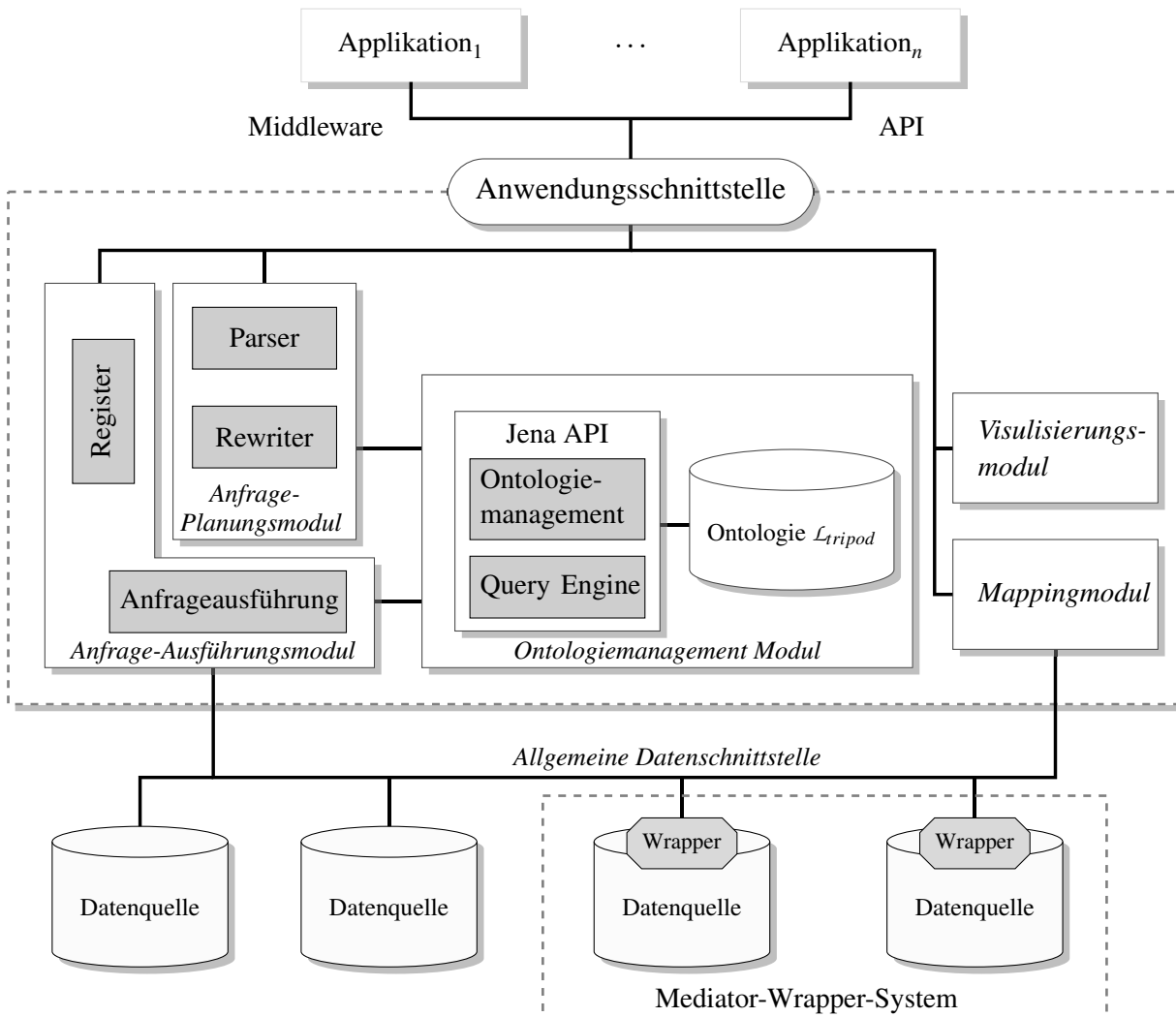


Abbildung 5.10.: Systemarchitektur

Kernkomponenten der Systemarchitektur sind die operativen Komponenten: *Anfrage-Planungs-*, *Anfrage-Ausführungs-*, *Ontologie-Verwaltungs-*, *Mapping-* sowie *Visualisierungsmodul*. Das Anfrage-Planungsmodul dient dem in Abschnitt 5.4.1 beschriebenen Umschreiben der Anfrage und erzeugt unter Hinzunahme der logischen Schlussfolgerung eine konjunktive Anfrage. Das Anfrage-Ausführungsmodul ist für die Föderation, Optimierung und Ausführung von Anfragen, die an das IIS gesendet werden, verantwortlich. Das Anfrage-Ausführungsmodul greift auf die konjunktive Anfrage zu, erzeugt auf Basis der beschriebenen Korrespondenzen verteilte SQL-Anfragen und delegiert diese an entsprechende Datenquellen bzw. dazugehörige Adap-

toren weiter. Dabei ist die Registerkomponente für das An- und Abmelden von Datenquellen verantwortlich. In der Regel erfordert die Registrierung einige Metadaten und ein Mappingmodell der Datenquellen. Die Anfrage-Ausführungskomponente liest mithilfe des Mappingmoduls die spezifizierten Korrespondenzen ein und baut analog der konjunktiven Anfrage eine vereinigte SQL-Anfrage auf, die die Ergebnisse eines Integrationsprozesses ermittelt. Das Ontologieverwaltungsmodul enthält eine OWL-Ablage für die globale \mathcal{L}_{tripod} -Ontologie und ein API zur Abfrage und Pflege der Ontologie. In diesem Fall ist es das Jena-API (Jena Community [Jen11]).

Die Nutzdaten umfassen alle Datenquellen, die über spezielle Datenschnittstellen, Interpretationsroutinen oder direkt (integrierte Produktmodelle wie z. B. STEP [ISO94] oder das integrierte Produkt- und Produktionsmodell (PPM) [GMH95], [GM97] an die *allgemeine Datenschnittstelle* angebunden sind.

Die Steuerung des Integrationsprozesses (Authentifizierung, Auswahl der zu integrierenden Datenbestände, Start der Anfrageausführung etc.) sowie das Abrufen der Integrationsergebnisse erfolgt über die *Anwendungsschnittstelle*, die die Anbindung externer *Anwendungsapplikationen* (z. B. ein PDM-System, dessen Metadaten gepflegt werden sollen) ermöglicht.

Das Visualisierungsmodul ermöglicht die Repräsentation der ontologischen Gesamtsicht auf den zu integrierenden Datenbestand, die Konfiguration des Anfrage-Planungsmoduls (z. B. Auswahl der anzuwendenden Parser-Methoden, Einschränkung des zu berücksichtigenden Datenbestand etc.) sowie die Visualisierung, Verifizierung und Anpassung der erzeugten Ergebnisse.

*Verstand besteht nicht nur im Wissen,
sondern auch in der Fähigkeit, das
Wissen in der Tat anzuwenden.*

ARISTOTELES (384 v. Chr.–322 v. Chr.)

6. Realisierung und Validierung des Ansatzes

In diesem Kapitel wird die Validierung des vorgestellten Konzepts anhand der Realisierung eines Prototyps des auf der Beschreibungslogik basierenden Informationsintegrationssystems erläutert. Die Zielsetzung der durchgeführten Realisierung war es, ein System auf Basis moderner Technologien zu entwickeln, das in einer verteilten Umgebung eingesetzt werden kann.

Um die Praxisrelevanz des Konzepts besser veranschaulichen zu können, werden hierfür wesentliche Punkte mithilfe eines konkreten Anwendungsszenarios aus der industriellen Praxis erläutert. Das Anwendungsszenario umfasst in erster Linie die Konzeption und Realisierung der Integrationslösung für einen *Mass-Customization*-Ansatz. Zu Beginn der Konzeptvalidierung wird das Anwendungsszenario sowie eingesetzte Entwicklungsumgebung und beteiligter Systeme vorgestellt.

6.1. Beschreibung des Anwendungsszenarios

Immer mehr Anbieter erkennen, dass in vielen Märkten nur noch eine radikale Abkehr von Massenproduktion zu dauerhaften Wettbewerbsvorteilen führen kann [OBAB11]. Zu den Ursachen zählen die allgemeine Tendenz zur Designbestimmung, Umweltbewusstsein und vor allem ein neues Qualitäts- und Funktionalitätsbewusstsein des Kunden [OBM11]. Um diese Anforderungen erfüllen zu können, sind stark verzahnte Kunden-Hersteller-Lieferanten-Beziehungen, die als Produktionsnetzwerke bezeichnet werden, unerlässlich. In diesem Kontext besteht ein essenzieller Bedarf an einer geeigneten informationstechnischen Unterstützung, um die Wettbewerbsfähigkeit beteiligter Unternehmen langfristig sicherstellen zu können.

Einen Realisierungsansatz dazu liefert das EU-Projekt *e-CUSTOM*¹. Ziel des Projekts ist die Entwicklung eines webbasierten Kollaborationssystems für das erweiterte Mass Customization-Produktionskonzept. Im Mittelpunkt stehen dabei Gestaltung und Abwicklung der innerbetrieblichen und unternehmensübergreifenden Informations- und Kommunikationsprozesse des Herstellers mit seinen Kunden und Lieferanten, da der Information eine zentrale Rolle innerhalb der kundenindividuellen Massenproduktion zukommt. Hierzu setzt sich das webbasierte Kollaborationssystem aus in Abbildung 6.1 dargestellten vier Hauptbestandteilen zusammen.

¹e-CUSTOM-A Web-based Collaboration System for Mass Customization. Call identifier: FP7-2010-NMP-ICT-FoF, Grant agreement no.: 260067, 06.2010-05.2013, <http://www.ecustom-project.eu/>

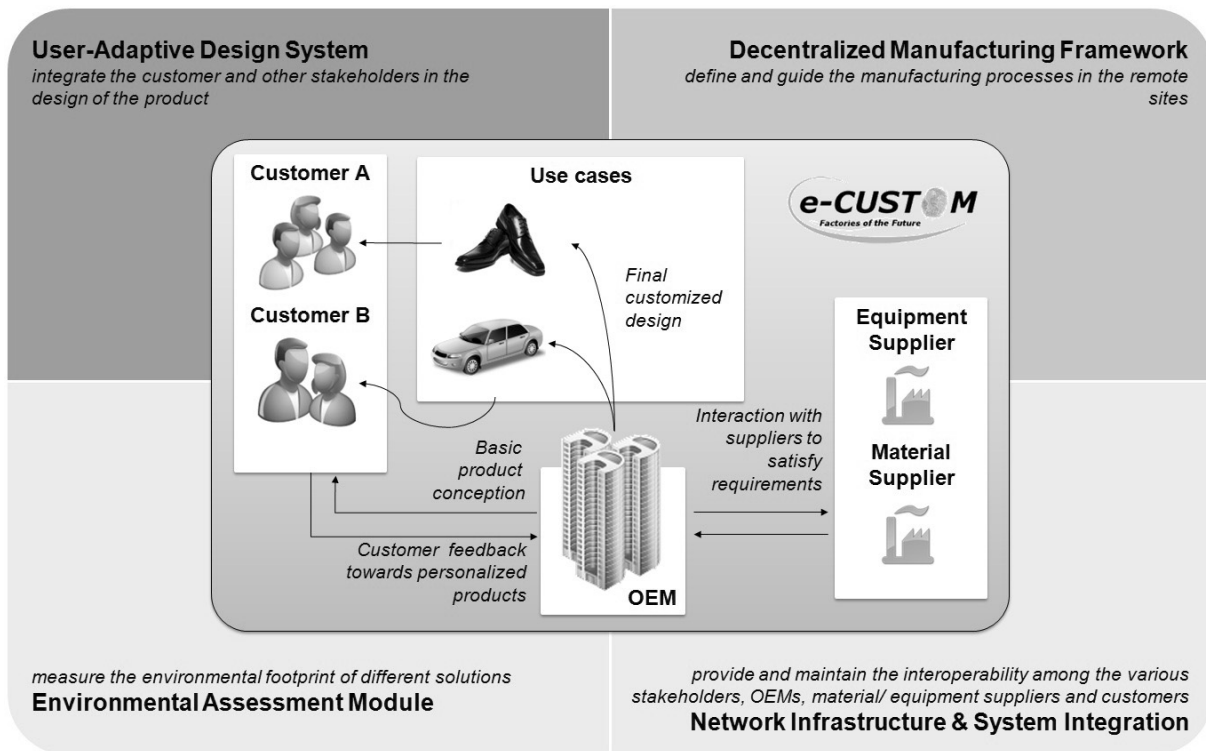


Abbildung 6.1.: e-CUSTOM Lösungsansatz [OBAB11]

Das User-Adaptive Design System stellt dem Kunden eine intuitiv zu bedienende Interaktionsschnittstelle zur Verfügung. Mithilfe dieser kann der Kunde einerseits ein Produkt nach seinen persönlichen Vorstellungen und Bedürfnissen gestalten, andererseits sein Feedback zu bestehenden Produktkonfigurationen und Produktmerkmalen mitteilen. Dem Kundenfeedback entsprechend können Produktverbesserungen bzw. Innovationen seitens des Herstellers angestoßen werden. Anders als herkömmliche Produktkonfiguratoren, gibt das User-Adaptive Design System dem Kunden auch die Möglichkeit, die Geometrie des Produkts anhand einer webbasierten CAD-Umgebung zu manipulieren. Ein weiterer Aspekt stellt die dezentrale Fertigung dar, welche durch das Decentralized Manufacturing Framework realisiert wird. Der Fokus liegt hierbei auf der Zusammenarbeit zwischen Hersteller, Zulieferer und Händler basierend auf der dezentralen Fertigungsstrategie, sodass Fertigungs- und Arbeitspläne kontextsensitiv generiert werden können. Dabei setzt sich der Kontext aus zahlreichen Umgebungseigenschaften zusammen. Diese werden durch das Environment-Assessment-Modul in Beziehung gesetzt und spiegeln die Restriktionen des Produktionsnetzwerks hinsichtlich verfügbarer Fertigungstechnologien, Materialien und Kapazitäten sowie die Präferenzen – z. B. Preis, Lieferdatum, CO₂-Bilanz etc. – des Kunden wider. Die Kollaboration mehrerer Teilnehmer setzt agile Netzwerkinfrastruktur und den Einsatz zahlreicher bestehender Anwendungen bzw. IT-Systeme voraus. Die bestehenden Informationssysteme können vielfältig sein: von klassischen relationalen Datenbanksystemen bis zu Daten, auf die man mittels Webservices zugreift. Daher stellt die Integration

heterogener Systeme einen wesentlichen Aspekt des Kollaborationssystems dar. Um die Informationsintegration gewährleisten zu können, soll durch eine zentrale, integrierte Komponente der Zugriff auf innerbetriebliche und unternehmensübergreifende Informationen durch Anwender (Hersteller, Zulieferer, Kunde) und Anwendungen (PDM, ERP, SCM) sichergestellt werden.

Die Validierung des webbasierten Kollaborationssystems für das erweiterte Mass-Customization-Produktionskonzept erfolgt anhand zweier industrieller Anwendungsbeispiele. Diese kommen zum einen aus dem Bereich der Automobilindustrie und zum anderen aus einem Teilbereich des Gesundheitswesens zur Herstellung individuellen orthopädischen Schuhwerks.

Die Abbildung 6.2 gibt eine abstrakte Architektur der e-CUSTOM-Plattform wieder.

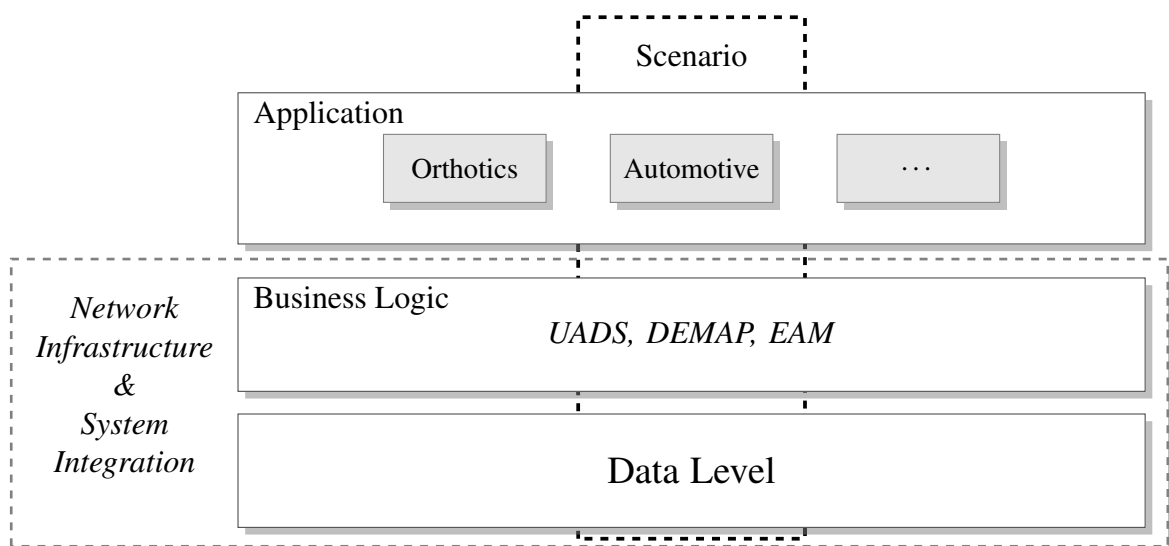


Abbildung 6.2.: Abstrakte Architektur des e-CUSTOM-Systems

In diesem Zusammenhang muss einerseits der automatisierte Datenaustausch zwischen dem e-CUSTOM-System und unternehmensspezifischen Systemen implementiert sowie dem Anwender eine konzeptionelle Sicht in Form einer Ontologie auf das System sichergestellt werden. Dabei stellt die Ontologie ein Modell des zu entwickelnden Produkts dar, bildet aber auch die Projekt- und Linienorganisation ab sowie die fachlichen Zusammenhänge und Interaktionen. Darauf basierend kann ontologiebasierte Informationsintegration die Koordination wesentlich unterstützen, indem folgende Fragestellungen beantwortet werden können:

- Welcher Hersteller stellt das Produkt x her?
- In welchen Dokumenten wird die Baugruppe y beschrieben?
- Wie hängt das Produkt x mit Zulieferer z zusammen?

- Mit wem muss man Änderungen am Produkt x koordinieren?
- Wer ist organisatorisch, wer ist fachlich für das Projekt p zuständig?

Die Antworten auf diese Fragestellungen sind in verschiedenen Dokumenten in unterschiedlichen Systemen festgehalten. Sie zu erschließen ist eine wichtige Aufgabe, für die sich eine Ontologie hervorragend eignet, vor allem wenn sie bereits den Entwicklungsprozess begleitet hat.

6.2. Entwicklungsumgebung und beteiligte Systeme

Zur Umsetzung und Validierung des entwickelten Konzepts zur semantischen Informationsintegration wurde unter Nutzung der im Folgenden dargestellten Entwicklungsumgebung basierend auf in dem vorangegangenen Kapitel beschriebener Systemarchitektur vorgenommen:

Hardware

- Intel®Core™2 Quad CPU Q9450 @ 2.66 GHz 2.67 GHz mit 8.0 GB Arbeitsspeicher, 170 GB freie Festplattenkapazität
- Betriebssystem: Windows 7 Professional[©] Service Pack 1, 64 Bit

Software-Entwicklungsumgebung

- NetBeans² IDE 7.1 Beta (Build 201109252201) (Java Integrated Development Environment (IDE))
- Java JDK³ 1.6.0_27; Java HotSpot(TM) Client VM 20.2-b06 (Java Entwicklungsumgebung)
- JavaFX⁴ 2.0 (Laufzeitumgebung der Fa. Oracle ursprünglich der Fa. Sun Microsystems)
- Jena⁵ API 2.6.4 (Java Framework zur Entwicklung von Semantic Web Anwendungen, ursprünglich von HP Labs entwickelt 2000)
- JDBC⁶ (Java Database Connectivity) Version 4.0
- MySQL⁷ Version 5.5.21 (Open-Source relationales Datenbankverwaltungssystem)

²<http://netbeans.org/>

³<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

⁴<http://javafx.com/>

⁵<http://incubator.apache.org/jena/>

⁶<http://www.oracle.com/technetwork/java/javase/jdbc/index.html>

⁷<http://www.mysql.com>

Die Programmiersprache Java erlaubt samt der verfügbaren Entwicklungsumgebung und der großen Anzahl verfügbarer Modullösungen (Java-Bibliotheken) eine zügige Software-Entwicklung. Die ausgereiften Kommunikationsmethoden und Java-Interfaces tragen zur Realisierung einer modularen Struktur des Prototyps bei. Die Implementierung einer Architektur, bei der ein Client auf mehrere Datenbanken zugreift, wurde problemlos durchgeführt.

Für die Entwicklung der grafischen Benutzeroberfläche des Softwareprototyps wurde die Programmiersprache Java verwendet. Hierbei wurde der Gebrauch von den Zusatzbibliotheken JavaFX gemacht, um die Anfrageergebnisse optisch besser darstellen zu können. JavaFX wurde primär aufgrund ihrer mächtigen Werkzeuge und gleichzeitig einfachen Handhabung gewählt; jedoch könnte die visuelle Kommunikation genauso mithilfe von Swing⁸-Bibliotheken realisiert werden. Ferner, aufgrund zahlreich verfügbarer Modullösungen, wurden die Methoden sowie die gesamte Softwarearchitektur des Prototyps in der objektorientierten Sprache Java umgesetzt. Bei der Implementierung der funktionalen sowie der grafischen Bestandteile des Prototyps hat sich die Entwicklungsumgebung *NetBeans* als sehr vorteilhaft erwiesen. Der entwickelte Java-Client erzielte eine höchst zufriedenstellende Leistung auf der zugrunde liegenden Hardware.

Zur Kommunikation zwischen dem Prototyp und den Datenbanksystemen wird die JDBC-Datenbankschnittstelle eingesetzt. Die Datenbankanbindung über JDBC wies keine Probleme auf. In der Datenbank sind drei logisch getrennte Bereiche jeweils für das Speichern von Produktdaten sowie unternehmensspezifischer Daten, die seitens *User Adaptive Design System* (UADS) verwaltet werden, Netzwerk- und Produktionsdaten, die seitens *Decentralized Manufacturing Framework* (DEMAP) verwaltet werden, Umweltfaktoren und damit in Zusammenhang stehende Hilfsdaten, die seitens *Environmental Assessment Module* (EAM) verwaltet werden. Das Datenrepository ist physisch bis dahin durch eine Datenbank realisiert, jedoch steht einer Erweiterung auf mehrere Datenbanksysteme nichts entgegen. Der verwendete MySQL-Datenbankserver überzeugte höchst akzeptable Antwortzeiten.

Für die Entwicklung der RDF(S)/OWL-Schnittstelle basierend auf der XML-Technologie wird Java-Klassensammlung *Jena API* verwendet. Sowohl die Einbindung als auch die Performanz von Jena API ist zufriedenstellend. Die zu verwaltende OWL-Datei, deren Struktur an XML angelehnt ist, wird zunächst einer Konsistenzprüfung unterzogen und anschließend als *File* zwischengespeichert.

⁸Die Java-Klassenbibliothek „Swing“ ist eine Programmierschnittstelle (API) und Sammlung von Klassen zur Entwicklung von grafischen Benutzerschnittstellen. Seit 1998 ist ein Bestandteil der frei verfügbaren Java Development Kit (JDK).

6.3. Vorgehensschritte bei einer semantischen Informationsintegration

Im Kapitel 5 wurde der Ansatz zur automatischen Abbildung relationaler Quellen in einer ontologischen Struktur vorgestellt. Gleichzeitig wurde angemerkt, dass dies nur als eine initiale Struktur betrachtet werden kann und eine manuelle Erweiterung bzw. Anreicherung durch eine Gruppe menschlicher Anwender bedarf.

Folglich besteht die Aufgabenstellung an dieser Stelle zunächst darin, basierend auf der ISO Normreihe 10303 STEP eine Basisontologie aufzustellen. Analog zu diesem Vorgang werden im Weiteren unternehmensspezifische Beschreibungen und Strukturierungen von Produktdaten abgebildet. Im nächsten Schritt findet das Schema Mapping statt, indem semantische Beziehungen zwischen korrespondierenden Begriffen aus den unterschiedlichen Datenquellen und der globalen Ontologie definiert werden. Sind quellenspezifische Schemata im System erfasst, erfolgt auf Basis dieser Informationen die kontextspezifische Wissensakquisition, indem Anwender produktspezifische Informationen entsprechend ausgewählter Datenquellen charakterisieren. Der Mehrwert aus Anwendersicht wird abschließend am Beispiel der semantischen Suche demonstriert.

Die Funktionalitäten der Softwareprototypen zur semantischen Informationsintegration sowie der Informationsbereitstellung werden anhand von drei Vorgehensbeschreibungen erläutert, die in den anschließenden Abschnitten vorgestellt werden:

- Die Vorgehensweise zum Ontologieentwurf beschreibt das Verfahren zur Erstellung bzw. manueller Anreicherung der globalen Ontologie.
- Die Spezifikation datenquellenspezifischer Korrespondenzen beschreibt semantische Zusammenhänge zwischen globaler Ontologie und den Elementen heterogenen Datenmodelle.
- Die Vorgehensweise zur ontologiebasierten Informationsbereitstellung beschreibt das Verfahren zur kontextspezifischen Anfrageformulierung und datenquellenübergreifenden Zugriff auf Wissen.

Für den einheitlichen Aufbau der Ontologie wird in dieser Arbeit die folgende Benennungskonvention für Konzepte, Rollen und Axiome verwendet. Da das Ontologiemodell das logische Informationsmodell des Integrationssystems darstellt, dient dieses der Verbesserung der Übersichtlichkeit und erleichtert das Zusammenführen unterschiedlicher Datenquellen. Dabei sind die Einzelheiten der Benennung von dem zum Aufbau der Ontologie eingesetzten Werkzeug abhängig. Es ist zu beachten, dass durch das eingesetzte Werkzeug reservierte Wörter nicht verwendet werden dürfen, z. B. *class*, *individual* etc. Im Allgemeinen werden folgende Bestimmungen festgelegt:

- Die Mehrsprachigkeit aller Bezeichnungsdefinitionen soll unterstützt werden.
- Bestimmte Namensdefinitionen können ebenfalls mit Synonymen versehen werden, die aus einem bereits vorhandenem Lexikon zugewiesen werden. Das Lexikon kann ebenfalls mit neuen Begriffen angereichert werden.
- Der erste Buchstabe eines Konzept- oder Rollennamens wird großgeschrieben.
- Zusammengesetzte Konzept-/Rollenbezeichnungen werden jeweils mit großen Anfangsbuchstaben ohne ein Trennzeichen geschrieben, z. B. „FertigungsPlan“.
- Konzepte werden in Singularform spezifiziert, z. B. „Produkt“, statt „Produkte“.

6.4. Vorgehensschritte beim Ontologieentwurf

Es soll das Wissen zahlreicher Produkt- und Dienstleistungsanbieter in einem Kollaborationssystem für Mass Customization aufgenommen werden, um eine organisationsübergreifende Zusammenarbeit zu unterstützen bzw. um das genannte Kollaborationssystem als alternative Entwicklungsplattform zur Kommunikation sowie als ein Vertriebswerkzeug zur Marktdurchdringung von Sach- und Dienstleistungen zu nutzen. Das Unternehmenswissen wird zunächst in Form von Best-Practice-Methoden, Dokumenten, Modellen aus der produkt- bzw. dienstleistungsspezifischen Sicht beschrieben und im Anschluss daran durch eine konzeptionelle Struktur repräsentiert. Um auf eine neutrale Art und Weise in einer nachfolgenden Bereitstellungsphase auf das vorhandene Wissen zuzugreifen, müssen die unternehmensspezifischen heterogenen Datenquellen durch Schema Mapping miteinander semantisch „verknüpft“ werden. Die zielgerichtete Bestrebung ist es, wenn z. B. entsprechend einem anwendungsspezifischen Schema nach bestimmten Informationen gesucht wird, sollen auch diejenigen Informationen aufgefunden werden, deren Charakterisierung unterschiedlicher Anwendungen bzw. Systemen zugrunde lagen.

6.4.1. Die Benutzungsoberfläche des Ontologie-Entwicklungswerkzeugs

Werkzeuge zur Unterstützung der Entwicklung einer Ontologie sollen über Funktionalitäten zur Erstellung, Visualisierung und Überprüfung von Ontologie verfügen. Die Benutzungsoberfläche dieser Werkzeuge ist vor allem für Wissensingenieure und Domänenexperten bestimmt und weniger für Softwareentwickler. Im Rahmen der vorliegenden Arbeit wird das an der Stanford University School of Medicine entwickelte Werkzeug *Protégé*⁹ eingesetzt. Das aktuelle System Protégé-OWL zeichnet sich durch Kompatibilität mit anderen Wissensrepräsentationssystemen,

⁹<http://protege.stanford.edu/>

durch die Unterstützung der Standards und einer benutzerfreundlichen Oberfläche aus. Die Abbildung 6.3 zeigt einen Bildschirmabzug des Protégé-OWL Systems.

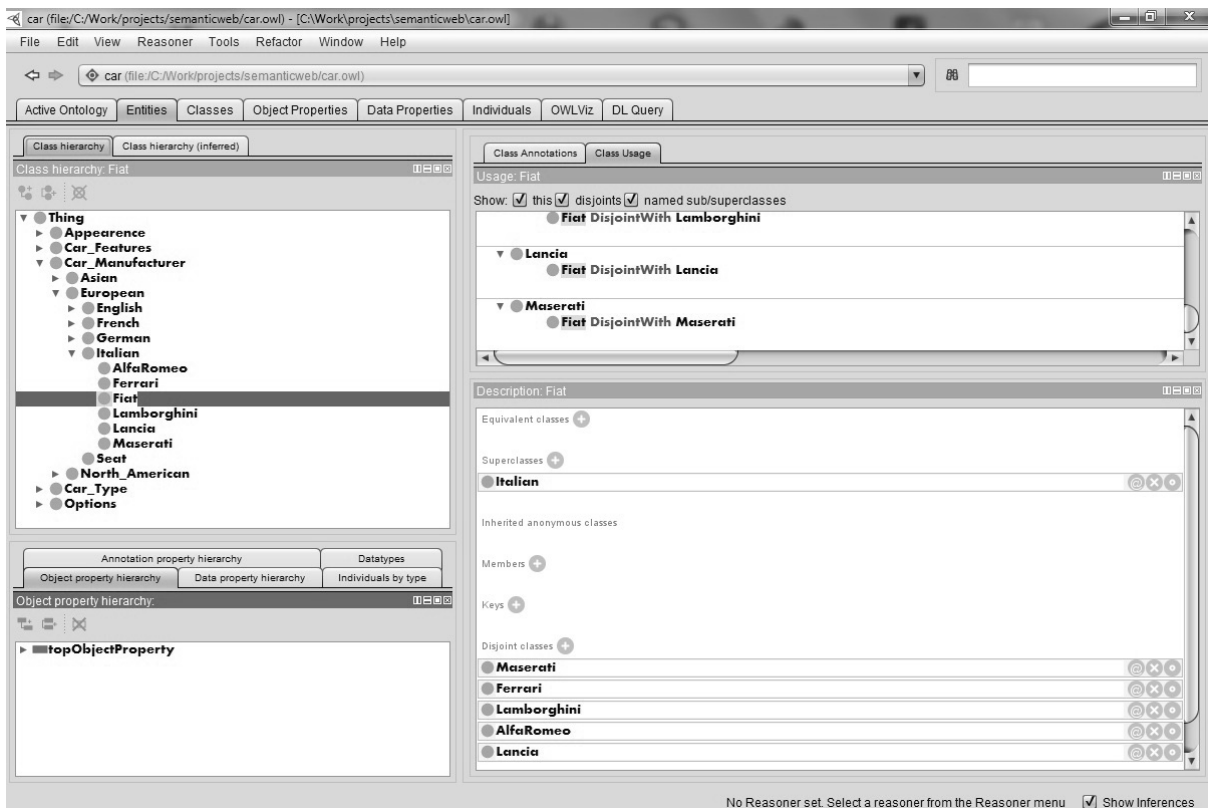


Abbildung 6.3.: Benutzungsoberfläche des Protégé-OWL Systems

Das Hauptfenster von Protégé besteht aus sogenannten Karteireitern (engl. *tabs*), um unterschiedliche Aspekte des zu entwerfenden Wissensmodell wiederzugeben. Einer der bedeutenden Karteireiter, zumindest zu Beginn der Anreicherung der Ontologie, stellt der Konzeptreiter *Classes* dar. Die *Klassen* in Protégé entsprechen den Objekten und Typen von Objekten, also Konzepten des betrachteten Anwendungsbereichs. In Anlehnung an das vorgestellte Anwendungsszenario enthalten die Klassen Begriffe wie „Baugruppe“, „Artikel“, „Teil“ etc. Die Klassen werden in Protégé in Form einer Vererbungshierarchie (engl. *inheritance hierarchy*), die sich im Navigationsbereich im linken Teil des Klassenkarteireiters befindet. Mithilfe des Navigationsbereichs ist es möglich, neue Klassen, Unterklassen zu erzeugen oder die bestehende Taxonomie zu ändern.

Wie bereits festgestellt wurde, bilden Klassen im Protégé-OWL System die Begriffe der behandelten Domäne ab. Hierbei stellen Klassen etwas mehr dar als nur Objekte, die in einer Hierarchie zusammengefasst sind. Sie können darüber hinaus über Attribute (Eigenschaften) z. B. „Länge“, „Gewicht“ und Beziehungen zwischen diesen, beispielsweise „ein Teil gehört zur Baugruppe“, verfügen. Attribute und Beziehungen einer Klasse werden durch einen Kon-

strukt namens *slot* beschrieben. Protégé bietet die Möglichkeit slots zu erzeugen, slots mit den Klassen zu verknüpfen und slots zur Beschreibung von Klassenbeziehungen einzusetzen. An dieser Stelle kommt der Vererbungsmechanismus von Klassen zum Tragen, indem jede Unterklasse automatisch alle slots der Basisklasse erbt. Unterklassen mit mehrfacher Vererbung, d. h. Unterklassen, die von mehreren Basisklassen erben, übernehmen slots von allen Basisklassen. Die Mehrfachvererbung ist eine der grundlegenden Eigenschaften des Systems Protégé. Die slots sind zunächst einfach, jedoch können diese ebenfalls mit Eigenschaften versehen werden. Zum Beispiel „Länge“ ist immer durch eine Zahl zu beschreiben. Slot-Eigenschaften werden in Protégé als *facets* bezeichnet und können entweder im Klassenkarteireiter oder im Slotkarteireiter erzeugt werden.

Darüber hinaus erlaubt Protégé die Erzeugung von konkreten Ausprägungen, den sogenannten *exemplars* von Klassen und Slots. Auf die Erklärung dieser wird jedoch bewusst verzichtet, da im Rahmen der vorliegenden Arbeit die Fakten nicht in eine Ontologie aufgenommen werden, sondern sind direkt von den beteiligten Datenquellen zu beziehen.

6.4.2. Erstellung der Ontologie

Im Kontext des vorliegenden Anwendungsszenarios „Kollaborationssystem für Mass Customization“ wird die internationale Norm ISO 10303 als Grundlage der globalen Ontologie betrachtet.

Im Fall eines Kollaborationssystems kann der betrachtete Diskursbereich erweitert werden, wenn beispielsweise andere Anwendungsbereiche in Betracht gezogen werden, kann die Norm ISO 10303 nach wie vor als globale Ontologie für die Anwendungsdomäne „Automobilindustrie“ angewandt werden, muss jedoch in die gesamte konzeptionelle Struktur eingegliedert werden. So kann z. B. die gesamte Norm ISO 10303 in einen umfassenderen Kontext eingeordnet werden, der durch die *Universal Decimal Classification* (UDC) repräsentiert werden kann.

Aufgrund der Kategorisierung des gesamten menschlichen Wissens stellt die UDC zwar ein Universalklassifikationssystem dar, gleichwohl wird dadurch das Wissen nur in die Breite jedoch nicht in die Tiefe behandelt. Folglich müssen spezielle Anwendungsdomänen mithilfe von Spezialklassifikationssystemen beschrieben werden. So kann durch Spezialklassifikationssysteme, z. B. ISO 10303, der Detaillierungsgrad von bestimmten Bereichen des Universalklassifikationssystems, z. B. UDC, erweitert werden.

Die Abbildung 6.4 zeigt ein Phasenmodell der Prozessaktivitäten zur Ontologierstellung im Umfeld der Produktentwicklung. Der Gesamtprozess von der Problemerkennung bis hin zur Implementierung des Ontologiemodells erfolgt in vier Teilprozessen, die nachfolgend beschrieben werden.

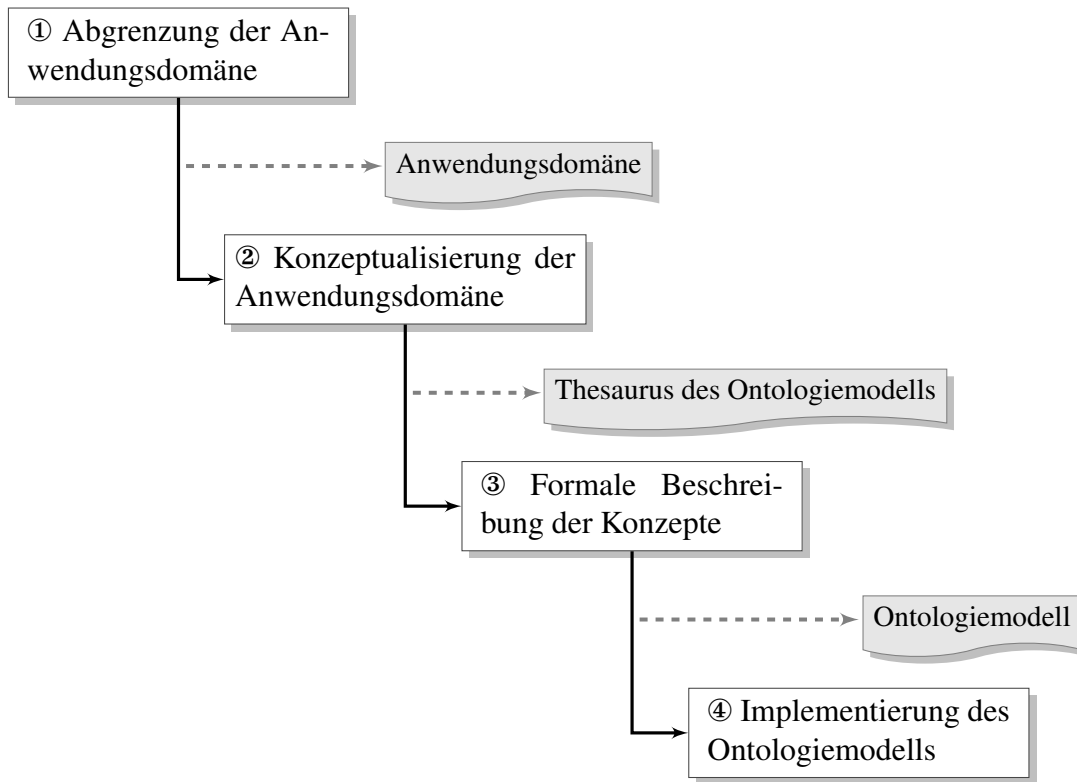


Abbildung 6.4.: Phasenmodell zur Ontologierstellung im Kontext der Produktentwicklung

① Abgrenzung der Anwendungsdomäne

Innerhalb dieses Teilprozesses erfolgt die Spezifikation von Anforderungen sowie die Aufstellung des Gesamtplans des allgemeinen Problemlösungsprozesses. Das Resultat dieser Phase ist einerseits ein klarer, jedoch zunächst informeller Entwurf der Anwendungsdomäne, andererseits wird im Zusammenhang mit der Anwendungsdomäne das zu lösende Problem analysiert und verdeutlicht. Darüber hinaus wird die Anwendungsdomäne intentionalisiert. Beispiele hierfür können sein:

- Produktinformationen
- disziplinübergreifende Begrifflichkeit zum Entwurf eines Kollaborationssystems für *Mass Customization*
- Unternehmensinformationen

② Konzeptualisierung der Anwendungsdomäne

In diesem Teilprozess werden für den gesamten Ontologierstellungsprozess relevante Konzepte und Rollen identifiziert und erzeugt. Bei diesen Aktivitäten findet die Konzeptualisierung im Sinne einer Begriffsbildung für den Anwendungsbereich statt. Das Ergebnis der Konzeptualisierungsphase ist ein „Thesaurus des Ontologiemodells“, der zur Beschreibung vorgegebener Domäne dient. Beispiele hierfür können sein:

- Begriffe zur Definition von produktbeschreibenden Informationen (Fragestellung: Welche prinzipielle Begriffe liegen der Charakterisierung von Sach- und Dienstleistungen zugrunde? Antwort: Prinzipielle Begriffe zu Konzeptbildung können sein: Bezeichnung, Revision, Iteration, Lebenszyklusstatus, Konfigurationsspezifikation etc.).
- Begriffe zur Festlegung einer Terminologie zur web-basierten Kollaboration im Rahmen eines *Mass Customization Systems* (Fragestellung: Die Kollaboration welcher Zielgruppen soll durch das gemeinsame Vokabular verbessert werden? Antwort: Zu unmittelbar profitierenden Zielgruppen können angehören: Hersteller (Original Equipment Manufacturer *OEM*), Zulieferer (engl. *supplier*) und Abnehmer (engl. *customer*); Ziel ist es, die einzelnen Begrifflichkeiten von korrespondierenden Adressaten zu erfassen und gegenüberzustellen. Fragestellung: Welche Begriffe sollen durch das Vokabular unterstützt werden? Antwort: Zu unterstützende Begriffe können sein: Herstellung, Umweltbeeinflussung, Portfolio etc.).
- Begriffe zur Definition von unternehmerischen Zusammenhängen, um Organisationsstruktur zu veranschaulichen. (Fragestellung: Welche Strukturen eines Unternehmens sollen beschrieben werden? Antwort: Zu berücksichtigende Unternehmensstrukturen können sein: Abteilung, Team, Mitarbeiter etc.).

③ Formale Beschreibung der Konzepte

In diesem Teilprozess erfolgt der Entwurf des Ontologiemodells, indem die Ergebnisse der vorangegangenen Konzeptualisierungsphase semantisch durch eine formale Sprache beschrieben werden. Basierend auf den Ergebnissen vorhergehenden Phasen sind folgende Ontologiemodelle für die vorgestellten Szenarien denkbar:

- Produktmodell nach ISO-10303-214. Alternativ dazu kann beispielsweise das *Core Product Model* (CPM) verwendet werden [Fen02].
- Entwurf eines Thesaurus zur Erstellung eines Kollaborationssystems für *Mass Customization*.
- „Informationstechnische Formalismen“ zur Beschreibung einer Unternehmensstruktur.

Hierzu sollen beim Modellieren einer Ontologie folgende Anforderungen mitberücksichtigt werden:

1. Darstellung von Begriffen, die im Hinblick sprachgebräuchlicher Verwendung eindeutig benannt sind.
2. Möglichkeit, Begriffe mehrsprachig darzustellen.
3. Grundbausteine zur Erstellung einer Klassifikation.

Um die konzeptionelle Sicht einer Anwendungsdomäne beschreibend abbilden zu können, werden hierfür folgende Teilaspekte zur Ontologiespezifikation ermittelt, deren Synergie in Abbildung 6.5 zusammengestellt sind:

- beschreibende Spezifikation des Begriffs in Textform
- beschreibende Spezifikation von Merkmalen bzw. Attributen
- Mechanismen zur Begriffskategorisierung und Aufstellung von Klassifikationsstruktur

Komplementär zu Abbildung 6.5 ist festzuhalten, dass Begriffe nach DIN2330 sowohl die Begrifflichkeit als auch die Kategorie bilden.

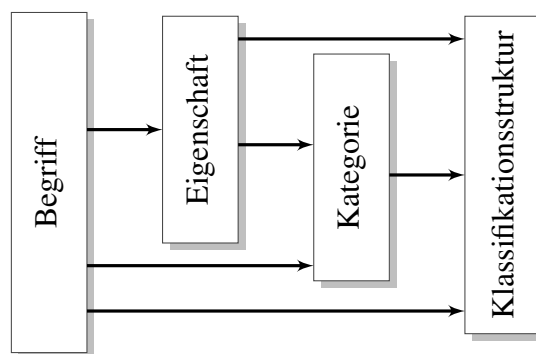


Abbildung 6.5.: Teilaspekte zur Spezifikation von ontologischen Konzepten

Im Folgenden werden zu den geforderten Abbildungsvorgängen die Lösungsmaßnahmen vorgestellt.

Darstellung von Begriffen im Sinne eindeutiger sprachgebräuchlicher Benennung

Zur Sicherstellung sprachlich eindeutiger Benennung werden im Rahmen der vorliegenden Arbeit folgende drei Elemente verwendet: Namen, Abkürzung und Definition:

- **Name:** Jeder Begriff erhält einen Namen, der in Abstimmung mit dem Kontext als sinnvoll, geläufig und ausdrucksstark erscheint. Dabei erfolgt die Namensvergabe gemäß internationaler Normen mit weitverbreiteten Ausdrücken.
- **Abkürzung:** Die Zuweisung einer Kurzbezeichnung zu einem Begriff soll auf eine Art erfolgen, dass keine Verwechslung mit dem Klassifikationsschlüssel entsteht.
- **Definition:** Eine in natürlicher Sprache zusammengefasste Beschreibung, die die Semantik des Terminus festlegt.

Hierzu ein Beispiel, angelehnt an das vorgestellte Anwendungsszenario:

Tripletbestandteil	Inhalt
Name	Bayerische Motoren Werke
Abkürzung	BMW
Definition	Ein deutscher Automobilhersteller mit Sitz in München

Abbildung der Mehrsprachigkeit

Das Ontologiemodell soll in der Lage sein, mehrsprachige Definitionen als unterschiedliche Repräsentationsformen derselben Begriffe abzubilden. Die Übersetzungen werden ähnlich zu XML-Syntax mithilfe des Attributs `xml:lang` mit Sprachangaben versehen. So kann das obige Beispiel folgendermaßen dargestellt werden:

Quelltext (Ausschnitt aus der Ontologie)
<pre><rdf:Description rdf:about="http://ecustom-project.eu/working-space"> <ex:Definition xml:lang="de"> Ein deutscher Automobilhersteller mit Sitz in München </ex:Definition> <ex:Definition xml:lang="en"> A German automobile manufacturer with headquarters in Munich </ex:Definition> </rdf:Description></pre>

Außerhalb eines XML-Dokuments werden Sprachinformationen mithilfe eines @-Zeichens dargestellt. In Turtle sieht das beispielsweise wie folgt aus:

Quelltext (Ausschnitt aus der Ontologie)
<pre><http://ecustom-project.eu/working-space> <http://ecustom-project.eu/working-space/Definition> "Ein deutscher Automobilhersteller mit Sitz in München"@de, "A German automobile manufacturer with headquarters in Munich"@en .</pre>

Nach dem Konzept der XML-Syntax haben alle Berücksichtigung gefundenen Sprachen einen eigenen Spracheintrag. Dieser Ansatz ist erweiterbar, sodass jederzeit neue Spracheinträge hinzugefügt werden können.

Abbildung von grundlegenden Klassifikationsbausteinen

Zur Informationskategorisierung und -strukturierung stellt die Klassifikation ein effizientes Mittel zur Verfügung. Dazu gehört jedoch auch die Festlegung von Beziehungen, denn diese bilden ebenfalls einen Bestandteil der Wissensressourcen und werden folglich als Begriffe behandelt. Diesbezüglich kann die Norm DIN 2331 als Referenz dienen, die nachfolgend aufgeführte Begriffsbeziehungstypen umfasst [DIN93]:

- Hierarchische Beziehung (Über- und Unterordnungsverhältnis)
- Abstraktionsbeziehung (Generalisierung: logische Beziehung, generische Beziehung)
- Bestandsbeziehung (Aggregation: Ganzes-Teil-Beziehung)
- Nachfolgebeziehung (Vorgänger–Nachfolger)
- Kausalbeziehung (Ursache–Wirkung)
- Genetische Beziehung (Hersteller–Produkt)
- Herstellerbeziehung (Material–Produkt)
- Transmissionsbeziehung (Sender–Empfänger)
- Instrumentelle Beziehung (Werkzeug–Anwendung des Werkzeuges)
- Funktionelle Beziehung (Argument–Funktion)

Basierend auf den implementierungstechnischen Restriktionen seitens des Wissensrepräsentationsformalismus \mathcal{L}_{tripod} (vgl. Abschnitt 5.2.2), werden in der Konzeptphase solche Begriffsbeziehungstypen im Voraus herausgenommen, die bezüglich ihrer Ausdruckstärke nicht umsetzbar sind. Ein Augenmerk wird auf die hierarchischen Taxonomien gelegt, da diese sich sehr gut zur Darstellung der Begriffsbeziehungstypen *Aggregation* und *Generalisierung* sowie Über- und Unterordnungsverhältnis eignen. Der Bedarf, Taxonomien im Ontologiemodell abbilden zu können, resultiert aus der hierarchischen Struktur des Domänenwissens in der Produktentwicklung.

Der Klassifikationsstrukturansatz wird mit den Konzepten *Kategorie* und *Klassifizierungsschlüssel* abgebildet. Dabei werden Kategorien als Begriffe behandelt und als zusätzliche semantische Metainformationen zur Kategoriebeschreibung (*Name*, *Abkürzung*, *Definition* Tupel) hinzugenommen.

Quelltext (Ausschnitt aus der Ontologie)

```
<owl:Class rdf:ID="Kategorie">
  <rdfs:subClassOf rdf:resource="#Thing"/>
  <rdfs:label>Kategorie</rdfs:label>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#Enthaeelt" />
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```


Die Abbildung des Begriffs Kategorie findet durch die Zuweisung eines Klassifizierungsschlüssels statt. Hierbei können Kategorien mehrere Klassifikationsschlüssel durch die inverse Beziehung `GehoertZu` zugewiesen werden.

Quelltext (Ausschnitt aus der Ontologie)

```
<owl:ObjectProperty rdf:ID="GehoertZu">
  <rdfs:domain rdf:resource="#Klassifizierungsschluessel"/>
  <rdfs:range rdf:resource="#Kategorie"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="Enthaelt">
  <owl:inverseOf rdf:resource="#GehoertZu"/>
```

Durch einen entsprechenden Klassifizierungsschlüssel können Kategorien innerhalb von Klassifikationen weiter untergliedert werden. Damit lassen sich Klassifizierungsschlüssel als gerichtete und zugleich zyklensfreie Graphen – Klassifikationsstrukturen – zusammenstellen. Die Klassifizierungsschlüssel vererben dabei Konzepteigenschaften den übergeordneten Konzepten, sodass in hierarchischer Kategorieaufstellung eine zunehmende Spezialisierung erfolgt. Dadurch wird die Verbalisierung von allgemeinen Konzepten auf abstrakter Ebene möglich.

Quelltext (Ausschnitt aus der Ontologie)

```
<owl:Class rdf:ID="Klassifizierungsschluessel">
  <rdfs:subClassOf rdf:resource="#Thing"/>
  <rdfs:label>Klassifizierungsschluessel</rdfs:label>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#GehoertZu" />
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Durch die hierarchische Strukturierung von Kategorien (d. h. taxonomischer Aufbau der Klassifizierungsschlüssel) werden Klassifikationssysteme gebildet.

④ Implementierung des Ontologiemodells

Die Implementierung des Ontologiemodells umfasst Aktivitäten, die das Bestücken der konzeptionellen Struktur mit dem Inhalt beabsichtigen, d. h. der *Materialisierung* globaler Ontologie. Dieser Begriff ist aus dem Bereich *Ontological Engineering* bekannt und bedeutet die Beschreibung aller Ontologieelemente durch eine konkrete formale Sprache [GPFLC04]. Dies kann manuell in einem Textverarbeitungsprogramm oder mithilfe einer Softwareanwendung

erfolgen. Auf die detaillierte Erläuterung dieses Teilprozesses soll im Folgenden nicht näher eingegangen werden, denn dieser Vorgang ist an den informationsmodellbasierten Softwareentwicklungsprozess angelehnt, wie er in der informationstechnischen Disziplin „Software Engineering“ vorkommt. Ausgehend von einem konzeptuellen Informationsmodell wird ein physisches Implementierungsmodell erstellt

Beschreibung und Klassifizierung von Produktinformationen

Produkte sind Artefakte, die Gegenstand einer Geschäftsbeziehung zwischen dem Produktanbieter und den Kunden sind. Dabei können Produkte physikalischer (z. B. technische Produkte) oder nicht physikalischer (z. B. Software) Natur sein. Im vorliegenden Ontologiemodell werden Produkte nicht definierend, wie im Fall eines traditionellen Produktmodells, sondern beschreibend charakterisiert. Das Konzept Produkt ist eine Spezialisierung des obersten Konzepts Thing. In der Sprache OWL ist die Klasse bzw. das Konzept owl:Thing vordefiniert und beinhaltet alle Klassen und Individuen der betrachteten Anwendungsdomäne. Somit ist sie die Oberklasse für alle Klassen einer OWL-Ontologie.

Quelltext (Ausschnitt aus der Ontologie)

```
<owl:Class rdf:ID="Produkt">
  <rdfs:subClassOf rdf:resource="#Thing"/>
  <rdfs:label>Produkt</rdfs:label>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#IstDefiniertIn" />
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#AnsprechPartner" />
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#IstGefertigtVon" />
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  ...
  ...
</owl:Class>
```

Den Produktinformationen entsprechend wird diese Wissensressource in der eCUSTOM-spezifischen

Klassifikationsstruktur in der Kategorie „Sachleistung“ klassifiziert. Durch die Betrachtung weiterer Produkt- bzw. Sachleistungsmerkmale kann die Beschreibung von Produktinformationen fortgeführt werden. Aufgrund der Zuordnung eines Produkts zu entsprechender Kategorie können andere Eigenschaften ermittelt und für die Beschreibung eingesetzt werden. Jedoch ist es empfehlenswert, den Eigenschaftenbestand vorab, d. h. bei der Definition der globalen Ontologie zu spezifizieren.

Als eine weitere Eigenschaft des Konzepts Produkt ist das Attribut „Ansprechpartner“ anzugeben. Dies kann zugleich als eine Beziehung zu weiteren Wissensressourcen verwendet werden. Zeitgleich ist zu prüfen, ob das Attribut mehrsprachig aufgeführt werden soll. Besteht der Bedarf, so wird der Ansprechpartner (engl. *contact person*) mit der Klasse Person definiert.

Beschreibung und Klassifizierung von Unternehmensinformationen

Um das Konzept „Unternehmen“ in globaler Ontologie zu beschreiben, werden Attribute „Name“, „Abkürzung“ und „Beschreibung“ verwendet. Im Rahmen des eCUSTOM-Projekts nehmen Unternehmen als Hersteller an einem Kollaborationssystem teil und sind in Zusammenhang mit dem jeweiligen Geschäftsbereich beschrieben.

Quelltext (Ausschnitt aus der Ontologie)

```
<owl:Class rdf:ID="Unternehmen">
  <rdfs:subClassOf rdf:resource="#Thing"/>
  <rdfs:label>Unternehmen</rdfs:label>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#FertigtProdukt" />
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#VertreibtProdukt" />
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#BietetAn" />
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  ...
  ...
</owl:Class>
```

Die Wissensressource „Unternehmen“ wird in der eCUSTOM-spezifischen Klassifikation in der Kategorie „Hersteller“ gemäß der Geschäftsbereichsklassifikation erfasst. Ferner kann mittels weiterer Merkmalspezifikation die Beschreibung des Geschäftsbereichs fortgeführt werden. Entsprechend der Kategorie, der das Konzept „Unternehmen“ zugeordnet wird, stehen Attribute wie „OEM“, „Zulieferer“ oder „Händler“ für weitere Beschreibungen zur Verfügung. Auch hier ist es empfehlenswert, den Eigenschaftenbestand vorab, d. h. bei der Definition der globalen Ontologie zu spezifizieren. Als weitere Eigenschaften des Konzepts Unternehmen können folgende Attribute – bei Bedarf auch mehrsprachig – angegeben werden: Organisation (engl. *Organization*), Geschäftsbereich (engl. *business unit*), Anschrift (engl. *address*), Internetpräsenz (engl. *homepage*), Telefon (engl. *phone*), E-Mail (engl. *email*), Telefax (engl. *fax*) und Sachkenntnisse (engl. *expertise*). Diese können zugleich als eine Beziehung zu weiteren Wissensressourcen verwendet werden.

Beschreibung und Klassifizierung von Dokumenten

Das Konzept „Dokument“ steht für alle physikalischen Ressourcen, die über Dokument-ähnliche Merkmale verfügen.

Quelltext (Ausschnitt aus der Ontologie)

```
<owl:Class rdf:ID="Dokument">
  <rdfs:subClassOf rdf:resource="#Thing"/>
  <rdfs:label>Dokument</rdfs:label>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#AnsprechPartner" />
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#Verfasser" />
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#DefiniertProdukt" />
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  ...
  ...
</owl:Class>
```

Zum Beispiel: digitale und gedruckte Informationsträger, Kataloge, Dateien, CD-ROM, Homepage-URL, digitales Bild- und Videomaterial etc. Um das Konzept „Dokument“ in globaler Ontologie zu beschreiben, werden die Attribute „Name“, „Abkürzung“ und „Beschreibung“ verwendet. Im Rahmen des eCUSTOM-Projekts kann das Attribut „Beschreibung“ eines Dokumentes technische Spezifikation, funktionale und nicht-funktionale Anforderungen sowie herstellerspezifische Angebotsangaben enthalten. Dem Dokumentinhalt entsprechend wird das Konzept „Dokument“ in die eCUSTOM-spezifischen Klassifikation z. B. in der Kategorie „Fahrzeughabue“ (engl. *car hood*) klassifiziert. Ferner kann mittels weiterer Merkmalspezifikationen die Beschreibung der Wissensressource fortgeführt werden. Entsprechend der Kategorie, der das Konzept „Dokument“ zugeordnet wird, können folgende Attribute – bei Bedarf auch mehrsprachig – angegeben werden: Auszug (engl. *abstract*), Internetpräsenz (engl. *homepage*), Quelle (engl. *source*), Format (engl. *format*), Version (engl. *version*), Urheber (engl. *author*), Verleger (engl. *publisher*), Ansprechpartner (engl. *contact person*). Diese können zugleich als eine Beziehung zu weiteren Wissensressourcen verwendet werden. Auch hier ist es empfehlenswert den Eigenschaftensbestand vorab, d. h. bei der Definition der globalen Ontologie zu spezifizieren.

Definition von Beziehungen zwischen Konzepten

Eine globale Ontologie definiert eine konzeptionelle Struktur sowie eine valide Begrifflichkeit zur Beschreibung von Informationen. Aufgrund der Anforderung, jegliche Terminologie unter der Berücksichtigung des im Abschnitt 5.2.2 definierten Wissensrepräsentationsformalismus \mathcal{L}_{tripod} abbilden zu können, sollen nach Möglichkeit alle in DIN 2331 spezifizierten Beziehungstypen unterstützt werden.

Quelltext (Ausschnitt aus der Ontologie)

```
<owl:ObjectProperty rdf:ID="IstDefiniertIn">
  <rdfs:domain rdf:resource="#Produkt"/>
  <rdfs:range rdf:resource="#Dokument"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="AnsprechPartner">
  <rdfs:domain rdf:resource="#Produkt"/>
  <rdfs:range rdf:resource="#Person"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="IstGefertigtVon">
  <rdfs:domain rdf:resource="#Produkt"/>
  <rdfs:range rdf:resource="#Unternehmen"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="FertigtProdukt">
  <owl:inverseOf rdf:resource="#IstGefertigtVon"/>
...
```

Ferner soll sichergestellt werden, dass nicht nur Beziehungen zwischen produktbeschreibenden Informationen untereinander in Verbindung stehen, sondern auch solche Wissensressourcen, die einen unterschiedlichen Beziehungstyp aufweisen.

Beziehungen zur Wissensressource Produkt:

- „wird produziert von“ (Unternehmen) / „is produced by“ (company)
- „wird beschrieben durch“ (Dokument) / „is described by“ (document)
- „wird informiert durch“ (Person) / „is informed by“ (person)

Beziehungen zur Wissensressource Unternehmen:

- „fertigt“ (Produkt) / „produces“ (product)
- „vermarktet“ (Produkt) / „markets“ (product)
- „anbietet“ (Produkt) / „offers“ (product)

Quelltext (Ausschnitt aus der Ontologie)

```
<owl:ObjectProperty rdf:ID="FertigtProdukt">
  <rdfs:domain rdf:resource="#Unternehmen"/>
  <rdfs:range rdf:resource="#Produkt"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="Vertreibt">
  <rdfs:domain rdf:resource="#Unternehmen"/>
  <rdfs:range rdf:resource="#Produkt"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="BietetAn">
  <rdfs:domain rdf:resource="#Unternehmen"/>
  <rdfs:range rdf:resource="#Produkt"/>
</owl:ObjectProperty>
...
```

Beziehungen zur Wissensressource Dokument:

- „wird verfasst von“ (Person) / „authored by“ (person)
- „wird herausgegeben von“ (Person) / „published by“ (person)
- „beschreibt“ (Produkt) / „describes“ (product)

Quelltext (Ausschnitt aus der Ontologie)

```

<owl:ObjectProperty rdf:ID="AnsprechPartner">
  <rdfs:domain rdf:resource="#Dokument"/>
  <rdfs:range rdf:resource="#Person"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="Verfasser">
  <rdfs:domain rdf:resource="#Dokument"/>
  <rdfs:range rdf:resource="#Person"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="DefiniertProdukt">
  <rdfs:domain rdf:resource="#Dokument"/>
  <rdfs:range rdf:resource="#Produkt"/>
  <owl:inverseOf rdf:resource="#IstDefiniertIn"/>
</owl:ObjectProperty>
...

```

6.5. Spezifikation datenquellenspezifischer Korrespondenzen

Eine für typische Anwendungsszenarien der Produktentwicklung relevante Wrapper-Technologie ist die Verknüpfung des Integrationssystems mit der relationalen Datenbanktechnologie. Dadurch kann das am Anfang dieses Kapitels vorgestellte Anwendungsszenario, bei dem die strukturierte Akkumulation größerer Informationsmengen erforderlich ist, die Funktionen leistungsfähiger relationaler Datenbanksysteme verwerten.

Eine Möglichkeit, zum Ontologiemodell konforme Informationen in relationalen Datenbanken speichern zu können, besteht darin, die \mathcal{L}_{tripod} TBox, in der die Metadaten vorliegen, auf ein relationales Datenbankschema abzubilden. Voraussetzung dafür ist, dass sich das jeweilige konzeptuelle Metamodell (\mathcal{L}_{tripod} TBox) und das relationale Modell in etwa entsprechen. Betrachtet man das Ontologiemodell aus Abschnitt 6.4.2, so zeigt sich, dass sich die grundlegenden Konstrukte auf das relationale Modell und die Datenbanksprache SQL abbilden lassen. Konzepte bzw. OWL-Klassen korrespondieren mit Tabellen, Rollen bzw. OWL-Eigenschaften je nach Kardinalität mit Fremdschlüsselbeziehungen oder eigenen Tabellen. Ferner können Konzepte und Rollen auf zusätzliche Spalten der jeweiligen Tabelle abgebildet werden.

Die instanziierte Produktinformation in der Sprache beschreibungslogischer Wissensbasen, sogenannte ABox, wird physisch in den Datenbanktabellen gespeichert. Da die Informationsanfragen auf Basis der TBox gestellt werden (die Abbildung auf das relationale Modell ist eben nur ein Implementierungsdetail), wird eine zur globalen Ontologie konforme Sicht mithilfe von Abbildungsvorschriften (Korrespondenzen) erzeugt.

Dieser Ansatz hat sich in vielen Softwareprodukten insofern bewährt, als er einerseits die Funk-

tionalität relationaler Datenbanksysteme vollständig ausnutzt, andererseits eine Schnittstelle anbietet, die objektorientierten Prinzipien entspricht.

Im vorliegenden Integrationssystem werden jedoch die Korrespondenzen nicht als Basiskomponente, sondern als austauschbarer Wrapper in der Rolle eines Adapters betrachtet und realisiert. Deshalb wird ein festes, implementierendes relationales Datenbankschema auf Seiten der Datenquellen vorausgesetzt. Dazu werden also Abstraktionen der Metamodellebene globaler Ontologie mittels Korrespondenzen auf die Elemente der Datenmodellebene implementierungstechnisch umgesetzt. Sie können zur Laufzeit von dem Mapping Modul (vgl. Abschnitt 5.4.4) gelesen werden. Dadurch müssen die Korrespondenzen zwischen der TBox globaler Ontologie und der Schemata der Datenquelle nicht vollständig sein, d. h. sie müssen nicht für jedes Konzept oder für die Rolle der TBox definiert werden. Auf diese Weise beeinflussen Änderungen des semantischen Modells das relationale Datenbankschema nicht.

6.5.1. Wrapper für relationale Datenbanken

Der im Rahmen des eCUSTOM-Projekts realisierte Wrapper stellt eine bedeutende Komponente für die auf der Grundlage des Integrationssystems umgesetzten informationstechnischen Lösungen zur Verfügung. Die Koppelung von relationalen, im speziellen SQL-basierten Datenbanken, kann – im Gegensatz zu dem J2EE¹⁰ -Umfeld – von allen für das bestimmte Datenbanksystem zugänglichen Optimierungsmechanismen profitieren. Dies liegt darin begründet, dass nicht das Integrationssystem selbst die SQL-Anfragen erzeugt und an die betreffende Datenbank weiterleitet, sondern dass zur Koppelung eines relationalen Datenbanksystems ein entsprechender Wrapper implementiert wird (siehe Abb. 6.6).

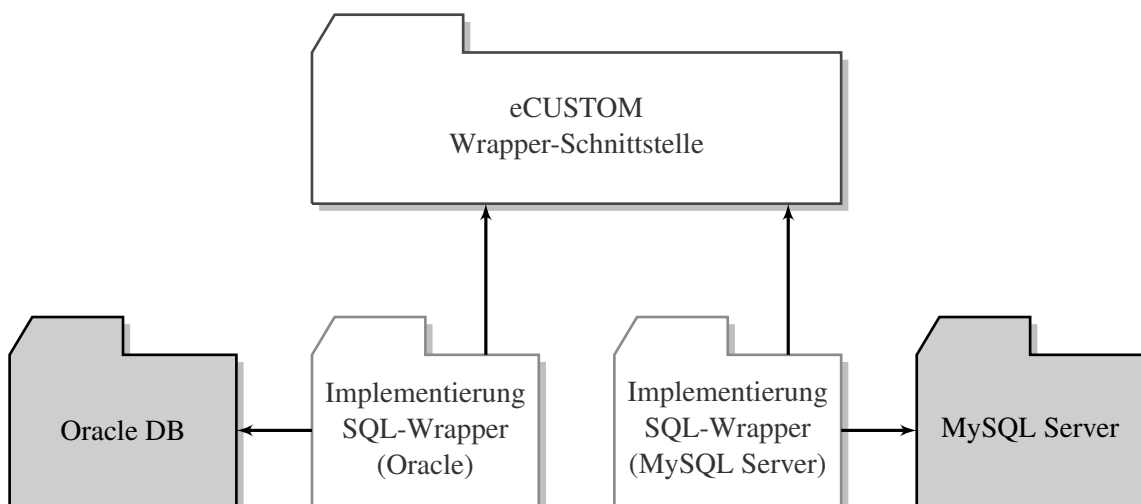


Abbildung 6.6.: SQL-Wrapper

¹⁰<http://java.sun.com/j2ee/overview.html>

Das führt dazu, dass für diverse Datenbanksysteme unterschiedliche Wrapper realisiert werden können, die von der jeweiligen Datenbank zur Verfügung gestellte Optimierungsmöglichkeiten zur Performanzsteigerung nutzen. Um die datenbanknahen Berechnungen vorzubereiten, verwendet das im Rahmen der vorliegenden Arbeit entwickelte Wrapper für MySQL-Server die Funktionalität des Prekompilierens von Informationsbereitstellungsanfragen als sogenannte *stored procedures*. Dadurch kann eine Leistungssteigerung erzielt werden, indem die Prozedur auf dem Datenbankserver gespeichert wird und somit der Datentransfer zwischen dem Aufrufer und der Datenbank verringert wird. Darüber hinaus führt der Einsatz von stored procedure zu einer größeren Plattform- und Programmiersprachenunabhängigkeit. Die Erstellung einer stored procedure unter MySQL 5 erfolgt durch das SQL-Statement CREATE PROCEDURE und hat folgenden Aufbau:

Quelltext (Ausschnitt zur Erstellung von *Stored Procedures*)

```
CREATE PROCEDURE procedure_name ([parameter]) [characteristics]
BEGIN
    /*... Anweisungsblock ...*/
END
```

Der Vorteil dieses Ansatzes besteht vor allem darin, dass er von der tatsächlichen Umsetzung eines relationalen Datenbankverwaltungssystems durch die Verwendung von SQL-Anfragen abstrahiert und damit eine Leistungssteigerung auf der Grundlage komponentenbasierten Optimierungsmechanismen herbeiführt.

6.5.2. Implementierung der Korrespondenzen

In den vorausgegangenen Abschnitten wurde die Spezifikation der Korrespondenzen durch einen datenbankspezifischen Wrapper auf der Menge der Abstraktionen globaler Ontologie veranschaulicht. Was dem Theoretiker womöglich bereits genügt, wirft spätestens in der Phase der prototypischen Realisierung die Frage nach einer softwaretechnischen Umsetzung auf.

Im Folgenden wird die Funktionsweise der Korrespondenz basierenden Verknüpfung am Beispiel des eCUSTOM-Systems vorgestellt. Aus Übersichtlichkeitsgründen wird für die weitere Darstellung nur ein Auszug des globalen Ontologiemodells verwendet. Hierbei enthält Ontologie die Informationen über Produkte und Hersteller mit ihren Bezeichnungen sowie den Typ der Hersteller (OEM, Zulieferer, Händler) hinsichtlich des gefertigten Produkts.

Quelltext (Ausschnitt aus der Ontologie)

```
<!-- Classes -->

<!-- http://www.imi.kit.edu/43_126.php/ontologies/2012/5/eCUSTOM.owl#Hersteller -->
  <owl:Class rdf:about="&eCUSTOM;Hersteller"/>

<!-- http://www.imi.kit.edu/43_126.php/ontologies/2012/5/eCUSTOM.owl#Produkt -->
  <owl:Class rdf:about="&eCUSTOM;Produkt"/>

<!-- Object Properties -->
  <owl:ObjectProperty rdf:about="&eCUSTOM;WirdHergestellt"/>
  <owl:ObjectProperty rdf:about="&eCUSTOM;fertigt">
    <rdfs:domain rdf:resource="&eCUSTOM;Hersteller"/>
    <rdfs:range rdf:resource="&eCUSTOM;Produkt"/>
    <owl:inverseOf rdf:resource="&eCUSTOM;WirdHergestellt"/>
  </owl:ObjectProperty>
```

Integriert werden sollen die in Tabelle 6.1 repräsentierten Datenquellen. Dabei verfügt jede Datenquelle über ein Schema mit jeweils drei Tabellen, die intensional exakt den drei Abstraktionen des semantischen Modells (globale Ontologie) entsprechen. Im Allgemeinen wird nicht jedes globale Konzept oder jede Rolle genau einer Tabelle (Relation) einer Datenquelle entsprechen. Wir begnügen uns in diesem Fall mit einem einfachen Beispiel. Darüber hinaus müssen die Korrespondenzen, wie bereits im Abschnitt 6.5 erläutert wurde, nicht vollständig bezüglich zugrundeliegender Schemata sein. Im Folgenden wird jede Datenbankrelation als eine eigene Datenquelle betrachtet (siehe Tab. 6.1).

Datenquelle und Datenbanktabelle	Beschreibung	globale Abstraktion
<code>imidb.produkt(name, kategorie)</code>	Alle Produkte der IMIDB	Produkt
<code>imidb.produziert(name, produzent_name, typ)</code>	Zuordnung von Herstellern zu Produkten der IMIDB	fertigt
<code>imidb.produzent(produzent_name, level)</code>	Alle Hersteller der IMIDB	Hersteller
<code>lmsdb.product(name, product_category)</code>	Alle Produkte der LMSDB	Produkt
<code>lmsdb.produce(name, manufacturer_name, type)</code>	Zuordnung von Herstellern zu Produkten der LMSDB	fertigt
<code>lmsdb.manufacturer(name, manufacturer_level)</code>	Alle Hersteller der LMSDB	Hersteller

Tabelle 6.1.: Zwei einfache Datenquellen für eine eCUSTOM-Produktontologie

Zu den gegebenen Datenquellen bzw. deren Relationen lassen sich folgende Mapping-Beziehungen (Korrespondenzen) aufstellen (siehe Tab. 6.2):

Kopf der Mapping-Beziehung	Körper der Mapping-Beziehung
Produkt	<pre>SELECT name, kategorie FROM imidb.produkt;</pre> <pre>SELECT name, product_category FROM lmsdb.product;</pre>
fertigt	<pre>SELECT name, produzent_name, typ FROM imidb.produziert;</pre> <pre>SELECT name, manufacturer_name, type FROM lmsdb.produce;</pre>
Hersteller	<pre>SELECT produzent_name, level FROM imidb.produzent;</pre> <pre>SELECT name, manufacturer_level FROM lmsdb.manufacturer;</pre>

Tabelle 6.2.: Mapping-Beziehungen zwischen eCUSTOM-Produktontologie und entsprechenden Datenquellen

Das Integrationssystem soll nun beispielsweise eine Anfrage nach allen Produkten und deren Kategorien beantworten, bei denen der Hersteller „*alfa-mimtech*“¹¹ ein Zulieferer ist.

In diesem Fall existieren für jede der zwei in der Anfrage vorkommenden globalen Abstraktionen des Ontologiemodells jeweils zwei mögliche Datenquellen. Da die Datenquellen über den Herstellernamen verknüpft werden können, ergeben sich vier unterschiedliche Kombinationen, die u. a. unterschiedliche Ergebnisse liefern können. Dies ist nicht zuletzt dadurch begründet, dass die Datenquellen vollständig autonom sind und daher unterschiedliche Datensätze enthalten können. Die vier möglichen Kombinationen sind in Form von lokalen Anfragen in Tabelle 6.3 repräsentiert.

¹¹Das Unternehmen alfa-mimtech ist einer der Partner des EU-Projektes eCUSTOM. Alfa-mimtech verfügt über einen neuartigen Fertigungsprozess, der die Flexibilität und hohe Produktivität des Kunststoffspritzgusses mit den mechanischen Eigenschaften metallischer Werkstoffe kombiniert. <http://www.mimtech-alfa.com>

Kombination	Lokale SQL-Anfrage
K_1	<pre>SELECT name, kategorie, typ FROM imidb.produkt, imidb.produziert WHERE imidb.produziert.produzent_name = 'alfa-mimtech' AND imidb.produziert.typ = 'zulieferer' AND imidb.produziert.name = imidb.produkt.name;</pre>
K_2	<pre>SELECT name, kategorie, type FROM imidb.produkt, lmsdb.produce WHERE lmsdb.produce.manufacturer_name = 'alfa-mimtech' AND lmsdb.produce.type = 'supplier' AND lmsdb.produce.name = imidb.produkt.name;</pre>
K_3	<pre>SELECT name, product_category, typ FROM lmsdb.product, imidb.produziert WHERE imidb.produziert.produzent_name = 'alfa-mimtech' AND imidb.produziert.typ = 'zulieferer' AND lmsdb.product.name = imidb.produziert.name;</pre>
K_4	<pre>SELECT name, product_category, type FROM lmsdb.product, lmsdb.produce WHERE lmsdb.produce.manufacturer_name = 'alfa-mimtech' AND lmsdb.produce.type = 'supplier' AND lmsdb.product.name = lmsdb.produce.name;</pre>

Tabelle 6.3.: Unterschiedliche Kombinationen der lokalen Anfragen

Darauf basierend kann eine Kombination ausgewählt werden, die z. B. die meisten Ergebnisse liefert. Alternativ dazu können ebenfalls alle Kombinationen ausgeführt werden, um eine möglichst vollständige Antwort zu berechnen.

Anschließend müssen die von den jeweiligen Kombinationen erzeugten Resultate zu einem für den Benutzer leicht verständlichen Gesamtergebnis zusammengeführt werden. Dadurch, dass alle logischen und strukturellen Hürden bereits überwunden sind, entspricht dies einer reinen Datenfusion. Eine gute Übersicht über das Gebiet der Datenfusion bieten [FS69], [SCS03].

6.6. Vorgehensweise zur ontologiebasierten Informationssuche

Das Verfahren zur Informationssuche beschreibt den Zugriff auf Wissensressourcen, bei dem die Suchanfrage sowie die Ergebnisdarstellung dem vorliegenden Anwenderkontext entsprechen. Damit wird sichergestellt, dass der Anwender genau auf seinen Spezifikationen entsprechende Wissensressourcen zugreifen kann. Im Fall, dass der Suchanfrage entsprechend kein

adäquates Wissen vorliegt, liefert das System nach kurzer Zeit ein Ergebnis ohne lange und Ressourcen verschwendend zu suchen.

Auf diese Weise wird die ontologiebasierte Suche nach Wissen in einer verteilten Umgebung auf einer semantisch höheren Ebene als bisher durchgeführt. Das nachfolgend beschriebene Verfahren ist gleichermaßen skalierbar und einsetzbar in unternehmensinternen Infrastrukturen sowie in global agierenden Kollaborationsnetzwerken.

Aufgabenstellung für die Informationssuche: *Beim mechanischen Entwurf der Motorhaube (engl. car hood) soll sowohl auf externe Zulieferer als auch auf unternehmensinterne Informationsquellen transparent zugegriffen werden, um ein technisches sowie wirtschaftliches Optimum zu erzielen. Zur Lösung des vorliegenden Entwicklungsproblems wird ein Hersteller von Scharnierträger (engl. hinge support) benötigt. Die Aufgabe besteht nun in der Suche in externen Informationsquellen nach dem konkreten Hersteller, der das Produkt „Scharnierträger“ herstellt.*

Der Suchvorgang erfolgt in zwei Schritten: Zunächst wird die Benutzeranfrage gemäß dem in Abschnitt 5.4.2 vorgestellten Algorithmus umgeschrieben, um relevante Abstraktionen (Konzepte und Rollen) globaler Ontologie ausfindig zu machen, die die gesuchten Wissensressourcen beinhalten können. Dann wird auf Basis des Korrespondenzvorrats die umgeschriebene Anfrage in eine SQL-Anfrage überführt und an die relationalen Datenquellen zur Auswertung gesendet.

Bevor eine Datenquelle für die Integration und damit auch für den Suchvorgang eingesetzt werden kann, muss sie beim Integrationssystem registriert werden. Zur Registrierung und eindeutigen Identifikation kann die Datenquelle mittels einer Maske mit einer Benutzeridentifikation (ID) und einem Passwort angemeldet werden. In Abbildung 6.7 ist das dazugehörige Registrierungsfenster des Integrationssystems dargestellt. Sollte eine Datenquelle im Fenster „Registrierte Datenbanken“ noch nicht aufgeführt sein, so kann sie anhand der Datenquellenverwaltungsfunktion durch die Schaltfläche „Neu“ hinzugefügt werden. Es sei nun angenommen, dass entsprechende Eintragungen in Form von neuen Datenquellen vorgenommen worden sind.



Abbildung 6.7.: Registrierungs Fenster für Datenquellen

In der in Abbildung 6.8 dargestellten Benutzerschnittstelle formuliert der Anwender an der mit ① markierten Stelle seinen Informationsbedarf in Form einer Suchanfrage an das semantische Informationsintegrationssystem. Hierzu bietet die als Baumstruktur dargestellte Ontologie ② dem Anwender eine Hilfestellung an, um seine Anfrage zusammenfassen zu können. Die Ontologierepräsentation ist in zwei große Gruppen aufgeteilt: An der obersten Stelle sind Klassen bzw. Konzepte in Form einer Taxonomie abgebildet; direkt darunter befinden sich Relationen bzw. Rollen. Die Suche wird mit dem Kommando „Search“ gestartet ③. Das Integrationssystem nimmt die Suchanfrage entgegen und schreibt diese basierend auf den in der globalen Ontologie enthaltenen Axiomen um. An der mit ④ gekennzeichneten Stelle wird die umgeschriebene Anfrage repräsentiert.

Entspricht die umgeschriebene Suchanfrage den im Integrationssystem vorhandenen Korrespondenzen, so kann diese in SQL-Anfrage transformiert werden. Der mit ⑤ markierte Bereich führt zur umgeschriebenen Anfrage konforme Korrespondenzen auf. Basierend darauf wird jedes Konzept und jede Rolle der umgeschriebenen Anfrage durch entsprechende Korrespondenz ersetzt und eine Vereinigung der resultierenden Anfragen gebildet. Es ist zu beachten, dass nicht zu jedem Konzept bzw. jeder Rolle globaler Ontologie eine Korrespondenz existieren muss. Daher werden solche Abstraktionen globaler Ontologie zu denen es keine Korrespondenzen gibt, nicht berücksichtigt. Der SW-Prototyp stellt die Möglichkeit bereit, weitere Korrespondenzen in das System einzupflegen.

6. Realisierung und Validierung des Ansatzes

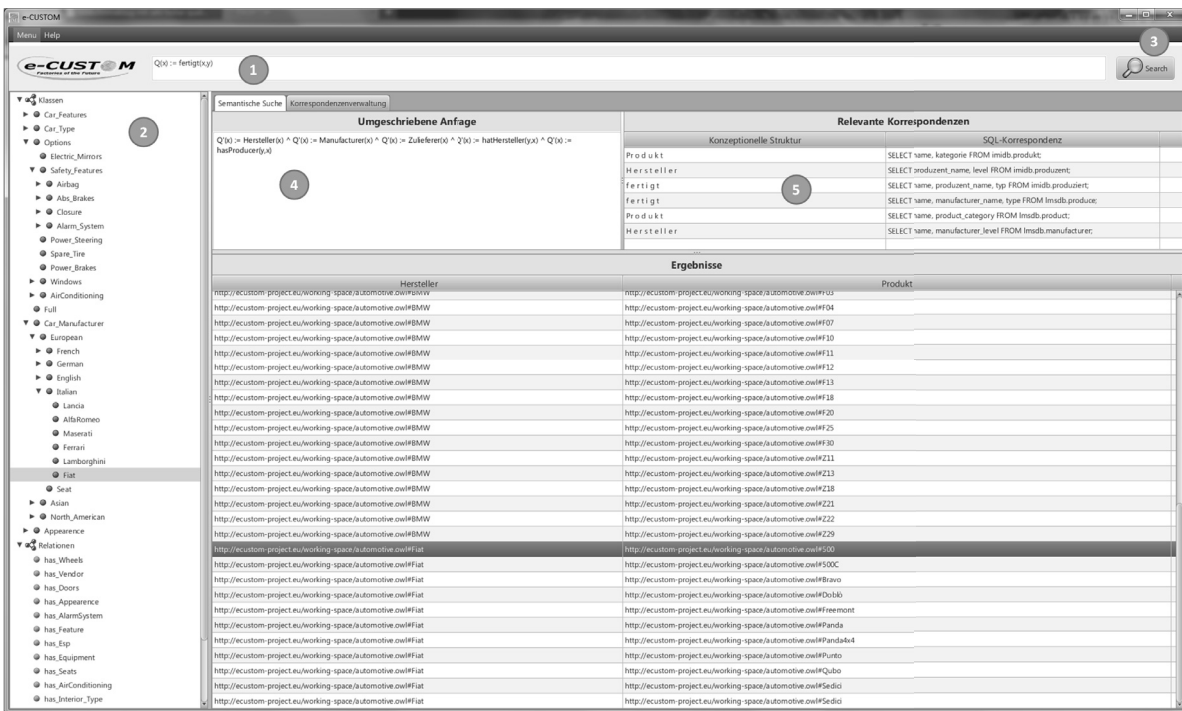


Abbildung 6.8.: Das Hauptfenster der Benutzerschnittstelle

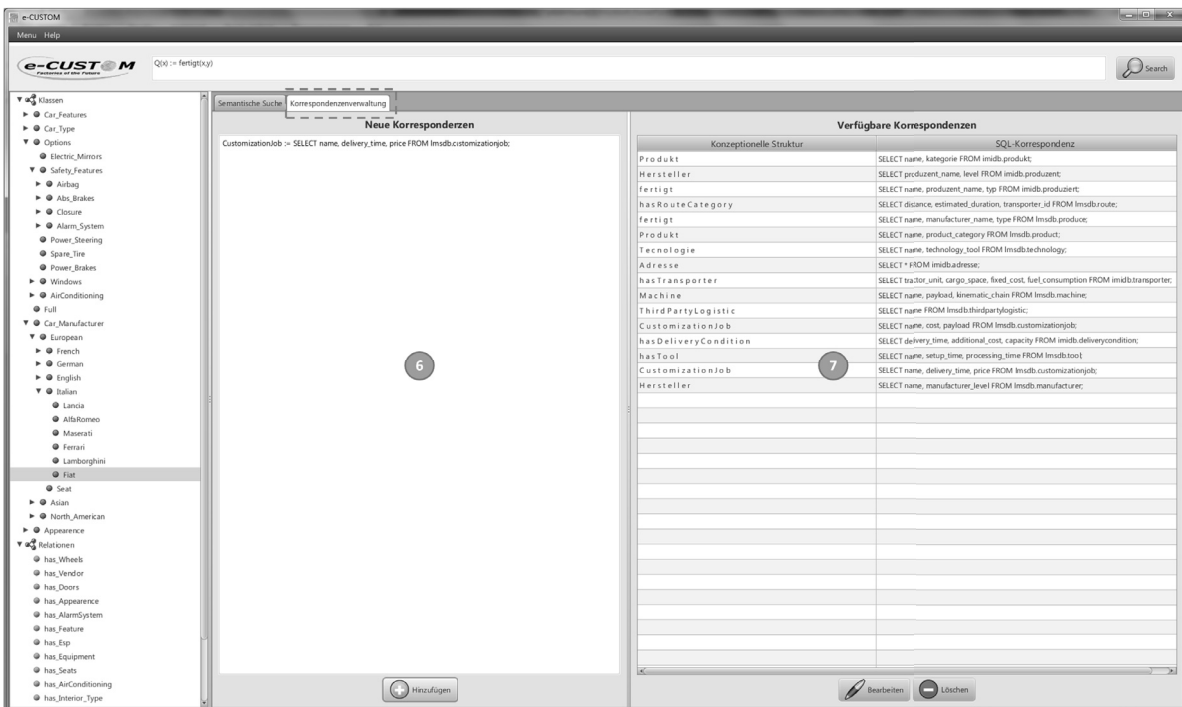


Abbildung 6.9.: Korrespondenzenverwaltung

Hierzu erfolgt die Verwaltung über den Karteireiter „Korrespondenzenverwaltung“. Im rechten Bereich ⑥ können neue Korrespondenzen eingetragen und über die Schaltfläche „Hinzufügen“ dem System hinzugefügt werden. Der Zugriff auf bestehende Korrespondenzen zwecks einer Modifizierung oder gar Entfernung erfolgt im mit ⑦ gekennzeichneten Bereich (Abb. 6.9). Folglich werden SQL-Anfragen an die betreffenden Wrapper delegiert, die dann auf der Grundlage ihrer lokalen Informationen datenquellenspezifische Anfragen vorbereiten. Anschließend werden Anfragen von entsprechenden Datenbanksystemen ausgeführt und das Ergebnis an das Integrationssystem zurückgeliefert.

Die Suchergebnisse werden in tabellarischer Form dargestellt. Durch Doppelklick auf Tabelleninhalte können weiterführende Detailinformationen zu ausgewählten Wissensressourcen in separatem Fenster dargestellt werden (Abb. 6.10).

The screenshot shows the e-CUSTOM application interface. On the left is a class hierarchy tree. The main area is divided into three sections: 'Semantische Suche' with a query $Q(x) = \text{fertigt}(x)$, 'Umgeschriebene Anfrage' with a complex query involving manufacturer and product relationships, and 'Relevante Korrespondenzen' with a table of SQL queries. Below these is a table of search results for 'Hersteller' with columns for manufacturer name and URL. On the right, a 'Details' window shows metadata for a Fiat 500 car, including manufacturer, product type, variant, weight, year, class, color, number of doors, length, width, height, and body style.

Abbildung 6.10.: Ergebnisdarstellung

An diesem Validierungsbeispiel konnte der Vorteil aufgezeigt werden, dass einerseits durch eine ontologiebasierte Wissensbasis, andererseits durch Inferenzmechanismus auf Basis von Umschreibalgorithmus der Suchraum im Kontext des Anwenders sinnvoll erweitert werden kann.

6.7. Zusammenfassung

In den hier beschriebenen Konzeptvalidierungen wurde exemplarisch aufgezeigt, welche Funktionalitäten ein auf Beschreibungslogik basierendes System zur Informationsintegration und semantischer Suche enthalten muss und welche Vorgehensweise hierfür notwendig ist. Die Aufgabenstellung einer unternehmensübergreifenden Kollaboration zeigt dabei, dass speziell beim Suchen in komplexen Informationssystemen wie in der Produktentwicklung, man Wissen über den Aufbau und Organisation der einzelnen Systeme benötigt. Genau hier liegt das Haupthindernis der Informationsintegration und damit auch der Suche, weil die Struktur von historisch gewachsenen Systemen nicht offensichtlich ist, sondern einer Interpretation bedarf. Daher werden an dieser Stelle, im Gegensatz zu einfachen syntaktischen Suchmethoden die Vorteile einer semantischen Suche vorgezogen.

Der Hauptvorteil ist, dass man nur wissen muss, *was* man suchen will und nicht, wo es zu finden ist. Der dafür notwendige Aufwand zur Erstellung einer initialen Menge an konzeptuellen Strukturen eines Gegenstandsbereichs sowie entsprechenden Abbildungsvorschriften zu den zugrundeliegenden Datenquellen zahlt sich mit zunehmender Nutzungsdauer des Integrationsystems aus. Der Mehrwert des Systems wird größer, da entweder durch die automatische oder die manuelle Anreicherung immer mehr Wissen in der Wissensbasis hinterlegt wird. Basierend darauf können bereits bei einer geringen Menge an konzeptuellen Strukturen deutlich bessere Suchergebnisse als mit herkömmlichen Methoden erzielt werden.

*Jeder Tag bringt Fragen und verlangt
Antworten, und solange wir fragen und
Antworten suchen, leben wir.*

ERWIN SCHRITTMATTER (1912–1994)

7. Schlussbemerkung

In diesem Kapitel sollen der Inhalt und Nutzen beschrieben und damit das Neuartige an dieser Arbeit hervorgehoben werden. Abschließend soll ein Ausblick auf noch anzugehende Problemstellungen gegeben werden.

7.1. Inhalt der Arbeit

Unternehmen, die im Zeitalter der Globalisierung ihre Wettbewerbsfähigkeit sichern und ausbauen wollen, müssen in der Lage sein, innovative und individuell auf die Zielmärkte zugeschnittene Produkte in immer kürzeren Zeitintervallen entwickeln, herstellen und vertreiben. Die zu diesem Zweck zahlreich eingesetzten Informationssysteme stellen höhere Anforderungen an Integrationslösungen als klassische Schnittstellenimplementierung oder Standarddatenmodelle. Komplexe Informationsstrukturen müssen unterstützt und den sich dynamisch ändernden Anforderungen mit vertretbarem Aufwand angepasst werden. Darüber hinaus sind kreative Arbeitsaktivitäten, beispielsweise bei der Entwicklung neuer Produktvarianten oder beim Kommunizieren neuer Designideen durch die Informationsintegration zu verbessern.

Aufgrund der steigenden Produktkomplexität, der sinkenden Halbwertzeiten von Informationen, der Notwendigkeit, unkontrolliert aufkommende Datenfluten zu beherrschen sowie der weltweiten Verteilung der Entwicklungsstandorte und Marktglobalisierung resultiert der dringende Handlungsbedarf, skalierbare Informationsintegrationslösungen zu finden, die einen effizienten und kontextbezogenen Zugriff auf Wissen unterstützen, der exakt den Anwenderbedürfnissen entspricht.

Eine Analyse unterschiedlicher Lösungsansätze zur Informationsintegration ergab, dass sowohl die Formalisierung von Wissen bzw. die Repräsentation vom konzeptuellen Wissen als auch der Zugriff auf vorhandene, aber ubiquitär verbreitete Informationen durch die einzelnen Ansätze nicht oder nur teilweise abgedeckt ist. So bieten gegenwärtige unternehmensintern eingesetzte Systeme keine Kommunikationsmechanismen an, die ein gemeinsames semantisches Verständnis der Sachverhalte ermöglichen. Des Weiteren wurde festgestellt, dass existierende Integrationssysteme in der Regel die unternehmensinternen Anforderungen nicht berücksichtigen, da sie naturgemäß nicht auf das Produktspektrum eines einzelnen Unternehmens angepasst sind. Ferner stellen viele Ansätze keine Möglichkeit zur Verfügung, eine Suchanfrage in Form einer benutzerzentrischen Sicht auf Basis eines benutzerspezifischen Vokabulars zu erstellen. Gegen-

stand der vorliegenden Arbeit ist es daher, Werkzeug in die Hand des Ingenieurs zu geben, das zum einen ein solches gemeinsames Vokabular bereitstellt, zum anderen – darauf aufbauend – die Möglichkeit einräumt, konzeptuelles Wissen in Form von beschreibungslogischen Formalismen, die eine Anwendungsdomäne repräsentieren, abzubilden. Darüber hinaus erhält der Anwender in der softwaretechnischen Umsetzung einen Prototyp, der es ermöglicht, kontextorientierte Suchanfragen, die sich sehr nahe an der menschlichen Denkweise orientieren, an das Integrationssystem zu stellen. Basierend auf dem gemeinsamen Verständnis über die verwendeten Begriffe wird die Suchanfrage mithilfe logischer Inferenzmechanismen evaluiert und an die integrierten Datenbestände in Form von datenquellenspezifischen Anfragen weitergeleitet. Um Wissen identifizieren und letztendlich auch nutzen zu können, werden Ergebnisse in einem einheitlichen Rahmen organisiert und dem Benutzer zugänglich gemacht.

Dem entwickelten Lösungsansatz liegt die Idee zugrunde, dass durch Repräsentation und semantische Verknüpfung komplexer Zusammenhänge, in Form einer globalen Ontologie sämtliche Kontextinformationen aus einer Vielzahl von Informationssystemen durch eine gemeinsame Begrifflichkeit modelliert und in einer dem Anwender vertrauten Art zugreifbar gemacht werden. Dafür findet in dieser Arbeit *SHOIN* ihren Einsatz, die einerseits eine sehr ausdrucksstarke Beschreibungslogik und die signifikante Grundlage für den W3C -Standardsprache OWL darstellt, andererseits aber entscheidbar ist. Die Entwicklung geeigneter Umschreiberalgorithmen zur automatisierten Auswertung benutzerspezifischer Suchanfragen in Form von logischer Inferenzmechanismen wird in der Nutzungsphase am Beispiel einer ontologiebasierten Anwendung evaluiert.

Das Neuartige an dieser Arbeit ist demnach:

- die formale Repräsentation komplexer Zusammenhänge aus dem Bereich Produktentwicklung durch einen – im Vergleich zu herkömmlichen Verfahren und Techniken – grundlegend anderen Ansatz,
- die Herleitung und Entwicklung von Algorithmen und Werkzeugen zur Unterstützung der Interoperabilität verteilter und heterogener Datenquellen in der Produktentwicklung,
- die Integration von wissensbasierten Ansätzen in relational-orientierten Informationssystemen. Dies wurde am Beispiel der semantischen Suche im Rahmen eines Forschungsprojekts gezeigt.

7.2. Nutzen der Arbeit

Der Nutzen der vorliegenden Arbeit liegt darin, dass die Integration von in der Produktentwicklung und darüber hinaus vorhandener Informationssysteme *einfacher*, d.h. schneller und

kostengünstiger wird. Durch den Einsatz konzeptueller Strukturen in Form einer globalen Ontologie erhöht sich die Entwicklungsproduktivität, da diese Strukturen nicht jedes Mal von Grund auf neu entwickelt werden müssen, sondern bereits bestehende erweitert bzw. wiederverwendet werden können. Dadurch, dass die zugrunde liegende Sprache \mathcal{L}_{tripod} der Ontologie theoriefundiert ist, sind die konzeptuellen Strukturen *zuverlässiger* und *robuster*.

Ein weiterer Nutzen liegt darin, dass die neu entwickelten Inferenzalgorithmen unabhängig von der Vollständigkeit der Kontextinformationen sind, d. h., dass sie trotz fehlender Informationen keine falschen Ableitungen vornehmen. Dies ist darin begründet, dass die verwendeten Beschreibungslogiken zur Formalisierung der Semantik von der *open world assumption* ausgehen und somit *unvollständige* Informationen behandeln können. Das Fehlen von Informationen führt nicht zu falschen Schlussfolgerungen, sondern es werden mithilfe von erarbeiteten Algorithmen bestmögliche Folgerungsantworten produziert. Mit dem entwickelten Ansatz wird eine semantische Integration unterstützt, die aufgrund der theoretischen Grundlagen nicht nur auf die momentanen Gegebenheiten der relationalen Datenquellen eingeht, sondern er kann – im idealen Fall – konsistent um weitere Quellen erweitert bzw. angepasst werden.

Ein verteilter Einsatz eines semantischen Integrationskonzepts in der Produktentwicklung erweitert den Wissensbeschaffungsraum des Ingenieurs enorm und bietet eine drastische Reduzierung bei der Wissensbeschaffung. Anstelle von schlüsselwortbezogenen Suchanfragen können kontextbezogene Problem- oder Aufgabenstellungen formuliert werden, ohne dass die Lösungswörter, nach denen in herkömmlichen Suchsystemen gesucht wird, antizipiert werden. Ubiquitär vorhandene Lösungen in Form von Zulieferteil-, Produktkomponenten-, Unternehmensinformationen oder Kompetenzträger können dem gemeinsamen Verständnis für Sachverhalte des Ingenieurs zugreifbar gemacht werden. Dadurch kann einerseits die Lösungsfindung erheblich unterstützt, andererseits der Innovationsgrad der zu entwickelnden Produkte gesteigert werden. Somit wird die Arbeitseffizienz des Ingenieurs erhöht, wodurch wiederum die Arbeitsprozesse beschleunigt werden können.

Das entwickelte Lösungsansatz- und Systemkonzept wurde im Rahmen einer Prototypimplementierung realisiert. Damit wurde eine durchgängige semantische Informationsintegration in der Produktentwicklung umgesetzt. Die Möglichkeit, zusammen mit den konzeptuellen Strukturen auch das Informationsmodell der jeweiligen Datenquelle in Beziehung zu setzen, vereinfacht nicht nur die Entwicklung von Integrationslösungen, sie vermeidet zudem den Informationsverlust durch semantisches Mapping und erhöht den Nutzwert der integrierten Datenquellen. Im Vordergrund steht neben dem Entwurf der semantischen Informationsintegration besonders deren effiziente Nutzung im Rahmen von Produktentwicklungsprojekten für *Mass Cu-*

stomization. Hierbei spielt die ganzheitliche Konzeption von konzeptuellen Strukturen in dem eCUSTOM-Kollaborationssystem eine entscheidende Rolle. Die Architektur des Systems stellt sicher, dass die auf Basis des mathematischen Modells entwickelten Module sowohl funktional unterstützt werden als auch durch Korrespondenzen auf geeignete relationale Strukturen abgebildet werden können.

Der Mehrwert für den Ingenieur wurde im Rahmen des Forschungsprojekts eCUSTOM durch die Anwendung auf den kontextbezogenen Wissenszugriff demonstriert. Im einzelnen ergeben sich daraus folgende Nutzen:

- Verkürzung der Entwicklungszeit für personalisierte Produkte bis zu 15 %
- Ausbau des Marktanteils um bis zu 10 %
- Reduzierung der Zeit bis zur Markteinführung (engl. *time to market*) um 15 %
- Reduzierung der Lieferzeit von ca. 15 % bis 20 %

Die Umsetzung eines Konzepts, wie es in dieser Arbeit vorgestellt wurde, stellt eine Grundlage für Unternehmen dar, wenn sie zukünftig innovative und personalisierte Produkte in einer Zusammenarbeit mit anderen Unternehmen herstellen wollen und dazu auf Basis einer einheitlichen Kommunikationsebene effizient auf Informationen zugreifen müssen.

7.3. Ausblick

In diesem Abschnitt werden einige weiterführende Fragestellungen vorgestellt, die – ausgehend von der Themenstellung dieser Arbeit – Potenzial für darauf aufbauende Forschungsvorhaben haben.

Ausdrucksstärke und Datenquellen-Wrapper

Die momentane Konzeption sieht eine auf Beschreibungslogiken \mathcal{L}_{tripod} basierende Modellierung globaler Ontologie (TBox) vor. Die Gründe für diese Entscheidung sind im Abschnitt 5.2.2 dargelegt. In diesem Zusammenhang sind Erweiterungen der Ausdrucksstärke von \mathcal{L}_{tripod} denkbar. Hierzu muss jedoch genau untersucht werden, welche zusätzlichen Konstrukte und die damit verbundene Erhöhung der Berechnungskomplexität in Bezug auf den zu erzielende Nutzen praktikabel sind. Einen ersten Schritt in diese Richtung bieten noch ausdrucksstärkere Beschreibungslogiken $\mathcal{SR}OIQ$ [HKS06], [KGH11]. Weiterhin ergeben sich interessante Parallelen zu probabilistischen Erweiterungen für Beschreibungslogiken: Auch hier werden u. a. Fragen nach der Modellierung von unsicherem Wissen (engl. *uncertain knowledge*) gestellt [Kli08], [KPS09]. Dass dies eine ambitionierte Aufgabe ist, wird erstens allein schon durch ganz ähnliche Fragestellungen in der jahrtausendealten Ontologiedebatte der Philosophie deutlich, zwei-

tens durch die Bemühungen, solche Erweiterungen für bereits bestehende Inferenzsysteme¹ zu realisieren.

Mithilfe von SQL-Wrapper können relationale Datenquellen auf Basis eines semantischen Mappings effizient integriert werden. Die Komplexität der Entwicklung der Wrapper hängt vom Umfang der zu integrierenden Datenquelle und der technischen Eignung der gegebenen Systemschnittstellen ab. Eine gewichtige Rolle spielt hierbei auch der Umfang der strukturellen Unterschiede zwischen dem Ontologiemodell und dem durch den Wrapper abgebildeten nativen Modell der Datenquelle. Soll eine nichtrelationale Datenquelle (z. B. web service, RDF-Repository etc.) in die Metamodellstruktur des Informationsintegrationssystems eingebunden werden, beschränkt sich der Aufwand auf die Implementierung eines entsprechenden Wrappers.

Die Realisierung solcher Wrapper auf Basis eines globalen Ontologiemodells und Funktionalitäten des Integrationssystems ist Gegenstand weiterführender Arbeiten.

Cloud Computing und Semantic Mashups

Schlagworte der aktuellen Informatikforschung sind *Cloud Computing* und *Semantic Mashups*. Es ist zu erwarten, dass die Forschungsergebnisse in diesen Bereichen Einfluss auf Systemarchitekturen und Webstandards haben werden.

Die Einordnung dieser Technologien in das Integrationssystem wird weitere konzeptionelle Arbeiten erfordern. Ob sich Cloud Computing und Semantic Mashups in Form eines Moduls nahtlos in das bestehende System einfügen, oder ob grundlegende Aspekte der Architektur neu zu überdenken sind, ist eine Fragestellung, die in weiterführenden Forschungsprojekten untersucht werden muss. Tendenziell ist die Verfolgung dieser Technologieansätze, insbesondere hinsichtlich der Aspekte Kommunikation und Suche, für die Unterstützung unternehmensinterner sowie unternehmensübergreifender Produktentwicklungsprozesse höchst perspektivreich.

Wir glauben, dass das Thema Ontologie, das ursprünglich aus dem philosophischen Bereich stammt, eine zentrale Rolle in der Debatte um semantische Informationsintegration spielen kann. Ein Beispiel hierfür ist die vorliegende Arbeit, die sowohl theoretisch fundierte Grundlagen zur Realisierung eines derartigen Integrationssystems legt, als auch ein neuartiges Konzept für semantische Informationsintegration in Produktentwicklung entwickelt und umsetzt.

¹<http://pellet.owldl.com/pronto>

A. Resource Description Framework

Das *Resource Description Framework* (RDF) ist eine formale Wissensrepräsentationssprache zur Beschreibung strukturierter Informationen. Durch RDF soll erreicht werden, dass Applikationen ihre Daten in einer verteilten Umgebung miteinander austauschen können, ohne dass die Semantik der Daten bei der Kommunikation verloren geht. RDF basiert auf Graphstrukturen und stellt Informationen in Form von Aussagen (engl. *statements*) dar. Dabei besteht jede Aussage aus drei Elementen, *Tripel* genannt, einem *Subjekt*, einem *Prädikat* und einem *Objekt*. Graphische Darstellung einer Aussage ist durch eine gerichtete Kante zwischen zwei Knoten repräsentiert (Abbildung A.1). Dabei steht die Kante für ein Prädikat und beide Knoten entsprechend für das Subjekt bzw. Objekt. RDF definiert zwei grundlegende Typen von Knoten:

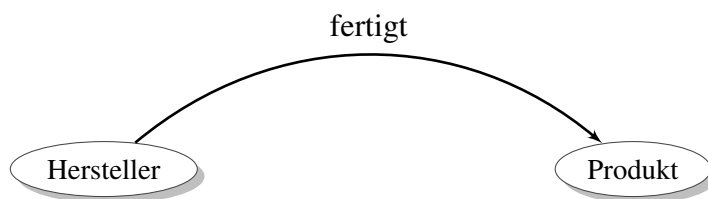


Abbildung A.1.: RDF - Tripel.

Literale und Ressourcen. Ein Literal ist ein konkreter Wert, z. B. eine Zeichenfolge oder einer Zahl. Ferner Literale können nicht für das Subjekt einer Aussage stehen, sondern nur das Objekt eines Statements sein. Eine Ressource kann dagegen, allgemein gesprochen, alles darstellen, was durch Uniform Resource Identifier (URI) ausgezeichnet werden kann. Beispiele für eine Ressource sind Mensch, Organisation, Ding und Konzept. Eine Ausnahme der vorgestellten Ressourcen-Konzeptuallisierung stellt eine besondere Art von Knoten dar, genannt *blank node*. Blank Nodes sind existentielle Variablen, die dazu verwendet werden, um Aussagen zu machen, die nicht eine bestimmte Ressource beschreiben. Ein Beispiel für eine Aussage mit einer existentiellen Variable ist „jede Dissertation wird von einer Person geschrieben“. In diesem Fall bezieht sich der Begriff „Person“ nicht auf eine einzelne Person, sondern dient dazu, die Idee zum Ausdruck zu bringen, dass jede Dissertation einen Autor hat, der eine Person ist. Die Erhaltung der ursprünglichen Semantik ist durch die Vergabe eines Namens an die Person nicht sichergestellt. Daher verwendet man bei der Formulierung dieser Art von Aussagen die blank nodes.

RDF-Aussagen können auf viele verschiedene Arten serialisiert werden. Einige der gängigsten

Formate sind heute RDF/ XML, das Terse RDF Triple Language (Turtle), und N-Triples. RDF/XML ist ein XML-Serialisierung, und es ist das einzige normative RDF-Format für den Austausch. Turtle ist kompakt und menschenfreundlich und ist daher das meist verwendete Format. N-Triples ist ein zeilen-basiertes Format, das einfach zu erzeugen und einfach zu analysieren ist, insbesondere wenn das Streaming-Serialisierung erforderlich ist.

RDF-Sprachmitteln werden um die Sprache *RDF-Schema* erweitert mit dem Ziel, auch allgemeine schematische Informationen über einen Datensatz ausdrücken zu können. Während RDF hauptsächlich dazu dient, grundlegende Aussagen über die Beziehungen zwischen Einzelobjekten (Individuen) zu treffen, bietet RDFS die Möglichkeit, terminologisches Wissen in Form von Klassen- und Propertyhierarchien und deren Zusammenhängen zu spezifizieren [HKRS08].

Nachfolgend fassen wir RDF(S)-Sprachkonstrukte zusammen: Grundsätzliche Einschränkun-

RDF(S)-Sprachkonstrukte		
RDF(S) Klassen		
rdfs:Class	rdf:Property	rdfs:Resource
rdfs:Literal	rdfs:Datatype	rdf:XMLLiteral
RDF(S) Eigenschaften		
rdfs:range	rdfs:domain	rdf:type
rdfs:subClassOf	rdfs:subPropertyOf	rdfs:label
rdfs:comment		
RDF Listen		
rdfs:Container	rdf:Bag	rdf:Seq
rdf:Alt	rdf:li	rdf:_1
rdf:_2
rdfs:ContainerMembershipProperty	rdfs:member	rdf:List
rdf:first	rdf:rest	rdf:nil
RDF Attribute		
rdf:about	rdf:ID	rdf:resource
rdf:nodeID	rdf:datatype	
XML Attribute		
xml:base	xmlns	xml:lang
Reifikation		
rdf:Statement	rdf:subject	rdf:predicate
rdf:object		
RDF(S) weitere Elemente		
rdf:RDF	rdfs:seeAlso	rdfs:isDefinedBy
rdf:value		

gen der Modellierungsfähigkeit von RDFS besteht in der Unmöglichkeit negativer Aussagen. Das heißt, dass RDFS keine Konstrukte zur Verfügung stellt, um auszudrücken, dass etwas *nicht gilt*. Gewiss besteht die Möglichkeit die Negation in Bezeichner von Klassen- und Rollen miteinzubinden, also z.B. einfach den Klassennamen `Nichteisenmetall` oder auch ein Rolle mit dem Namen `NichtGefertigtVon` zu spezifizieren. Allerdings liegt hierbei keine Semantik vor, die eine entsprechende Interpretation solcher Beschreibungselemente belegt. So würden die beiden Aussagen nicht zu einem Widerspruch führen, auch wenn man dies intuitiv annehmen

Beispiel	(RDF-Aussagen)	
<code>präfix:Gold</code>	<code>rdf:type</code>	<code>präfix:Nichteisenmetall</code>
<code>präfix:Stahl</code>	<code>rdf:type</code>	<code>präfix:Eisenmetall</code>

würde. Darüber hinaus gibt es in RDFS keine Möglichkeit zu beschreiben, dass es keine Elemente bzw. Instanzen gibt, die gleichzeitig beider Klassen angehören. Im nächsten Abschnitt stellen wir eine Ontologiesprache vor, die über solche Mittel verfügt.

B. The OWL Web Ontology Language

RDF(S) eignet sich zur Modellierung einfacher Ontologien. Jedoch zur Darstellung komplexerer Zusammenhänge genügen die zur Verfügung stehende Ausdrucksmittel der Sprache RDF(S) nicht. Folglich um komplexes Wissen darzustellen, verwendet man ausdrucksstarke Wissensrepräsentationssprachen, die auf formaler Logik basieren. Basierend auf dem logischen Schlußfolgern wird damit auch der Zugriff auf implizites Wissen ermöglicht [HKRS08].

Das Akronym OWL steht für *Web Ontology Language*. Die erste Version von OWL wurde im Jahr 2004 vom W3C standardisiert. Bei der Entwicklung von OWL wurde sehr stark auf das Gleichgewicht zwischen Ausdrucksstärke der Sprache einerseits und effizienten Schlussfolgern andererseits geachtet. Folglich ist OWL eine Erweiterung von RDF(S) und bietet ein zusätzliches Vokabular von Eigenschaften und Klassen, das mit der Semantik versehen ist. Diese Klassen und Eigenschaften werden verwendet, um ausdrucksstarke Ontologien zu bauen, die wiederum verwendet werden, um Ressourcen zu beschreiben. Nachfolgend werden Sprachkonstrukte von OWL zusammenfassend beschrieben.

Annotationseigenschaften	(Annotation Property)
owl:AnnotationProperty	Die Klasse aller Annotationseigenschaften.
rdfs:label	Das Label bietet die Möglichkeit eine Ressource benutzerfreundlicher als beispielsweise durch URI darzustellen.
rdfs:comments	Textuelle Beschreibung einer Ressource.
rdfs:seeAlso	Eine Eigenschaft, die eine Ressource angibt, die zusätzliche Informationen liefert.
rdfs:isDefinedBy	Eine Eigenschaft, die eine Ressource angibt, die das Subjekt einer Ressource definiert.
owl:deprecated	Eine Eigenschaft, die angibt, ob der Subjekt- URI veraltet ist oder nicht.
owl:DeprecatedClass	Die Klasse, die alle veralteten Klassen beinhaltet.
owl:DeprecatedProperty	Die Klasse, die alle veralteten Eigenschaften beinhaltet.

owl:priorVersion	Eine Eigenschaft, die eine frühere Version der Ontologie spezifiziert, die das Subjekt der Aussage ist.
owl:backwardCompatibleWith	Eine Eigenschaft, die die URI einer Ontologie angibt, die kompatibel mit der Ontologie ist, die das Subjekt der Aussage ist.
owl:incompatibleWith	Eine Eigenschaft, die die URI einer Ontologie angibt, die nicht kompatibel mit der Ontologie ist, die das Subjekt der Aussage ist.

Tabelle B.1.: Annotationseigenschaften von OWL

Klassen	(Classes)
rdfs:subClassOf	Eine Beziehung zwischen zwei Klassen, die besagt, dass eine Klasse spezifischer als eine andere ist.
owl:equivalentClass	Eine Beziehung, die angibt, dass Instanzen bzw. Individuals von zwei Klassen äquivalent sind.
owl:Thing	Eine Klasse, die als Oberklasse für alle Instanzen einer Ontologie gilt.
owl:Nothing	Eine Klasse, die keine Instanzen enthält.
owl:oneOf	Die Instanzen einer Klasse sind auf eine bestimmte Menge beschränkt.
owl:intersectionOf	Die Instanzen dieser Klasse sind Instanzen alle angegebenen Klassen.
owl:unionOf	Die Instanzen dieser Klasse sind Instanzen mindestens einer der genannten Klassen.
owl:complementOf	Die Instanzen dieser Klasse können nicht Instanzen der spezifizierten Klasse sein.
owl:disjointWith	Eine Beziehung, die angibt, dass zwei Klassen keine gemeinsame Instanzen besitzen.
owl:disjointUnionOf	Eine Beziehung, die angibt, dass diese Klasse eine Vereinigung der angegebenen Klassen ist und dass diese Klassen paarweise disjunkt sind.

Tabelle B.2.: Klassen in OWL

Individuen	(Individuals)
rdf:type	Eine Beziehung, die angibt welcher Klasse ein Individuum bzw. eine Instanz angehört.
owl:sameAs	Eine Beziehung, die angibt, dass zwei unteschiedliche Individuenbezeichner dasselbe Individuum spezifizieren.
owl:differentFrom	Eine Beziehung, die angibt, dass zwei Individuen unterschiedlich sind.
owl:AllDifferent	Eine Beziehung, die angibt, dass alle Individuen in einer Menge paarweise disjunkt sind.

Tabelle B.3.: Instanzen in OWL

Rollen	(Properties)
owl:ObjectProperty	Eine Klasse aller Rollen, die zwei Individuen verknüpft.
owl:DatatypeProperty	Eine Klasse aller Rollen, die ein Individuum mit einem Literal verknüpft.
owl:topObjectProperty	Eine Rolle, die alle möglichen Paare von Individuen miteinander verbindet.
owl:bottomObjectProperty	Eine Rolle, die keine Paare von Individuen miteinander verbindet.
owl:topDataProperty	Eine Rolle, die alle möglichen Individuen mit allen möglichen Literalen verbindet.
owl:bottomDataProperty	Eine Rolle, die keine Individuen mit einem Literal verbindet.
rdfs:domain	Eine Rolle, die eine Zuweisung von Typen zu Subjekten gestattet.
rdfs:range	Eine Rolle, die eine Zuweisung von Typen zu Objekten gestattet.
rdfs:subPropertyOf	Eine Beziehung zwischen zwei Rollen, die angibt, dass eine Rolle spezifizierter als die andere ist.

owl:equivalentProperty	Eine Beziehung, die angibt, dass zwei Rollen gleichwertig sind.
owl:inverseOf	Eine Beziehung, die angibt, dass zwei Rollen invers zueinander sind.
owl:SymmetricProperty	Eine Klasse aller Rollen, die symmetrisch sind.
owl:AsymmetricProperty	Eine Klasse aller Rollen, die explizit nicht symmetrisch sind.
owl:ReflexiveProperty	Eine Klasse aller Rollen, die reflexiv sind.
owl:IrreflexiveProperty	Eine Klasse aller Rollen, die nicht reflexiv sind.
owl:TransitiveProperty	Eine Klasse aller Rollen, die transitiv sind.
owl:FunctionalProperty	Eine Klasse aller Rollen für welche, die gegebene Domäne nur einen Wertebereich haben kann.
owl:InverseFunctionalProperty	Eine Klasse aller Rollen für welche, das gegebene Objekt einer Aussage nur ein Subjektwert haben kann.
owl:propertyDisjointWith	Eine Beziehung, die angibt, dass zwei Rollen disjunkt sind.
owl:propertyDisjointWith	Eine Beziehung, die angibt, dass zwei Rollen disjunkt sind.

Tabelle B.4.: Rollen in OWL

Rollenrestriktionen (Property Restrictions)	
owl:Restriction	Eine Klasse aller Restriktionen.
owl:SelfRestriction	Eine Klasse aller Selbst -Restriktionen.
owl:onProperty	Eine Rolle, die die Rolle identifiziert für welche eine Einschränkung gilt.
owl:allValuesFrom	Eine Rolle, die angibt, dass alle Instanzen in dieser Klasse nur den Werte aus dem angegebenen Bereich bezüglich der angegebenen Rolle annehmen können.
owl:someValuesFrom	Eine Rolle, die angibt, dass alle Instanzen in dieser Klasse mindestens einen Werte aus dem angegebenen Bereich bezüglich der angegebenen Rolle annehmen können.

owl:hasValue	Eine Rolle, die angibt, dass alle Instanzen dieser Klasse nur den angegebenen Wert bezüglich der angegebenen Rolle annehmen können.
owl:minCardinality	Eine Rolle, die angibt, dass jede Instanz spezifizierter Klasse über mindestens n spezifizierten Rollen verfügen muss.
owl:maxCardinality	Eine Rolle, die angibt, dass jede Instanz spezifizierter Klasse über maximal n spezifizierten Rollen verfügen kann.
owl:cardinality	Eine Rolle, die angibt, dass jede Instanz spezifizierter Klasse über genau n spezifizierten Rollen verfügen muss.
owl:minQualifiedCardinality	Eine Rolle, die angibt, dass jede Instanz über mindestens n spezifizierten Rollen verfügen muss. Hierbei gehört der Bereich dieser Rolle einer spezifizierten Klasse an.
owl:maxQualifiedCardinality	Eine Rolle, die angibt, dass jede Instanz über maximal n spezifizierten Rollen verfügen muss. Hierbei gehört der Bereich dieser Rolle einer spezifizierten Klasse an.
owl:qualifiedCardinality	Eine Rolle, die angibt, dass jede Instanz über genau n spezifizierten Rollen verfügen muss. Hierbei gehört der Bereich dieser Rolle einer spezifizierten Klasse an.

Tabelle B.5.: Rollenrestriktionen in OWL

Abbildungsverzeichnis

1.1	Relevanz „Effizienter Suche“ für Unternehmen [Nie12]	2
1.2	Marktzuwach für Software zur Informationsintegration [Olo08]	4
1.3	Struktur der vorliegenden Arbeit	8
2.1	Zusammenhang zwischen Daten, Informationen und Wissen nach [VDI5610] .	10
2.2	Ansätze zur Informationsintegration: virtuell vs. materialisiert nach [Kud07] . .	13
2.3	Vereinfachte Architektur einer mediatorbasierten Informationsintegration . . .	14
2.4	Konventionelle Systeme vs. wissensbasierte Systeme [Pup93]	22
2.5	Syntaktische und semantische Ebene [SS11]	23
2.6	Syntaktische Ableitung und semantische Folgerung. $Mod(F)$ steht für die Menge der Modelle einer Formel F [Ert09]	25
2.7	Inferenzmechanismen	28
2.8	Beispiel eines induktiven Schlusses (vgl. [Los06])	28
2.9	Beispiel eines abuktiven Schlusses (vgl. [Los06])	29
2.10	Beispiel eines deduktiven Schlusses (vgl. [Los06])	29
2.11	Wissensbasiertes System mit Beschreibungslogiken	35
3.1	CORBA Architektur [MR97]	47
3.2	Webservice-Architektur [STK02]	49
3.3	Architektur des .NET Frameworks [Cou10]	50
3.4	BizTalk Server Architektur nach [Cha07]	53
3.5	Möglichkeiten von IBM WebSphere nach [SBF ⁺ 08]	54
3.6	SOA Tempel [Mel10]	56
3.7	Das Semantic Web in Schichten nach TIM BERNERS-LEE [BLHL01]	59
3.8	Modulkonzept der PDM Enabler nach [KEME99]	64
3.9	PDM-Systemarchitektur [VDI02]	68
3.10	SAP NetWeaver Architektur nach [AG12]	70
3.11	iViP-Plattformarchitektur [KTA03]	72
3.12	PDTnet - Umsetzung von Data Exchange und Data Sharing [Pro02]	73
3.13	Semantic Virtual Engineering Environment [Sch07]	76
4.1	Modularer Aufbau des Integrationssystems.	87

5.1	Informationsinhalte der Metadaten	90
5.2	Die vier Phasen des Lösungskonzepts	91
5.3	Vier nahezu äquivalente Schemata in unterschiedlichen Datenmodellen	96
5.4	Beispiel einer Baumstruktur [Ehr95]	98
5.5	Schemaauszug aus einer relationalen Datenbank	103
5.6	Ontologiemodell zum relationalen Schemaauszug	104
5.7	Mapping-Prozess im Überblick in Anlehnung an LESER UND NAUMANN [LN07]	112
5.8	Beispiel für ein GLaV-Mappingprozess	115
5.9	Direktes Mapping basierend auf G(L)aV-Ansatz nach [Biz03], [LN07], [Lan09]	117
5.10	Systemarchitektur	136
6.1	e-CUSTOM Lösungsansatz [OBAB11]	140
6.2	Abstrakte Architektur des e-CUSTOM-Systems	141
6.3	Benutzungsoberfläche des Protégé-OWL Systems	146
6.4	Phasenmodell zur Ontologieerstellung im Kontext der Produktentwicklung . . .	148
6.5	Teilaspekte zur Spezifikation von ontologischen Konzepten	150
6.6	SQL-Wrapper	160
6.7	Registrierungsfenster für Datenquellen	167
6.8	Das Hauptfenster der Benutzerschnittstelle	168
6.9	Korrespondenzenverwaltung	168
6.10	Ergebnisdarstellung	169
A.1	RDF - Tripel.	177

Tabellenverzeichnis

2.1	Wissensarten [Rud98]	11
3.1	Merkmale einer SOA [Mel10]	57
3.2	Die Unterteilung der STEP-Normenreihe	63
3.3	PDM – übergeordnete Ziele	66
5.1	Gruppenzuordnung der Datenquellen	95
5.2	Drei Vorschriften zur Abbildung der Datenbankschemata	102
5.3	Semantik der Sprache \mathcal{L}_{tripod}	106
5.4	Die Regeln zum Umschreiben von Anfragen hinsichtlich \mathcal{L}_{tripod} -Terminologie	128
6.1	Zwei einfache Datenquellen für eine eCUSTOM-Produktontologie	163
6.2	Mapping-Beziehungen zwischen eCUSTOM-Produktontologie und entsprechenden Datenquellen	164
6.3	Unterschiedliche Kombinationen der lokalen Anfragen	165
B.1	Annotationseigenschaften von OWL	182
B.2	Klassen in OWL	182
B.3	Instanzen in OWL	183
B.4	Rollen in OWL	184
B.5	Rollenrestriktionen in OWL	185

Literaturverzeichnis

- [ABM06] S. Aier, Competence Center Enterprise Application Integration Berlin, and S. Marten. *Enterprise application integration: Serviceorientierung und nachhaltige Architekturen*. Reihe Enterprise architecture. Gito-Verl., 2006.
- [AD98] Serge Abiteboul and Oliver M. Duschka. Complexity of answering queries using materialized views. In *PODS*, pages 254–263, 1998.
- [ADE05] V. Arnold, H. Dettmering, and T. Engel. *Product Lifecycle Management beherrschen: Ein Anwenderhandbuch für den Mittelstand*. Springer, 2005.
- [AG12] SAP AG. Sap netweaver: Architektur. <http://help.sap.com/> Stand vom 05.02.2012, February 2012.
- [AHV95] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
- [Aie06] S. Aier. *Integrationstechnologien als Basis einer nachhaltigen Unternehmensarchitektur: Abhängigkeiten zwischen Organisation und Informationstechnologie*. Reihe Enterprise architecture. Gito-Verl., 2006.
- [AKS05] Markus Aleksy, Axel Korthaus, and Martin Schader. *Implementing distributed systems with Java and CORBA*. Springer, 2005.
- [ALSS03] S. Abeck, P. Lockemann, J. Schiller, and J. Seitz. *Verteilte Informationssysteme - Integration von Datenübertragungstechnik und Datenbanktechnik*. dpunkt.verlag, 2003.
- [AMN⁺06] Jürgen Angele, Eddie Mönch, Andreas Nierlich, Heiko Rudat, and Hans-Peter Schnurr. Anwendungen und Good Practices Semantischer Technologien. pages 337–356. 2006.
- [And00] R. Anderson. *Professional XML*. Programmer to programmer. Wrox Press, 2000.
- [AS99] C. H Antoni and T. Sommerlatte. Wissensmanagement - wie deutsche firmaen ihr wissen profitabel machen. *Symposium Publishing*, 1999.

- [AT00] R. Anderl and D. Trippner. *STEP. Standard for the Exchange of Product Model Data.: Eine Einführung in die Entwicklung, Implementierung und industrielle Nutzung der Normenreihe ISO 10303 (STEP)*. Teubner, 2000.
- [AvH08] Grigoris Antoniou and Frank van Harmelen. *A Semantic Web Primer*. MIT Press, Cambridge, MA, 2. edition, 2008.
- [AVR08] Reiner Anderl, Diana Völz, and Thomas Rollmann. Knowledge integration in global engineering. In *IESA*, pages 471–482, 2008.
- [Bai08] Elisabeth Baier. *Semantische Technologien in Wissensmanagementlösungen*. Number 13 in FAZIT-Schriftenreihe. MFG-Stiftung Baden-Württemberg, Stuttgart, 2008.
- [Bas03] D. Basler. *Komponenten für SAP mit Java*. Software-und-Support-Verl., 2003.
- [BB04] C. Britton and P. Bye. *IT architectures and middleware: strategies for building large, integrated systems*. Unisys Series. Addison-Wesley, 2004.
- [BBL05] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the el envelope. In *IJCAI*, pages 364–369, 2005.
- [BBM⁺06] Wolfgang Beer, Dietrich Birngruber, Hanspeter Mössenböck, Herbert Praehofer, and Albrecht Wöß. *Die .NET-Technologie - Grundlagen und Anwendungsprogrammierung: aktualisiert auf .NET 2.0, 2. Auflage*. dpunkt.verlag, 2006.
- [BCM⁺03] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.
- [BCP01] Lothar Buhl, Jörg Christ, and Ulrich Pape. Marktstudie: Softwaresysteme für enterprise application integration. *ALB-HNI Verlagsschriftenreihe, Bd. 7, Fraunhofer-Anwendungszentrum für Logistikorientierte Betriebswirtschaft*, 2001.
- [BCVB01] Sonia Bergamaschi, Silvana Castano, Maurizio Vincini, and Domenico Beneventano. Semantic integration of heterogeneous information sources. *Data Knowl. Eng.*, 36(3):215–249, 2001.
- [BF92] Wolfgang Bibel and Ulrich Furbach. Logik, ki und intellektik. *KI*, 6(3):91–94, 1992.
- [BH03] Markus Burghardt and Svenja Hagenhoff. *Web Services - Grundlagen und Kern-technologien*. Technical report, Georg-August-Universität Göttingen Institut für Wirtschaftsinformatik, 2003.

- [BH08] Philip A. Bernstein and Laura M. Haas. Information integration in the enterprise. *Commun. ACM*, 51(9):72–79, 2008.
- [Biz03] Christian Bizer. D2r map-a database to rdf mapping language. 2003.
- [BK10] Simon M. Becker and Anne-Thérèse Körtgen. Graph transformations and model-driven engineering. chapter Integration tools for consistency management between design documents in development processes, pages 683–718. Springer-Verlag, Berlin, Heidelberg, 2010.
- [BKI06] Christoph Beierle and Gabriele Kern-Isberner. *Methoden wissensbasierter Systeme - Grundlagen, Algorithmen, Anwendungen*. Computational intelligence. Vieweg, 2006.
- [BLHL01] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 2001.
- [BLR97] Catriel Beeri, Alon Y. Levy, and Marie-Christine Rousset. Rewriting queries using views in description logics. In *roc. of the 16th ACM SIGACT SIGMOD SIGART Sym. on*, pages 99–108, 1997.
- [Bos11] Johan Bos. A survey of computational semantics: Representation, inference and knowledge in wide-coverage text understanding. *Language and Linguistics Compass*, 5(6):336–366, 2011.
- [BT06] Andreas Blumauer and Pellegrini Tassilo. Semantic web und semantische technologien: Zentrale begriffe und unterscheidungen. pages 9–25. Springer Verlag, Berlin, Heidelberg, 2006.
- [CCGL02] Andrea Cali, Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. On the expressive power of data integration systems. In *ER*, pages 338–350, 2002.
- [CCGL04] Andrea Cali, Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Data integration under integrity constraints. *Inf. Syst.*, 29(2):147–163, 2004.
- [CDGL⁺09] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati, and Marco Ruzzi. *Using OWL in Data Integration*. Springer, 2009.
- [CDGLR11] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. View-based query answering in description logics: Semantics and complexity. *J. of Computer and System Sciences*, 2011. Accepted manuscript available online at <http://dx.doi.org/10.1016/j.jcss.2011.02.011>.

- [CDK⁺07] J. Conrad, T. Deubel, C. Köhler, S. Wanke, and C. Weber. Comparison of knowledge representation in pdm and by semantic networks. *16th International Conference on Engineering Design - ICED 07 The Design Society*, 2007.
- [CGL⁺05] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Data complexity of query answering in description logics. In *Description Logics*, 2005.
- [CGL⁺07] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The *dl-lite* family. *J. Autom. Reasoning*, 39(3):385–429, 2007.
- [CGL⁺09] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Conceptual modeling for data integration. In *Conceptual Modeling: Foundations and Applications*, pages 173–197, 2009.
- [CGLR08] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. View-based query answering over description logic ontologies. In *KR*, pages 242–251, 2008.
- [CGLV00] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. What is view-based query rewriting? In *KRDB*, pages 17–27, 2000.
- [CGT90] Stefano Ceri, Georg Gottlob, and Letizia Tanca. *Logic Programming and Databases*. Springer, 1990.
- [Cha07] David Chappell. Introducing biztalk server 2006 r2. Technical report, Microsoft Corporation, 2007.
- [CHKT06] Stefan Conrad, Wilhelm Hasselbring, Arne Koschel, and Roland Tritsch. *Enterprise Application Integration: Grundlagen – Konzepte – Entwurfsmuster – Praxisbeispiele*. Spektrum, Heidelberg, 2006.
- [Cho07] Jan Chomicki. Semantic optimization techniques for preference queries. *Inf. Syst.*, pages 670–684, 2007.
- [Cho08] Jan Chomicki. Consistent query answering: The first ten years. In *SUM'08*, pages 1–3, 2008.
- [CJ02] David Chappell and Tyler Jewell. *Java Web Services*. O'Reilly, german edition, 2002.

- [CLR03] Andrea Cali, Domenico Lembo, and Riccardo Rosati. Intensional query processing in data integration systems under integrity constraints. In *SEBD*, pages 475–482, 2003.
- [Com12] European Commission. Sevenpro - semantic virtual engineering environment for product design. <http://sevenpro.org/> Stand vom 05.02.2012, February 2012.
- [Con02] Stefan Conrad. Schemaintegration integrationskonflikte, lösungsansätze, aktuelle herausforderungen. *Inform., Forsch. Entwickl.*, 17(3):101–111, 2002.
- [Cou10] Microsoft Official Academic Course. *70-536: Microsoft .NET Framework Application Development Foundation, Package*. John Wiley & Sons, 2010.
- [DIN79] DIN-Norm 2330 Begriffe und Benennungen, 1979.
- [DIN80] DIN 2331. Deutsches Institut für Normung: Begriffssysteme und ihre Darstellung, 1980.
- [DIN93] DIN 2330. Deutsches Institut für Normung: Begriffe und Benennungen - Allgemeine Grundsätze, 1993.
- [DLNS94] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. Deduction in concept languages: From subsumption to instance checking. *J. Log. Comput.*, 4(4):423–452, 1994.
- [DM02] Phan Minh Dung and Paolo Mancarella. Production systems with negation as failure. *IEEE Trans. Knowl. Data Eng.*, 14(2):336–352, 2002.
- [Ebe03] Heinz Ebensperger. *IBM WebSphere Applikationsserver*. Addison-Wesley, 2003.
- [Edm94] Ernest A. Edmonds. Knowledge-based systems: a new perspective. *Knowl.-Based Syst.*, 7(1), 1994.
- [Ehr95] Klaus Ehrlenspiel. *Integrierte Produktentwicklung, Methoden für Prozessorganisation, Produkterstellung und Konstruktion*. VDI-Buch. Carl Hanser Verlag, Wien, 1., Aufl. edition, 1995.
- [Ehr04] Alex Ehrler. *Eine Integrierte Plattform für Semantische Standards und Softwarekomponenten zur Unterstützung Unternehmensübergreifender Produktentwicklungsprozesse*. PhD thesis, Universität Karlsruhe (TH), Institut RPK, 2004.
- [Eng05] Torsten Engel. PLM im Mittelstand: Verstehen, was man einführt. *eDM-Report*, February 2005.

- [Eng06] T. Engel. *Ein Beitrag zur unternehmensübergreifenden Integration von Informationssystemen*. PhD thesis, 2006.
- [Ert09] Wolfgang Ertel. *Grundkurs Künstliche Intelligenz: eine praxisorientierte Einführung*. Number ISBN 978-3-8348-0783-0. Vieweg + Teubner, Wiesbaden, second edition, 2009.
- [ES09] Martin Eigner and Ralph Stelzer. *Product Lifecycle Management: Ein Leitfaden für Product Development und Life Cycle Management*. VDI-Buch. Springer, Berlin, 2., neu bearb. Aufl. edition, 2009.
- [FC99] D. Flanagan and W. Crawford. *Java Enterprise in a nutshell: a desktop quick reference*. Java series. O'Reilly, 1999.
- [Fen02] Steven J Fenves. NISTIR 6736 A core product model for representing design information . *Technology, National Institute of Standards and Technology*, 2002.
- [Fit96] Melvin Fitting. *First-order logic and automated reasoning (2. ed.)*. Graduate texts in computer science. Springer, 1996.
- [Fow95] J. Fowler. *STEP for Data Management, Exchange and Sharing*. Technology Appraisals, 1995.
- [FS69] I. P. Fellegi and A. B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64:1183–1210, 1969.
- [Fuc08] F. Fuchs. *Semantische Modellierung und Reasoning für Kontextinformationen in Infrastrukturnetzen*. PhD thesis, LMU München: Fakultät für Mathematik, Informatik und Statistik, 2008.
- [FZ09] P. Finger and K. Zeppenfeld. *Soa Und Webservices*. Informatik Im Fokus. Springer, 2009.
- [GAWP93] H. Grabowski, R. Anderl, H.J. Warnecke, and A. Polly. *Integriertes Produktmodell*. Beuth, 1993.
- [GHMS09] Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, and Ulrike Sattler, editors. *Proceedings of the 22nd International Workshop on Description Logics (DL 2009)*, Oxford, UK, July 27-30, 2009, volume 477 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.
- [GKD97] Michael R. Genesereth, Arthur M. Keller, and Oliver M. Duschka. Infomaster: An information integration system. In *SIGMOD Conference*, pages 539–542, 1997.

- [GLHS08] Birte Glimm, Carsten Lutz, Ian Horrocks, and Ulrike Sattler. Answering conjunctive queries in the \mathcal{SHIQ} description logic. *Journal of Artificial Intelligence Research*, 31:150–197, 2008.
- [GLW02] H. Grabowski, R.S. Lossack, and J. Weißkopf. *Datenmanagement in der Produktentwicklung: automatische Klassifikation von Produktdaten aus 3D-CAD-Systemen, PDM- und ERP-Systemen, XML- und Office-Dokumenten*. Hanser, 2002.
- [GM97] Hans Grabowski and Eike Meis. Using ontologies for the integrated product model development. *AAAI Technical Report*, 1997.
- [GMH95] Hans Grabowski, Eike Meis, and Karl Hain. Integrated product and production model. (integriertes produkt- und produktionsmodell.). *it + ti. Informationstechnik und Technische Informatik*, 37(5):32–38, 1995.
- [Goo97] Gerhard Goos. *Vorlesungen über Informatik Band 3: Berechenbarkeit, formale Sprachen und Spezifikationen*. Springer, Jan 1997.
- [Gou99] Arthur Gould. The EAI Challenge. *Enterprise Systems Journal*, 1999.
- [GPFLC04] A. Gómez-Pérez, M. Fernández-López, and O. Corcho. *Ontological Engineering: With Examples from the Areas of Knowledge Management, E-Commerce and the Semantic Web*. Advanced Information and Knowledge Processing. Springer, 2004.
- [GPL08] Andreas Gaag, Josef Ponn, and Udo Lindemann. SSuchen und Finden im Maschinen- und Anlagenbau - Eine Studie in Zusammenarbeit mit dem VDMA. CiDaD Working Paper Series(4), Juni 2008. Lehrstuhl für Produktentwicklung, Technische Universität München.
- [GRG98] H. Grabowski, S. Rude, and G. Grein. *Universal Design Theory*. Shaker Verlag, 1998.
- [Gru93] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199 – 221, 1993.
- [Gru94] T. R. Gruber. Towards principles for the design of ontologies used for knowledge sharing. *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Kuwer Academic Publishers, 1994.
- [Gru09] Tom Gruber. Ontology. In *Encyclopedia of Database Systems*, pages 1963–1965. 2009.

- [Haa07] Laura M. Haas. Beauty and the beast: The theory and practice of information integration. In *ICDT*, pages 28–43, 2007.
- [Hal01] Alon Y. Halevy. Answering queries using views: A survey. *VLDB J.*, 10(4):270–294, 2001.
- [Hal09] Alon Y. Halevy. Information integration. In *Encyclopedia of Database Systems*, pages 1490–1496. 2009.
- [Har94] A. Hars. *Referenzdatenmodelle*. Gabler Wirtschaft. Gabler, 1994.
- [HK07] L. Heilig and S. Karch. *SAP NetWeaver*. SAP Press. Galileo Press GmbH, 2007.
- [HKRS08] P. Hitzler, M. Krötzsch, S. Rudolph, and Y. Sure. *Semantic Web: Grundlagen*. Springer, 2008.
- [HKS06] Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The even more irresistible sroiq. In *KR*, pages 57–67, 2006.
- [Hor04] Oliver Hornberg. *Collaborative Engineering in interkulturellen Entwicklungspartnerschaften*. PhD thesis, Universität Karlsruhe, Institute for Information Management in Engineering (IMI), 2004.
- [HST99] Theo Härder, Günter Sauter, and Joachim Thomas. The intrinsic problems of structural heterogeneity and an approach to their solution. *VLDB J.*, 8(1):25–43, 1999.
- [HWG05] F.J. Heiss, E. Weirich, and G. Gratzl. *SAP NetWeaver Web Application Server*. Addison-Wesley, 2005.
- [ISO94] ISO. *ISO 10303-1: Industrial Automation Systems and Integration—Product Data Representation—and Exchange—Part 1: Overview and fundamental principles*. ISO Standard, 1994.
- [iVi12] ProSTEP iViP. PDTnet XML Schema @ONLINE, 2012.
- [Jen01] Frank Jenne. *PDM-basiertes Entwicklungsmonitoring. Ein Beitrag zur Planung und Steuerung von Entwicklungsprozessen*. PhD thesis, Universität Karlsruhe (TH), Institut RPK, 2001.
- [Jen11] Jena Community. Jena - A Semantic Web Framework for Java. available: <http://jena.sourceforge.net/index.html>, 2011.

- [JLJ⁺10] René J. Jorna, Kecheng Liu, Guorui Jiang, Keiichi Nakata, and Lily Sun, editors. *ICISO 2010 - Proceedings of the Twelfth International Conference on Informatics and Semiotics in Organisations, Reading, U.K., July 19-21, 2010*. SciTePress, 2010.
- [Joh06] Michael John. Semantische Technologien in der betrieblichen Anwendung: Ergebnisse einer Anwenderstudie. Fraunhofer-Institut für Rechnerarchitektur und Softwaretechnik (FIRST), Berlin, September 2006.
- [JRCHB11] Ernesto Jiménez-Ruiz, Bernardo Cuenca Grau, Ian Horrocks, and Rafael Berlanga Llavori. Supporting concurrent ontology development: Framework, algorithms and tool. *Data and Knowledge Engineering*, 70(1):146—164, 2011.
- [Kai02] M. Kaib. *Enterprise Application Integration: Grundlagen, Integrationsprodukte, Anwendungsbeispiele*. Deutscher Universitäts-Verlag, 2002.
- [Kar12] Andreas Karcher. Enzyklopädie der Wirtschaftsinformatik - Object-Request-Broker. public, Juli 2012. letzter Aufruf.
- [KEME99] Krisch, Ehrler, Mühl, and Ertel. PDM Enabler-Workshop innerhalb des iVip-Projekts. Technical report, SAP AG, 1999.
- [KFG07] Frank-Lothar Krause, Hans J. Franke, and Jürgen Gausemeier. *Innovationspotenziale in der Produktentwicklung*. Hanser Verlag, 2007.
- [KGH11] Ilianna Kollia, Birte Glimm, and Ian Horrocks. Query answering over sroiq knowledge bases with sparql. In *Description Logics*, 2011.
- [Kil10] F. Kiltz. *Java Webservices*. mitp-Verlag, 2010.
- [KLA05] *Unternehmensspezifisches Klassifikationssystem zur effizienten Datenverwaltung(mit Anwendungsszenarien aus der Praxis): Abschlußbericht des Verbundprojektes"Klassifikationssysteme automatisiert erstellen"(KLAUSTER)*. Univ.-Verlag Karlsruhe, 2005.
- [Kli08] Pavel Klinov. Pronto: A non-monotonic probabilistic description logic reasoner. In *ESWC*, pages 822–826, 2008.
- [KLSS] Thomas Kirk, Alon Y. Levy, Yehoshua Sagiv, and Divesh Srivastava. The information manifold. In *In Proceedings of the AAAI 1995 Spring Symp. on Information Gathering from Heterogeneous, Distributed Enviroments*, pages 85–91.
- [Kop01] Franz Kopper. Supply chain management benötigt robuste integrationslösungen. *Information, Management & Consulting*, 16:92–95, 2001.

- [KPS09] Pavel Klinov, Bijan Parsia, and Ulrike Sattler. On correspondences between probabilistic first-order and description logics. In *Description Logics*, 2009.
- [KTA03] F.-L. Krause, T Tang, and U Ahle. Leitprojekt ivip: Integrierte virtuelle produktentstehung. Technical report, 2003.
- [Kud07] T. Kudraß. *Taschenbuch Datenbanken*. Hanser Fachbuchverlag, 2007.
- [Lan09] Andreas Langegger. *A Flexible Architecture for Virtual Information Integration based on Semantic Web Concepts*. PhD thesis, Institut fuer Anwendungsorientierte Wissensverarbeitung, 2009.
- [Len02] M. Lenzerini. Data integration: A theoretical perspective. In *In Proceedings of the 21st ACM SI-GACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 233–246, 2002.
- [Lif85] Vladimir Lifschitz. Closed-world databases and circumscription. *Artif. Intell.*, 27(2):229–235, 1985.
- [LKS99] Harald Ließmann, Thomas Kaufmann, and Benno Schmitzer. Bussysteme als schlüssel zur betriebswirtschaftlich-semantischen kopplung von anwendungssystemen. *Wirtschaftsinformatik*, 41(1):12–19, 1999.
- [LLS05] Jochen Löhl, Mario Lörcher, and Ingo Sahn. Historische entwicklung von datenbanken. public, handout, July 2005. Fachhochschule Heilbronn, Hochschule für Technik und Wirtschaft.
- [LM07] C. Lutz and M. Milicic. A tableau algorithm for DLs with concrete domains and GCIs. *Journal of Automated Reasoning*, 38(1–3):227–259, 2007.
- [LN07] Ulf Leser and Felix Naumann. *Informationsintegration*. dpunkt.verlag, 2007.
- [Lof07] R.J. Loftin. *First Steps: Developing BizTalk Applications*. Springer eBooks collection: Professional and applied computing. Apress, 2007.
- [Los06] Ralf-Stefan Lossack. *Wissenschaftstheoretische Grundlagen für die rechnerunterstützte Konstruktion*. Springer-Verlag Berlin Heidelberg, 2006.
- [LR02] Paul Levi and Ulrich Rembold. *Einführung in die Informatik - für Naturwissenschaftler und Ingenieure (4. Aufl.)*. Hanser, 2002.
- [LRO96] A.Y. Levy, A. Rajaraman, and J Ordille. Querying heterogeneous information sources using source description. In *In Proceedings of the International Conference on Very Large Databases (VLDB)*, 1996.

- [Mar03] Mike Marin. Business process technology: From eai and workflow to bpm. *File-NET Corporation, USA*, page 133 f, 2003.
- [Mel10] I. Melzer. *Service-orientierte Architekturen mit Web Services: Konzepte-Standards- Praxis*. Spektrum Akademischer Verlag, 2010.
- [Mer02] M. Merz. *E-Commerce und E-Business: Marktmodelle, Anwendungen und Technologien*. dpunkt-Verl., 2002.
- [Min81] M. Minsky. A framework for representing knowledge. In J. Haugeland, editor, *Mind Design*. MIT Press, 1981. republished in *Readings in Knowledge Representation*.
- [Mit01] Prasenjit Mitra. An algorithm for answering queries efficiently using views. In *Proceedings of the 12th Australasian database conference, ADC '01*, pages 99–106, Washington, DC, USA, 2001. IEEE Computer Society.
- [MR97] T.J. Mowbray and W.A. Ruh. *Inside CORBA: distributed object standards and applications*. The Addison-Wesley object technology series. Addison-Wesley, 1997.
- [MST⁺10] Ioana Manolescu, Stefano Spaccapietra, Jens Teubner, Masaru Kitsuregawa, Alain Léger, Felix Naumann, Anastasia Ailamaki, and Fatma Özcan, editors. *EDBT 2010, 13th International Conference on Extending Database Technology, Lausanne, Switzerland, March 22-26, 2010, Proceedings*, volume 426 of *ACM International Conference Proceeding Series*. ACM, 2010.
- [Nag03] Manfred [Hrsg.] Nagl, editor. *Modelle, Werkzeuge und Infrastrukturen zur Unterstützung von Entwicklungsprozessen*. Symposium / Deutsche Forschungsgemeinschaft. Wiley-VCH, Weinheim, 2003.
- [Nie12] Frank Niemann. ECM-Studie 2012: Welche Trends Unternehmen bewegen. public, March 2012. Eine Trendstudie im Auftrag von OpenText, durchgeführt von PAC.
- [NT97] Ikujiro Nonaka and Hirotaka Takeuchi. *Die Organisation des Wissens: Wie japanische Unternehmen eine brachliegende Ressource nutzbar machen*. Campus, Frankfurt am Main, 1997.
- [OAS12] OASIS. OASIS Web Services Business Process Execution Language (WSBPEL) TC @ONLINE, 2012.

- [OBAB11] J. Ovtcharova, V. Bittel, R. Awad, and A. Burger. A web-based collaboration system for mass customization (e-custom). *Berliner Kreis News*, 16:16–17, 2011.
- [OBB10] J. Ovtcharova, V. Bittel, and A. Burger. Ontologiebasiertes Informationsmodell zur Strukturierung von Produktwissen - Repräsentation und kontextorientierte Bereitstellung von Produktwissen am Beispiel Vertrieb. *Berliner Kreis News*, 15:10–11, 2010.
- [OBM11] J. Ovtcharova, V. Bittel, and Dimitris Mourtzis. Ein webbasiertes kollaborationssystem für das mass customization-konzept. *ZWF - Zeitschrift für wirtschaftlichen Fabrikbetrieb*, 106, 2011.
- [OJ94] Jivka Ovtcharova and Uwe Jasnoch. Feature-based design and consistency management in cad applications: a unified approach. *Adv. Eng. Softw.*, 20:65–73, March 1994.
- [Olo08] Carl W. Olofson. Worldwide data integration and access software 2008-2012 forecast. *IDC*, 30, April 1, 2008.
- [OMG12a] OMG. Product Data Management Enablers OMG Formally Released Versions of PDME, Object Management Group @ONLINE. <http://www.omg.org/spec/PDME/1.3/> Stand vom 05.04.2012, April 2012.
- [OMG12b] OMG. Relationship Service OMG Formally Released Versions of REL, Object Management Group @ONLINE, 2012.
- [ÖV11] M. Tamer Özsu and Patrick Valduriez. *Principles of Distributed Database Systems, 3rd Edition*. Prentice-Hall, 2011.
- [Pas04] Thomas B. Passin. *Explorer's Guide to the Semantic Web*. Manning, Greenwich, 2004.
- [PH01] Rachel Pottinger and Alon Y. Halevy. Minicon: A scalable algorithm for answering queries using views. *VLDB J.*, 10(2-3):182–198, 2001.
- [PRCD94] Oscar Pastor, Isidro Ramos, Jose Cuevas, and Jaume Devesa. Oasis 2.0: An object definition language for object oriented databases. In *GULP-PRODE (2)*, 1994.
- [Pro02] ProSTEP. Pdtnet-product data technology and communication in an oem and supplier network - joint development project with the automotive industry. ProSTEP Symposium, 2002.
- [PRR06] G. Probst, S. Raub, and K. Romhardt. *Wissen managen - wie Unternehmen ihre wertvollste Ressource optimal nutzen*. Dr. Th. Gabler Verlag, 2006.

- [PSWW05] M. Pohl, M. Steinbach, C. Weber, and H. Werner. Neue wege im wissensmanagement. *CAD/CAM Report*, 24 7/8:46–49, 2005.
- [PUMH10] Héctor Pérez-Urbina, Boris Motik, and Ian Horrocks. Tractable query answering and rewriting under description logic constraints. *J. of Applied Logic*, 8(2):186–209, 2010. DOI 10.1016/j.jal.2009.09.004.
- [Pup93] Frank Puppe. *Systematic introduction to expert systems - knowledge representations and problem-solving methods*. Springer, 1993.
- [Qia96] Xiaolei Qian. Query folding. In *In ICDE*, pages 48–55, 1996.
- [Qui67] Ross M. Quillian. Word concepts: A theory and simulation of some basic semantic capabilities. *Behavioral Science*, 12:410–430, 1967.
- [RDH⁺04] Alan L. Rector, Nick Drummond, Matthew Horridge, Jeremy Rogers, Holger Knublauch, Robert Stevens, Hai Wang, and Chris Wroe. Owl pizzas: Practical experience of teaching owl-dl: Common errors & common patterns. In *EKAW*, pages 63–81, 2004.
- [Rei77] Raymond Reiter. On closed world data bases. In *Logic and Data Bases*, pages 55–76, 1977.
- [RLSR⁺06] Uwe Radetzki, Ulf Leser, S. C. Schulze-Rauschenbach, J. Zimmermann, Jens Lüssem, Thomas Bode, and Armin B. Cremers. Adapters, shims, and glue - service interoperability for *in silico* experiments. *Bioinformatics*, 22(9):1137–1143, 2006.
- [RMC11] Mariano Rodriguez-Muro and Diego Calvanese. Dependencies: Making ontology based data access work. In *AMW*, 2011.
- [RN10] Stuart J. Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach (3. internat. ed.)*. Pearson Education, 2010.
- [Roo02] N.F.M. Roozenburg. *Engineering design synthesis: understanding, approaches, and tools*, chapter Defining Synthesis: On the Senses and the Logic of Design Synthesis, pages 1 – 16. Springer, 2002.
- [Rud98] S Rude. *Wissensbasiertes Konstruieren*. Habilitationsschrift, Universität Karlsruhe (TH), 1998.
- [SA94] Rolf Socher-Ambrosius. *Deduktionssysteme*. BI-Wissenschaftsverlag, 1994.
- [Say99] A. Sayegh. *Corba.: Standard, Spezifikationen, Entwicklung*. O’Reilly, 1999.

- [SBF98] R. Studer, V. R. Benjamins, and D. Fensel. Knowledge engineering: principles and methods. *Data Knowledge Engineering*, 25(1-2):161 – 197, 1998.
- [SBF⁺08] Carla Sadtler, Srinivasa Rao Borusu, Sergiy Fastovets, Thalia Hooker, Ernese Norelus, Fabio Paone, and Dong Yu. *Getting Started with IBM WebSphere Process Server and IBM WebSphere Enterprise Service Bus*. IBM Redbooks, 2008.
- [Sch06] M. Schenk. *Virtual Reality und Augmented Reality zum Planen, Testen und Betreiben technischer Systeme: 9. IFF-Wissenschaftstage, Juni 2006, [Magdeburg ; Tagungsband]*. IFF-Wissenschaftstage. IFF, 2006.
- [Sch07] M. Schenk. *Virtual Reality und Augmented Reality zum Planen, Testen und Betreiben technischer Systeme: 10. IFF-Wissenschaftstage, 27. - 28. Juni 2007, [Magdeburg ; Tagungsband]*. IFF-Wissenschaftstage. IFF, 2007.
- [SCS03] Kai-Uwe Sattler, Stefan Conrad, and Gunter Saake. Interactive example-driven integration and reconciliation for accessing database federations. *Inf. Syst.*, 28(5):393–414, 2003.
- [SH09] Luciano Serafini and Martin Homola. Modular knowledge representation and reasoning in the semantic web. In *Semantic Web Information Management*, pages 147–181. 2009.
- [She84] John C. Shepherdson. Negation as failure: A comparison of clark’s completed data base and reiter’s closed world assumption. *J. Log. Program.*, 1(1):51–79, 1984.
- [SK05] B. Schäppi and M. Kirchgeorg. *Handbuch Produktentwicklung*. Hanser, 2005.
- [SM01] K. Schott and R. Mäurer. Auswirkungen von eai auf die it-architektur in unternehmen. *Information Management & Consulting*, 16, 2001.
- [SPD92] Stefano Spaccapietra, Christine Parent, and Yann Dupont. Model independent assertions for integration of heterogeneous schemas. *VLDB J.*, 1(1):81–126, 1992.
- [SS11] Ashish Sabharwal and Bart Selman. S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, Third Edition. *Artif. Intell.*, 175(5-6):935–937, 2011.
- [SSFH08] Steffen Staab, Hans-Peter Schnurr, Thomas Franz, and Daniel Hansch. Semantische technologien und auswirkung auf informations- und wissensmanagement-systeme. *IM - die Fachzeitschrift für Information Management & Consulting*, 2, 2008.

- [SSH10] Gunter Saake, Kai-Uwe Sattler, and Andreas Heuer. *Datenbanken - Konzepte und Sprachen*, volume 4. mitp-Verlag, 2010.
- [SSS91] Manfred Schmidt-Schauß and Gert Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1 – 26, 1991.
- [ST06] York Sure and Christoph Tempich. Wissensvernetzung in organisationen. In Tassilo Pellegrini and Andreas Blumauer, editors, *Semantic Web*, X.media.press, pages 291–305. Springer Berlin Heidelberg, 2006.
- [Sta73] Herbert Stachowiak. *Allgemeine Modelltheorie*. Springer-Verlag, Wien, New York, 1973.
- [STK02] James Snell, Doug Tidwell, and Pavel Kulchenko. *Webservice-Programmierung mit SOAP*. O'Reilly, 2002.
- [SZ03] Christof Schmitz and Betty Zucker. *Wissensmanagement. Schnelleres Lernen in Unternehmen*. Metropolitan Verlag, 2003.
- [TBHH07] R. Troncy, W. Bailer, M. Hausenblas, and M. Höffernig. Vamp: Semantic validation for mpeg-7 profile descriptions. Technical report, *Centrum voor Wiskunde en Informatica*, 2007.
- [The01] S. Thessaris. *Questions and Answers: Reasoning and Querying in Description Logic*. PhD thesis, University of Manchester, 2001.
- [Ull97] Jeffrey D. Ullman. Information integration using logical views. In *ICDT*, pages 19–40, 1997.
- [Ull00] Jeffrey D. Ullman. Information integration using logical views. *Theor. Comput. Sci.*, 239(2):189–210, 2000.
- [Usc03] Michael Uschold. Where are the semantics in the semantic web? *AI Magazine*, 24(3):25–36, 2003.
- [VDA05] VDA-4958. Langzeitarchivierung (LZA) nicht-zeichnungsbasierter, digitaler Produktdaten. Technical report, Verband der Automobilindustrie, 2005.
- [VDI02] VDI-2219. Informationsverarbeitung in der Produktentwicklung - Einführung und Wirtschaftlichkeit von EDM/PDM-Systemen. Technical report, Verein Deutscher Ingenieure, 2002.

- [VDI09] VDI-5610. VDI-Richtlinie 5610 Blatt 1: Wissensmanagement im Ingenieurwesen - Grundlagen, Konzepte, Vorgehen. Technical report, Verein Deutscher Ingenieure, March 2009.
- [vdM98] Ron van der Meyden. Logical approaches to incomplete information: A survey. In *Logics for Databases and Information Systems*, pages 307–356, 1998.
- [VSA12] José M. González Vázquez, Jürgen Sauer, and Hans-Jürgen Appelrath. Methoden zum management von informationsquellen für die unterstützung von softwareproduktmanagern in der energiewirtschaft - ein referenzmodellkatalog für die energiewirtschaft. *Wirtschaftsinformatik*, 54(1):3–15, 2012.
- [W3C] W3C. work revolves around the standardization of web technologies. W3C Standards available online at <http://www.w3.org/standards/semanticweb/>.
- [W3C12] W3C. World Wide Web Consortium, XML in 10 Punkten @ONLINE, 2012.
- [WJ06] Daniel Wuwer and Ulrich Jumar. Studie - embedded web-server. public, February 2006.
- [WMT02] Junhu Wang, Michael Maher, and Rodney Topor. Rewriting general conjunctive queries using views. *Aust. Comput. Sci. Commun.*, 24:197–206, January 2002.
- [Wol03] Frank Wolf. *SAP Web Application Server - Internet-Anwendungen entwickeln mit ABAP, HTML und JavaScript*. dpunkt.verlag, 2003.
- [Woo91] W.A. Woods. *Understanding subsumption and taxonomy: a framework for progress*. Harvard University, Center for Research in Computing Technology, Aiken Computer Laboratory, 1991.
- [WRW01] Thomas Winkeler, Ernst Raupach, and Lothar Westphal. Enterprise Application Integration als Pflicht vor der Business-Kür. *Die Fachzeitschrift für Information Management & Consulting*, 1, 2001.
- [WWPS05] H. Werner, C. Weber, M. Pohl, and Mi. Steinbach. Innovatives wissensmanagement auf basis semantischer netze. *ZWF Zeitschrift für wirtschaftlichen Fabrikbetrieb*, 100:212–218, 2005.
- [Zol10] Evgeny Zolin. Description Logic Complexity Navigator, June 2010.

C. Index

Symbols

(GaV)-Ansatz	17
(LaV)-Ansatz	17
.NET Framework	49

A

Abduktion	29
Accelerators	54
Agent	52
Aggregation	152
Anfragekörper	39
Anfragekopf	39
Anfrageumschreibung <i>siehe</i> Umschreibung	
Annahme einer geschlossenen Welt	27
Annahme einer offenen Welt	27
Annotationseigenschaften	181
application protocols	62
Applikationsinfrastruktur	54
Auffalten	124
Axiom	28

B

Backbone	68
Baumstruktur	97
Belegung	<i>siehe</i> Interpretation
Beschreibungslogiken	34
BizTalk	52
BizTalk-Server	52
body	<i>siehe</i> Anfragekörper
business on demand	53
Business Process Execution Language ..	57

C

Class	182
closed world assumption ... <i>siehe</i> Annahme einer geschlossenen Welt	
Common Language Runtime	50
Common Object Request Broker Architec- ture	47
CORBA	47
Common Type Systems	49
Conformance Testing Methodology and Fra- mework	62

D

Daten	9
Deduktion	29
Description Logics	<i>siehe</i> Beschreibungslogiken
Description Methods	62
Dienstanbieter	48
Dienstmakler	48
Dienstinachfrager	48
Durchführbarkeit	31

E

Effizienz	31
Engineering Memory Module	75
Engineering Ontologien	74
entailment ... 26, <i>siehe</i> Folgerungsproblem	
Enterprise Application Integration	51
Enterprise Resource Planing	68
Entscheidbarkeit	30

Erfüllbarkeit	26	Inklusionsabhängigkeit	101
ERP-System <i>siehe</i> Enterprise Resource Planing		Instanz	101
Extension	15	Integrationskonflikt	14
Extrahierung	100	Beschreibungskonflikt	15
F		Extensionaler Konflikt	15
Folgerung	26	Heterogenitätskonflikt	15
Folgerungsproblem	41	Struktureller Konflikt	15
formal	33	Integrationsystem	13
Formel	26	Interpretation	26
Framework Class Library	50	iViP	71
Fremdschlüsselbeziehung	101	iViP-Projekt	<i>siehe</i> iViP
G		K	
General Inter-ORB-Protocl	47	Kalkül	28
Generalisierung	152	Komplexitätstheorie	31
Global-as-View	<i>siehe</i> (GaV)-Ansatz	Komplexität	31
H		Konjunktive Anfrage	39
head	<i>siehe</i> Anfragekopf	konsistent	120
heterogen	<i>siehe</i> Heterogenität	Konsistenz	38
Heterogenität	16	Konzeptkonsistenz	38
Hub-and-Spoke-Architektur	51	Konzeptinklusion	38, 105
I		Subsumption	38
Implementation Methods	62	Konzeptkonsistenz	39
Individuals	183	Konzeptzugehörigkeit	38, 39
Induktion	28	Korrektheit	30
Inferenzproblem	38	Korrespondenz	
Inferenzregel	28	Anfragekorrespondenz	111
informal	33	Mapping-Beziehung	111
Informationen	9	L	
Informationsintegration	12	Literal	26
materialisierte Integration	13	Local-as-View	<i>siehe</i> (LaV)-Ansatz
data warehouse	14	Logik	24
virtuelle Integration	13	Attributive Language with Complement (ALC)	35
Mediator-basierten Informationssystem 14		Logische Folgerung	<i>siehe</i> Folgerung
Informationsintegrationssystem	18	Logische Formel	<i>siehe</i> Formel

M	
Mappings	90
Materialisierung	153
Message Box	52
Metadaten	89
Metawissen	100
Middleware-basierte Integration	46
Modell	
Mathematische Modell	118
Modellmerkmale	118
Abbildungsmerkmal	118
Pragmatisches Merkmal	118
Verkürzungsmerkmal	118
Modus ponens	29
Modus tollens	29
most general unifier	<i>siehe</i> Unifikator
N	
Nachricht	52
negation as failure	27
negation as unsatisfiability	27
Normalform	<i>siehe</i> Normalisierung
Normalisierung	110
Normmenreihe	62
Nutzdaten	89
O	
Object Request Broker	47
Ontologie	32
Ontologieschemata	89
Ontologiestruktur	<i>siehe</i> Ontologie
open world assumption	<i>siehe</i> Annahme einer offenen Welt
Orchestration	52
OWL	<i>siehe</i> Web Ontology Language
P	
P2P-Integration	45
PDM Enabler	
<i>siehe</i> Product Data Management Enabler	
PDM-System	
<i>siehe</i> Product Data Management	
PDM-Systemarchitektur	67
PDTnet	72
Probabilistic Ontology Model	99
Product Data Management	66
Produktdatenintegration	61
Produktdatenmodell	61
Produktentwicklungswissen	12
Produktinformationen	12
Produktmerkmale	60
Beschaffenheitsmerkmale	60
Funktionsmerkmale	60
Relationsmerkmale	60
Produktmodell	<i>siehe</i> Produktdatenmodell
Produktwissen	12
Properties	183
Property Restrictions	184
Punkt-zu-Punkt-Verbindung	45
Q	
query rewriting <i>siehe</i> Anfrageumschreibung	
R	
Reasoning	23
Receive Adapter	52
Receive Pipeline	52
Relational Data Minig Module	75
Relationales Datenbankschema	100
Resolution	29
Ressourcen	62
application resources	62
generic resources	62
integrated resources	62
S	
SAP NetWeaver	69

Enterprise Services Architecture	69	tractability	<i>siehe</i> Durchführbarkeit
Schemaintegration	14	Transformation	97
Semantic Engineering Module	74	Tripel	177
Semantic Virtual Reality Module	75	Typ einer Korrespondenz	114
Semantic Web	58	äquivalent	115
Semantik	23	korrekt	114
Semantische Technologien	58	vollständig	114
semi-formal	33	U	
semi-informal	33	Umschreiben einer Anfrage	123
Send Adapter	52	Äquivalente Umschreibung	125
Service-orientierte Architektur		Maximale Umschreibung	125
SOA-Tempel	56	Partielle Umschreibung	125
Serviceorientierte Architektur	56	Perfektes Umschreiben	123
SEVENPRO	74	Umschreibung	19
sichere Antwort	40	unfolding	<i>siehe</i> Auffalten
Sichten-basierte Anfragebearbeitung	20	Unicode	58
Sichten-basierte Anfragebeantwortung	20	Unifikator	129
Sichten-basierte Anfrageumschreibung	20	Universal Decimal Classification	147
Simple Object Access Protocol	48	Universal Description, Discovery and Integration	48
Standard-Datenmodelle	61	URI	58
Product Data Management Enabler	64	V	
Standard for the Exchange of Product Model Data	62	Variablenzuordnung	128
stored procedures	161	view-based query answering . <i>siehe</i> Sichten-basierte Anfragebearbeitung	
Structural Semantic Interconnections	99	view-based query processing . <i>siehe</i> sichten-basierte Anfragebearbeitung	
Structured Query Language	119	view-based query rewriting . <i>siehe</i> Sichten-basierte Anfrageumschreibung	
Strukturelemente	97	Virtual Reality Reasoning Module	75
Attribut	97	Vollständigkeit	30
Entität	97	W	
System	97	Web Ontology Language	181
Substitution	<i>siehe</i> Variablenzuordnung	Webservice	48
Subsumptionsrelation	104	Webservices Description Language	48
Syntax	23	WebSphere Business Integration	53
T			
TBox	105		
Term	25		
The Semaril	75		

Wissen.....	9
Wissensarten	11
Wissensbegriff.....	9
Wissensbasis	26
Beschreibungslogische Wissensbasis	34
Wissenspraktiker	92
Wissensrepräsentation.....	23
Wissensrepräsentationsformalismus ...	104
\mathcal{L}_{tripod}	104