

GESCHÄFTSPROZESSORIENTIERTE SERVICE DISCOVERY AUF BASIS HETEROGENER SEMANTISCHER BESCHREIBUNGEN

vorgelegt von
DIPLOM-ING.
JANNIS RAKE-REVELANT
aus Berlin

von der Fakultät IV - Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
– Dr.-Ing. –
genehmigte Dissertation

Promotionsausschuss:

Vorsitzender: PROF. DR. VOLKER MARKL
Gutachter: PROF. DR. HERMANN KRALLMANN
Gutachter: PROF. DR. YORK SURE-VETTER

Tag der wissenschaftlichen Aussprache: 17. April 2013

Berlin 2013
D 83

Zusammenfassung

Zur Sicherung nachhaltiger Wettbewerbsvorteile ist es für Unternehmen unabdingbar, ihre Organisation effizient und effektiv anpassen zu können. In der Vergangenheit hat sich die Flexibilität von Geschäftsprozessen als ein maßgeblicher Einflussfaktor für diese Anpassungen herausgestellt. Ein wichtiger Treiber zur Erhöhung dieser Flexibilität ist die Informationstechnologie (IT). Eines der Hindernisse für den erfolgreichen Einsatz von Technologien bei der Erhöhung der Geschäftsprozessflexibilität stellt die fehlende Abstimmung zwischen Fach- und IT-Seite dar, welche oft als *Business-IT-Gap* bezeichnet wird. Die Anpassungen, welche von der Fachseite benötigt werden, ändern sich teilweise schneller als die IT diese umsetzen kann.

Zur Reduzierung des *Business-IT-Gaps* und der damit erhofften Erhöhung der Geschäftsprozessflexibilität wird seit einiger Zeit der Einsatz sogenannter serviceorientierter Architekturen (SOA) diskutiert. Der Einsatz und die Rekombination von feingranularen Services verspricht neben einer gezielten Unterstützung der Prozesse eines Unternehmens auch die Reduzierung von Zeit und Kosten in neuen Entwicklungsprojekten durch die Wiederverwendung von Services. Grundlage dieser Versprechen ist zumindest teilweise die Annahme, dass in der Zukunft eine Vielzahl von Serviceanbietern eine große Zahl von Services unterschiedlicher Funktionalität zur Verfügung stellt. Ausgehend von einem solchen Szenario wäre es relativ einfach möglich, die passenden Services auszuwählen, um eine benötigte Funktionalität mit wenigen Anpassungen zu realisieren. Eine fundamentale Voraussetzung für diesen Prozess ist jedoch ein effizientes und effektives Verfahren, Services mit der passenden Funktionalität auffinden zu können. Um eine solche sogenannte Service Discovery umsetzen zu können, verspricht der Einsatz semantischer Servicebeschreibungen die Erhöhung der Ergebnisqualität und in Zukunft auch eine (halb-)automatische Komposition der Services. Existierende Ansätze im Bereich der semantischen Service Discovery gehen allerdings von einer homogenen Wissensbasis zur Beschreibung des Services aus. In der Realität ist ein solches Szenario höchst unrealistisch.

Die vorliegende Arbeit entwickelt einen Ansatz für die geschäftsprozessorientierte Service Discovery auf Basis heterogener Beschreibungen und die Machbarkeit dieses Ansatzes in Form einer prototypischen Implementierung. Auf Basis aktueller Forschungsergebnisse und Erfahrungen aus Praxisprojekten werden geschäftsprozessorientierte Servicebeschreibungen entwickelt und ein Konzept zum Umgang mit Heterogenität durch den Einsatz automatisierter Alignment-Werkzeuge. Die Implementierung des Ansatzes zeigt zum einen dessen Machbarkeit und ermöglicht zum anderen eine quantitative Auswertung der Qualität der heterogenen Service Discovery. Weiterhin wird auf Basis von Fallstudienbeispielen die Praxistauglichkeit des Ansatzes demonstriert. Dabei kann gezeigt werden, dass der Einsatz von Alignment-Werkzeugen die Qualität der Service Discovery in heterogenen Umgebungen deutlich verbessert.

Abstract

Companies must be able to carry out organizational adjustments effectively and efficiently to secure sustainable competitive advantages. For these necessary adjustments flexibility of business processes constitutes a key influence factor. Information technology (IT) has proved to be a crucial driver of business process flexibility in dynamic business environments. So far, the potential of IT for the organizational adaptation could not be fully accessed, which regularly appears as a lack of alignment between IT and the organization and/or the business processes – the so-called business-IT gap. Requirements defined by the business often change faster than the IT department can carry out these changes.

In recent years the paradigm of serviceoriented architectures (SOA) has surfaced and is being discussed as a way to reduce the business-IT gap and thus increase the flexibility of business processes. Deploying and recombining fine-grained services promises, amongst systematic support of an organisation's processes, to also reduce time and costs of development projects by reusing existing services. A fundamental to this promise is at least partially the assumption that, in the near future, a multitude of service providers will offer a myriad of services with different functionalities. Based on the described scenario it would be easy to select and combine a number of services to achieve a desired functionality within minimal effort of customization. A fundamental prerequisite to such an approach is an efficient and effective way of finding the services with the right functionality. In order to achieve this so-called service discovery, Semantic Web Services promise to increase quality of search results and, in the future, also (semi-)automatic compositions of services. However, existing approaches often depend on homogenous knowledge bases to describe the services. In reality such homogeneity seems highly unrealistic.

The dissertation at hand develops an approach for business-process-oriented service discovery based on heterogeneous descriptions and proves its feasibility by providing a prototypical implementation. Based on state-of-the-art research and experiences gained in the industry business-process-oriented service descriptions are designed and a concept to deal with heterogeneity by utilizing automated ontology alignment tools is devised. The prototype on the one hand shows the feasibility of the approach while on the other hand allows a quantitative analysis of the quality of such a heterogeneous service discovery. Additionally, the practical use of the approach is demonstrated by two case studies. Thus it can be shown that by employing alignment tools the quality of a service discovery in heterogeneous environments can be significantly increased.

Gliederung

Zusammenfassung	III
Abstract	V
Gliederung	VII
Inhaltsverzeichnis	IX
1 Einführung	1
2 Definition grundlegender Begriffe	10
3 Stand der Forschung	51
4 Geschäftsprozessorientierte Service Discovery	131
5 Prototypische Implementierung	187
6 Evaluierung	210
7 Zusammenfassung und Ausblick	219
Abbildungsverzeichnis	225
Tabellenverzeichnis	231
Abkürzungsverzeichnis	233
Verzeichnis eigener Veröffentlichungen	239
Literatur	241
Anhang A	259

Inhaltsverzeichnis

Zusammenfassung	III
Abstract	V
Gliederung	VII
Inhaltsverzeichnis	IX
1 Einführung	1
1.1 Zielsetzung.....	2
1.1.1 Motivation	2
1.1.2 Ziel der Dissertation.....	3
1.1.3 Abgrenzung des Ziels.....	4
1.2 Forschungsmethodik.....	4
1.3 Struktur der Arbeit.....	7
1.4 Beiträge für Wissenschaft und Praxis	8
2 Definition grundlegender Begriffe	10
2.1 Serviceorientierte Architekturen.....	10
2.1.1 Serviceorientierung.....	10
2.1.2 Unternehmensarchitekturen.....	11
2.1.3 Begriffsdefinition SOA.....	12
2.1.4 Prinzipien der Serviceorientierung.....	14
2.1.4.1 Grundprinzipien.....	14
2.1.4.2 Zielprinzipien.....	18
2.1.5 Technische Grundlagen einer SOA.....	20
2.1.5.1 Web Service und Web Service Description Language	20
2.1.5.2 Technische Komponenten einer SOA	21
2.2 Semantic Web und Semantic Web Services.....	22

2.2.1	Ontologien.....	23
2.2.1.1	Repräsentation von Wissen.....	24
2.2.1.2	Ontologien in der Informationsverarbeitung.....	24
2.2.1.3	Formen von Ontologien.....	24
2.2.2	Semantic Web	25
2.2.2.1	Ontologiesprachen.....	25
2.2.3	Semantic Web Services (SWS)	28
2.2.3.1	Web Ontology Language for Services (OWL-S).....	30
2.2.3.2	Web Service Modeling Ontology (WSMO).....	32
2.2.3.3	Semantic Annotations for WSDL and XML Schema (SAWSDL)	34
2.3	Ontology Matching und Alignment.....	35
2.3.1	Ontology Matching	36
2.3.2	Ontology Alignment.....	38
2.4	Geschäftsprozessorientierung und -management	41
2.4.1	(Geschäfts-)Prozessorientierung.....	42
2.4.2	Geschäftsprozessmanagement.....	43
2.4.3	Geschäftsprozessmodellierung.....	45
2.4.3.1	Modellbegriff	45
2.4.3.2	Definition von Geschäftsprozessmodellen, -modellierung und -modellierungssprachen.....	46
2.4.3.3	Modellierungssprachen	46
2.4.3.4	Ausführbarkeit von Geschäftsprozessen	49
3	Stand der Forschung	51
3.1	Service Discovery.....	51
3.1.1	Ebenen der Service Discovery	52
3.1.1.1	Keyword Discovery.....	52
3.1.1.2	„Leichtgewichtige“ Semantische Discovery	52
3.1.1.3	„Schwergewichtige“ Semantische Discovery.....	53
3.1.2	Bestehende Ansätze	53

3.1.2.1	Discovery mit OWL-S	53
3.1.2.2	Discovery mit WSDL-S/SAWSDL	59
3.1.2.3	Discovery mit WSMO	64
3.2	Semantisches Geschäftsprozessmanagement	71
3.2.1	Modellierung semantischer Geschäftsprozesse	74
3.2.1.1	Semantische Geschäftsprozessmodellierungssprachen.....	75
3.2.1.2	Umsetzung der Modellierung semantischer Geschäftsprozesse..	78
3.2.2	Implementierung semantischer Geschäftsprozesse.....	81
3.2.3	Ausführung semantischer Geschäftsprozesse	85
3.2.4	Analyse semantischer Geschäftsprozesse	89
3.2.4.1	Ontologische Grundlage der semantischen Geschäftsprozessanalyse.....	91
3.2.4.2	Semantisches Prozess-Mining	93
3.3	Ontology Matching, Alignment und Integration.....	95
3.3.1	Ontology-Matching-Techniken und Strategien	95
3.3.1.1	Klassifikation von Ontology-Matching-Techniken.....	96
3.3.1.2	Grundtechniken des Matchings	98
3.3.1.3	Matching Strategien.....	108
3.3.2	Evaluierung von Ontology Matching und Alignment	111
3.3.3	Werkzeuge.....	114
3.3.3.1	Falcon-AO	115
3.3.3.2	Lily	117
3.3.3.3	Automated Semantic Matching of Ontologies with Verification (ASMOV).....	119
3.4	Heterogene Service Discovery	122
3.4.1	Semantische Service Discovery in einer Multi-Ontologie-Umgebung	122
3.4.2	Nutzung von Konzept-Matching zum effizienten Umgang mit heterogenen Ontologien	124
3.4.3	Implementierung von WSMO-Mediatoren in GLUE.....	126
3.4.4	Umgang mit heterogenen Ontologien in METEOR-S.....	129

4	Geschäftsprozessorientierte Service Discovery	131
4.1	Geschäftsprozessorientierte Service Beschreibungen.....	132
4.1.1	Serviceklassifizierung.....	132
4.1.2	Ableitung relevanter Informationen.....	134
4.1.3	Beschreibung der Geschäftsobjekte.....	137
4.1.3.1	Geschäftsobjekte und Geschäftsartefakte	137
4.1.3.2	Abgrenzung zu Eingabe- und Ausgabeparametern.....	141
4.1.4	Beschreibung der Funktionalität	142
4.1.4.1	Statusinformationen	142
4.1.4.2	Vorbedingungen und Effekte.....	144
4.1.4.3	Einschränkungen.....	147
4.1.5	Nutzung bestehender formaler Beschreibung	148
4.1.5.1	WSMO.....	148
4.1.5.2	OWL-S.....	151
4.1.5.3	SAWSDL.....	153
4.2	GPM-getriebene Service Discovery	154
4.2.1	Zielgruppe und Vorteile	154
4.2.2	Nutzung von Kontextinformationen	156
4.2.2.1	Extraktion von Informationen mittels Natural Language Processing.....	159
4.2.3	Integration in die Modellierung.....	162
4.2.3.1	Darstellung, Manipulation und Extraktion verwertbarer Informationen	163
4.2.3.2	Aufruf und Formulierung der Service-Anfrage	164
4.2.3.3	Auswahl und Verwendung von Services.....	165
4.3	Heterogene Service Discovery.....	166
4.3.1	Grundlegende Service Discovery	166
4.3.2	Herstellen von Interoperabilität.....	171
4.3.3	Umsetzung des Ontology Matchings.....	176
4.3.4	Modularisierung zur Verbesserung der Discovery	178

4.4	Benutzerinteraktion.....	180
4.4.1	Kontrolliertes Vokabular.....	180
4.4.2	Kontextorientierte Annotation	182
4.4.3	Autovervollständigung.....	183
4.4.4	Weiterverwendung.....	185
5	Prototypische Implementierung	187
5.1	Umsetzung GPM-getriebener heterogener Service Discovery.....	187
5.1.1	Anforderungen.....	187
5.1.2	Architektur	191
5.1.2.1	Service & Ontologie Repository.....	192
5.1.2.2	Reasoning-Komponente.....	194
5.1.2.3	Alignment-Komponente.....	195
5.1.2.4	Service-Discovery-Komponente.....	198
5.1.2.5	BPMN-Editor-Komponente.....	200
5.2	Modulares Ontology Matchmaking System	204
5.2.1	Architektur.....	204
5.2.2	Implementierung der Komponenten.....	205
5.2.3	Benutzerinteraktion.....	208
6	Evaluierung	210
6.1	Quantitative Auswertung.....	210
6.1.1	Metriken und Testdaten.....	210
6.1.2	Ergebnisse.....	211
6.2	Qualitative Auswertung.....	214
6.2.1	Fallstudien.....	214
6.2.1.1	Praxisprojekt Telekommunikation und Unternehmenssoftware.....	215
6.2.1.2	Praxisprojekt Telekommunikationsanbieter	216
6.2.2	Vergleich mit bestehenden Ansätzen.....	217

7 Zusammenfassung und Ausblick	219
7.1 Zusammenfassung	219
7.2 Ausblick.....	222
Abbildungsverzeichnis	225
Tabellenverzeichnis	231
Abkürzungsverzeichnis	233
Verzeichnis eigener Veröffentlichungen	239
Literatur	241
Anhang A	259

I Einführung

Zur Sicherung nachhaltiger Wettbewerbsvorteile ist es für Unternehmen unabdingbar, ihre Organisation effizient und effektiv anpassen zu können (Cyert und March (1963)). In der Vergangenheit hat sich die Flexibilität von Geschäftsprozessen als ein maßgeblicher Einflussfaktor für diese Anpassungen herausgestellt (Moitra und Ganesh (2005); Regev et al. (2007)). Ein wichtiger Treiber zur Erhöhung dieser Flexibilität ist die Informationstechnologie (IT) (Clemons (1986); King et al. (1989)). Eines der Hindernisse für den erfolgreichen Einsatz von Technologien bei der Erhöhung der Geschäftsprozessflexibilität stellt die fehlende Abstimmung zwischen Fach- und IT-Seite dar, welche oft als *Business-IT-Gap* bezeichnet wird (Masak (2006); Aier und Winter (2009)). Die Anpassungen, welche von der Fachseite benötigt werden, ändern sich teilweise schneller als die IT diese umsetzen kann. (Holschke et al. (2010), S. 3)

Zur Reduzierung des *Business-IT-Gaps* und der damit erhofften Erhöhung der Geschäftsprozessflexibilität wird seit einiger Zeit der Einsatz sogenannter serviceorientierter Architekturen (SOA) diskutiert (Offermann et al. (2007)). Der Einsatz und die Rekombination von feingranularen Services verspricht neben einer gezielten Unterstützung der Prozesse eines Unternehmens auch die Reduzierung von Zeit und Kosten in neuen Entwicklungsprojekten durch die Wiederverwendung von Services. Grundlage dieser Versprechen ist zumindest teilweise die Annahme, dass in der Zukunft eine Vielzahl von Serviceanbietern eine große Zahl von Services unterschiedlicher Funktionalität zur Verfügung stellt. Ausgehend von einem solchen Szenario wäre es relativ einfach möglich, die passenden Services auszuwählen, um eine benötigte Funktionalität mit wenigen Anpassungen zu realisieren. Eine fundamentale Voraussetzung für diesen Prozess ist jedoch ein effizientes und effektives Verfahren, Services mit der passenden Funktionalität auffinden zu können. Um eine solche sogenannte Service Discovery umsetzen zu können, verspricht der Einsatz semantischer Servicebeschreibungen die Erhöhung der Ergebnisqualität und in Zukunft auch eine (halb-)automatische Komposition der Services (Polleres et al. (2006)). Existierende Ansätze im Bereich der semantischen Service Discovery gehen allerdings von einer homogenen Wissensbasis zur Beschreibung des Services aus. In der Realität ist ein solches Szenario höchst unrealistisch. Es existieren zwar Konzepte zur Lösung einer Heterogenität, meist sind diese aber mit einem nicht unerheblichen manuellen Aufwand verbunden. (Rake et al. (2009))

Die vorliegende Arbeit beschreibt einen Ansatz für die geschäftsprozessorientierte Service Discovery auf Basis heterogener Beschreibungen und die Machbarkeit dieses Ansatzes in Form einer prototypischen Implementierung. In dieser Einführung wird dazu zunächst die *Zielsetzung* der Dissertation *motiviert, formuliert und abgegrenzt*. Anschließend werden *Forschungsmethodik* sowie die resultierende *Struktur der Arbeit* vorgestellt. Abschließend erfolgt eine Diskussion der *Beiträge* der Arbeit für *Wissenschaft und Praxis*.

I.1 Zielsetzung

In den folgenden Abschnitten soll das Ziel der Dissertation geklärt werden. Zunächst wird anhand eines Anwendungsfalls das Problem heterogener semantischer Servicebeschreibungen motiviert. Anschließend erfolgt die Definition des Ziels der Dissertation und eine Abgrenzung von anliegenden Bereichen.

I.1.1 Motivation

Der Mehrwert von SOAs liegt unter anderem in der Möglichkeit die Geschäftsprozesse eines Unternehmens flexibel und effektiv unterstützen zu können. Zum besseren Verständnis eines möglichen Anwendungsfalls zum Umgang mit heterogenen Servicebeschreibungen wird im Folgenden das Szenario in Abbildung 1 näher erläutert.

Als Anwender ist in diesem Fall ein Geschäftsprozessexperte (GPE) gewählt worden, welcher einen bestehenden oder gerade erstellten Prozess mit passenden Services unterstützen will. Weiterhin ist davon auszugehen, dass eine Anzahl n Serviceanbieter existieren, welche ihre Services semantisch annotiert haben. Zunächst analysiert der GPE den Prozess, um die benötigten Funktionalitäten einzelner Prozessschritte und Teilprozessschritte zu identifizieren. Möglich ist auch, dass der Prozess schon Beschreibungen über die benötigte Funktionalität beinhaltet. In beiden Fällen müssen jedoch die Anforderungen in einem gewissen Maße formalisiert werden, um eine Suche nach Services durchführen zu können. Anschließend kann – meist durch Unterstützung eines passenden Software-Werkzeuges – eine Suche nach passenden Services, also eine Service Discovery, durchgeführt werden.

Im Idealfall würde eine solche Suche problemlos die gewünschten Services liefern, falls Services mit entsprechender Funktionalität existieren. Allerdings ist davon auszugehen, dass die Anbieter der Services nicht unbedingt die gleichen Ansätze zur Annotation der Services verwenden. Erschwerend kommt hinzu, dass auch die Konzepte zur Beschreibung der Services (meist in Form eines kontrollierten Vokabulars) durchaus unterschiedlich sind. Problematisch wird es vor allem falls Konzepte mit gleicher Bedeutung (Semantik) mit unterschiedlichen Begriffen repräsentiert werden.

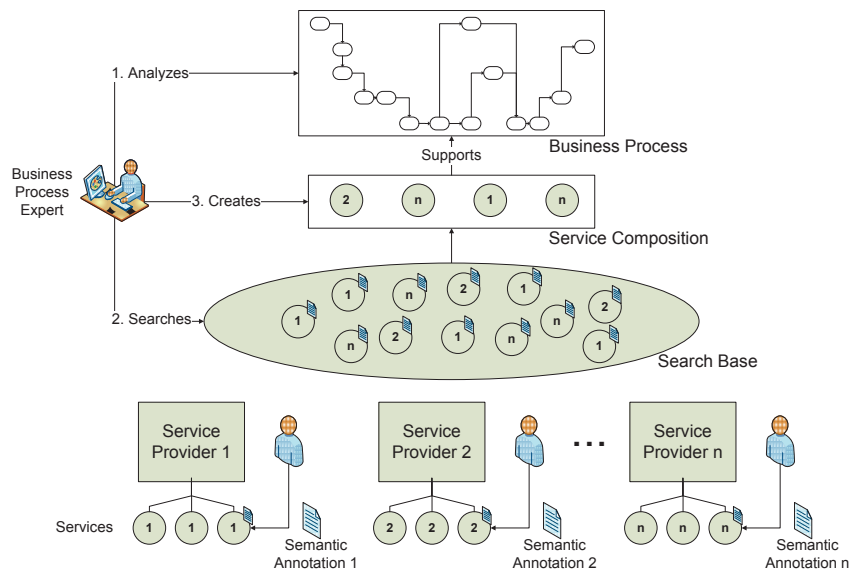


Abbildung 1: Heterogenität von Servicebeschreibungen unterschiedlicher Service-Anbieter

1.1.2 Ziel der Dissertation

Die angestrebte Dissertation hat zum Ziel einen Ansatz zu entwickeln, welcher es ermöglicht fachlich relevante Services in einer SOA effektiv und effizient aufzufinden. Weiterhin gilt die zusätzliche Bedingung, dass diese Services ihre Funktionalität in Form von heterogenen semantischen Beschreibungen bekanntmachen können.

Für die Erreichung dieses Ziels ist es erforderlich, unterschiedliche Teilbereiche zu untersuchen und möglicherweise zu erweitern. Es soll davon ausgegangen werden, dass die aufzufindenden Services direkt oder indirekt die Geschäftsprozesse eines Unternehmens unterstützen. Folglich müssen Aspekte der Geschäftssemantik untersucht werden, vor allem im Hinblick auf die Formulierung von Zielen zur Suche nach Funktionalität auf Geschäftsprozessebene. Weiterhin ist es unabdingbar, unterschiedliche Ansätze zur Beschreibung und somit zur Formalisierung von Servicefunktionalitäten zu untersuchen und zu vergleichen. Dabei gilt es diesen Vergleich auf Basis von Kriterien durchzuführen, welche durch den voraussichtlichen Einsatz in einem heterogenen Serviceumfeld motiviert sind. Es muss zudem gewährleistet werden, dass eine Formulierung von Suchanfragen mit akzeptablem Aufwand und zielgruppengerecht (z.B. Experten der Geschäftsprozessmodellierung) durchgeführt werden kann. Letztlich soll eine Lösung zum expliziten Umgang mit Heterogenität von Servicebeschreibungen konzipiert werden, wobei vor allem auf ein gewisses Maß an Automatisierung Wert gelegt wird.

1.1.3 Abgrenzung des Ziels

Die geplante Dissertation sieht nicht vor, neue Ansätze für die Beschreibung oder das Management von Geschäftsprozessen zu erforschen. Dennoch ist der zu entwickelnde Ansatz durch diese Bereiche motiviert und es werden die benötigten Grundlagen erläutert werden. Dabei werden Ansätze und Technologien zum Einsatz kommen, welche auf dem neuesten Stand der Forschung basieren. Dies bezieht sich unter anderem auf die bereits erwähnte Fragestellung, welche Möglichkeiten zur Festlegung von Semantik in Geschäftsprozessen und der Nutzung dieser im Rahmen der Formulierung von Anforderungen an die Funktionalität von Services existieren.

Zur Beschreibung der Services sollen Erweiterungen bestehender Beschreibungssprache untersucht und umgesetzt werden. Eine Neuentwicklung einer Sprache zur Spezifikation von Ontologien befindet sich aber außerhalb des Forschungsziels. Eine solche Neuentwicklung würde weitreichende Auswirkungen auf Bestandteile wie das Reasoning haben und die Möglichkeit verringern beziehungsweise verhindern bestehende Arbeiten in diesen Bereichen wiederzuverwenden.

Weiterhin ist nicht vorgesehen, eine Ausführungsumgebung für semantische Web Services zu entwickeln. Für eine prototypische Implementierung können entweder bestehende Ausführungsumgebungen für semantische Web Services (SWS) oder traditionelle Orchestrierungs-Engines verwendet werden. Es werden in diesem Bereich jedoch entsprechende Anforderungen formuliert werden.

1.2 Forschungsmethodik

Die Forschungsmethodik der vorliegenden Arbeit richtet sich nach Hevner et al. (2004) und ist somit der sogenannten Design Science des Information Systems Research (ISR) zuzuordnen. Die Design Science, welche ihre Wurzeln vor allem in den Ingenieurwissenschaften hat, versucht Innovationen in verschiedenen Bereichen (Ideen, Praktiken, Produkte) zu generieren, um die Analyse, das Design, die Implementierung, das Management und die Benutzung von Informationssystemen effektiver und effizienter zu gestalten (Hevner et al. (2004), S. 76). Im Bereich der Informationssysteme ist eine klare Trennung zwischen Technologie, Organisation und Menschen allerdings nicht ohne Weiteres möglich. Hevner et al. (2004) schlagen daher das in Abbildung 2 dargestellte Rahmenwerk für die Forschung im Bereich der Informationssysteme (IS) vor, welches kurz vorgestellt werden soll. Anschließend ist eine Beschreibung des Vorgehens der vorliegenden Arbeit anhand des Rahmenwerks möglich.

Als Grundlage des Rahmenwerks dient der Gedanke, dass die Forschung im Bereich von Informationssystemen in einem Spannungsfeld aus Wissenschaft und Praxis existiert. Beiden Bereichen soll gerecht werden, indem zum einen eine praktische Relevanz der entwickelten Theorien und Artefakte und zum anderen eine wissenschaftliche Strenge bei der Entwicklung und Auswertung dieser sichergestellt werden.

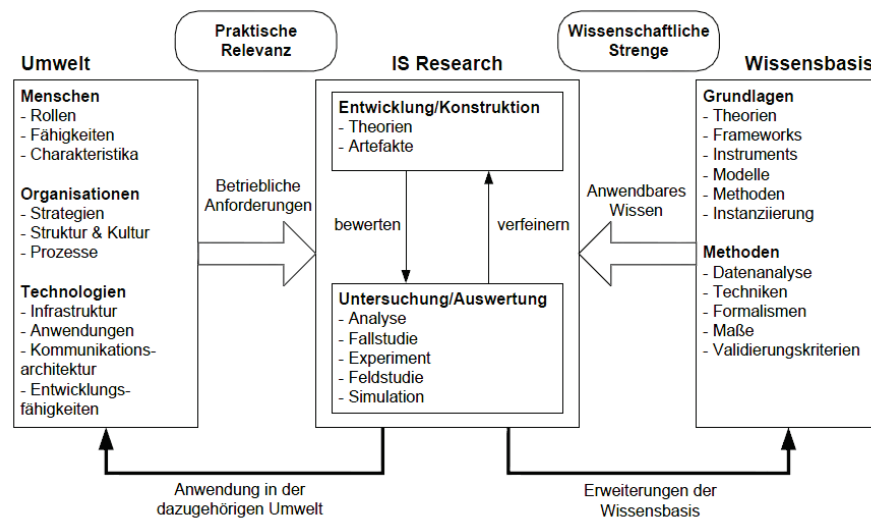


Abbildung 2: Rahmenwerk des Information Systems Research (aus dem Englischen, nach Hevner et al. (2004), S. 80).

Im Rahmen der IS Forschung werden Theorien und Artefakte entwickelt, welche iterativ bewertet und anschließend verfeinert werden können. Bei der Entwicklung liefert die Praxis die betrieblichen Anforderungen an das Artefakt und die Wissensbasis Grundlagen und Methoden. Für die Untersuchung beziehungsweise Auswertung bietet die Umwelt Möglichkeiten für Fallstudien, Feldstudien und Experimente sowie reale Daten für Analyse und Simulation. Zur Durchführung der Untersuchungen stellt die existierende Wissensbasis wiederum Grundlagen und Methoden. Der Rückfluss aus der IS Forschung in die Praxis bedeutet die (erfolgreiche) Anwendung der entwickelten Artefakte und Theorien in der dazugehörigen Umwelt. Der Rückfluss in die Wissensbasis geschieht durch eine Erweiterung dieser, meist in Form neuer Grundlagen und verbesserter oder neuer Methoden.

Für den Prozess der Design Science im Information Systems Research liefert Hevner et al. (2004) sieben Richtlinien (Hevner et al. (2004), S. 83):

- (1) Entwickeln eines Artefaktes (*Design as an Artifact*): Forschung im Design Science muss ein brauchbares Artefakt in Form eines Konstrukts, Modells, einer Methode oder einer Instanziierung erstellen.
- (2) Problemrelevanz (Problem Relevance): Das Ziel der Design Science Forschung ist die Entwicklung Technologie-basierter Lösungen für wichtige und relevante Probleme des Unternehmens.
- (3) Evaluierung des Designs (Design Evaluation): Der Nutzen, die Qualität und die Wirksamkeit (Effizienz und Effektivität) des Artefaktes sollen durch gut durchgeführte Evaluierungsmethoden demonstriert werden.

- (4) Beiträge zur Forschung (Research Contributions): Effektive Design Science Forschung muss klare und nachweisbare Beiträge im Bereich des entwickelten Artefaktes, der Grundlagen der Entwicklung und/oder der Entwicklungsmethoden liefern.
- (5) Wissenschaftliche Strenge (Research Rigor): Design Science Forschung basiert auf der strengen Anwendung von Methoden in der Konstruktion und Evaluierung des Artefaktes.
- (6) Entwicklung als Suchprozess (Design as a Search Process): Die Suche nach einem effektiven Artefakt basiert auf der Verwendung der zur Verfügung stehenden Mittel unter Berücksichtigung der Gesetze ihrer Umwelt (und sollte iterativ geschehen).
- (7) Kommunikation der Forschung (Communication of Research): Design Science Forschung muss effektiv sowohl technologie- als auch management-orientierten Zielgruppen präsentiert werden.

Die von Hevner et al. (2004) erstellten Richtlinien wurden in dieser Arbeit auch als solche genutzt, also nicht als strenge Vorgabe, der auf alle Fälle gefolgt werden muss. Dies zeigte sich auch durch die Anwendung in Praxisprojekten und bei der Verwendung von Methoden anderer Disziplinen, wie zum Beispiel die der klassischen Informatik bei der Entwicklung eines Prototyps. Nachfolgend soll kurz auf die Anwendung der einzelnen Richtlinien in der vorliegenden Arbeit eingegangen werden:

- (1) Die Entwicklung eines beziehungsweise mehrerer Artefakte steht im Kern der Arbeit. Entwickelt wurden ein Servicebeschreibungsmodell, eine Methode zur Durchführung heterogener Service Discovery sowie eine Instanziierung beider in Form eines Prototyps.
- (2) Die Relevanz des Problems, welches durch Modelle und Prototyp gelöst wird, war durch die Anforderung zweier Projekte mit drei unterschiedlichen Industriepartnern gegeben.
- (3) Die Evaluierung erfolgte sowohl quantitativ, unter Verwendung akzeptierter Methoden des Information Retrievals, als auch qualitativ durch die Beschreibung der Erkenntnisse aus den Industrieprojekten in Form von Fallstudien. (Aus Hevner et al. (2004) kamen somit die Evaluierungsmethoden *Fallstudie*, *Funktionales Testen* und *Szenarien* zum Einsatz, siehe Hevner et al. (2004), S. 86)
- (4) Die wissenschaftlichen Fortschritte in den Bereichen der entwickelten Artefakte wurden von bestehenden Ansätzen erfolgreich abgegrenzt und der wissenschaftlichen Fachwelt in Form verschiedener Veröffentlichungen präsentiert.
- (5) Für die Entwicklung wurde auf existierende Methoden zurückgegriffen: Analyse und Synthese bestehender Ansätze, iterative Erstellung des Beschreibungsmodells und des Prototypen. Die Evaluierung erfolgte in einer Fallstudienbetrachtung sowie durch qualitative Methoden des Information Retrieval.
- (6) Sowohl das Servicebeschreibungsmodell als auch der erstellte Prototyp wurden in mehreren Iterationen unter Einbeziehung der Praxispartner entwickelt.

- (7) Die Ergebnisse der Arbeit wurden im Rahmen des Projektes und auch auf wissenschaftlichen Konferenzen sowohl technisch- als auch management-orientierten Zielgruppen präsentiert.

1.3 Struktur der Arbeit

Die Arbeit gliedert sich nach der Einleitung in fünf inhaltliche Kapitel und wird mit einem Kapitel zur Zusammenfassung und einem Ausblick abgeschlossen.

Die Grundlage der Arbeit legt Kapitel 2, in welchem die wichtigsten Begriffe und Theorien für das Verständnis der darauf folgenden Kapitel beschrieben werden. Zunächst werden *serviceorientierte Architekturen* im Allgemeinen beschrieben, sowohl auf konzeptioneller als auch auf technischer Ebene. Dieser Kontext ist wichtig, um Servicebeschreibungen generell und speziell semantische Servicebeschreibungen zu verstehen. Diese werden im daran anschließenden Abschnitt vorgestellt, basierend auf der Annahme des sogenannten *Semantic Webs*. Für den Umgang mit Servicebeschreibungen auf Basis heterogener Ontologien wird in Abschnitt 2.3 das *Ontology Matching und Alignment* erläutert. Abgeschlossen wird das Kapitel mit den Grundlagen der *Geschäftsprozessorientierung und des Geschäftsprozessmanagements*, um den Kontext der anvisierten Service Discovery verstehen zu können.

Aufbauend auf den Grundlagen kann in Kapitel 3 der *Stand der Forschung* der unterschiedlichen Teilbereiche des entwickelten Ansatzes diskutiert werden. Darunter fallen zunächst Ansätze für die allgemeine *Service Discovery*. Der Abschnitt zum *semantischen Geschäftsprozessmanagement* stellt bestehende Ansätze für die Modellierung, Implementierung, Ausführung und Analyse semantischer Geschäftsprozesse vor. Nachdem in Kapitel 2 die Grundlagen des *Ontology Matching und Alignments* besprochen wurden, werden diese in Abschnitt 3.3 vertieft und um wichtige Techniken und Vorgehen auf diesem Gebiet erweitert. Wichtigster Bestandteil des dritten Kapitels ist der letzte Abschnitt, welcher sich den bestehenden Ansätzen für die *heterogene Service Discovery* widmet. Die Erkenntnisse des Abschnitts 3.4 sind kritisch für die Entwicklung und Abgrenzung des in dieser Arbeit entwickelten Ansatzes.

Der Kern der Arbeit ist die Vorstellung des Ansatzes für die *geschäftsprozessorientierte Service Discovery* unter der Annahme *heterogener Servicebeschreibungen* in Kapitel 4. Dieser wird in vier Abschnitten beschrieben. Zunächst wird das Konzept *geschäftsprozessorientierter Servicebeschreibungen* vorgestellt. Anschließend wird erläutert wie die *Service Discovery* über den Kontext der *Geschäftsprozessmodellierung* initiiert werden kann. Abschnitt 4.3 beschreibt die Konzepte zur Realisierung einer *Service Discovery auf Basis heterogener Beschreibungen* beziehungsweise heterogenen Ontologien. Ein zusätzlicher Bestandteil des Konzeptes sind Überlegungen einer verbesserten *Benutzerinteraktion* für die Implementierung, welche das vierte Kapitel abschließen.

Die Beschreibung der Umsetzung des in Kapitel 4 vorgestellten Ansatzes in Form einer *prototypischen Implementierung* erfolgt in Kapitel 5. Nach einer Beschreibung der *Anfor-*

derungen an den Prototypen wird die umgesetzte *Architektur* bestehend aus fünf Komponenten vorgestellt. Für eine Verbesserung des Matchings von Ontologien wird eine zusätzliche Implementierung eines *Modularen Ontology Matchmaking Systems* beschrieben. Dabei werden neben *Architektur* und *Implementierung* des Systems auch die *Interaktion mit dem Benutzer* erläutert.

Der Forschungsmethodik (siehe Abschnitt 1.2) folgend widmet sich das sechste Kapitel der *Evaluierung* des entwickelten Ansatzes und der Implementierung. Die Evaluierung besteht aus einer *quantitativen* und einer *qualitativen Auswertung*. In der quantitativen Auswertung wird die Implementierung der heterogenen Service Discovery anhand anerkannter Metriken des Information Retrievals auf Basis eines Testdatensets bewertet. In der qualitativen Auswertung erfolgt zunächst eine Diskussion des Ansatzes auf Basis zweier Fallstudien. Die Fallstudien basieren auf Praxisprojekten mit internationalen Industriepartnern. Der zweite Teil der qualitativen Analyse besteht aus einer Abgrenzung des entwickelten Ansatzes und der Implementierung zu bestehenden Lösungen aus dem Stand der Forschung.

Kapitel 7 schließt die Arbeit mit einer *Zusammenfassung* der Erkenntnisse und einem *Ausblick* für die Weiterentwicklung des Ansatzes und der Implementierung ab.

1.4 Beiträge für Wissenschaft und Praxis

Im folgenden Abschnitt soll kurz auf die Beiträge der Arbeit für Wissenschaft und Praxis eingegangen werden:

(1) Modell zur geschäftsprozessorientierten Servicebeschreibung

Obwohl bereits Ansätze für die Beschreibung von Services auf Basis der Geschäftsprozessorientierung existieren, zielt die vorliegende Arbeit auf die Erweiterung und Verbesserung dieser ab. Zum einen wird somit die existierende Wissensbasis um ein alternatives Modell zur Servicebeschreibung erweitert, welches Stärken vor allem im Bereich der fachlichen funktionalen Beschreibung besitzt. Zum anderen werden Beispiele für den Einsatz dieser Beschreibungen in der Praxis geliefert, welche von Unternehmen direkt zum Projekteinsatz verwendet werden können. Unterstützt wird der mögliche Einsatz in der Praxis durch die Ableitung der Servicebeschreibungen aus realen Services der durchgeführten Industrieprojekte.

(2) Konzept einer heterogenen Service Discovery

Der Umgang mit Heterogenität in der semantischen Service Discovery ist vor allem in der Wissenschaft ein wenig diskutiertes Problem. Die meisten Lösungen setzen vor allem auf eine manuelle Lösung dieses Problems. Die vorliegende Arbeit liefert ein Konzept, die Service Discovery auf Basis heterogener Beschreibungen zu realisieren und gleichzeitig bestehende Automatismen einzusetzen. Auf diese Weise entsteht ein modularer Ansatz, welcher zum einen von anderen Implementierungen umgesetzt und zum anderen in der Zukunft

sehr leicht um zusätzliche Werkzeuge des Ontology Alignments erweitert werden kann. Eine Demonstration für den Einsatz in der Praxis in Form einer Machbarkeitsstudie wird durch eine prototypische Implementierung realisiert.

(3) **Machbarkeitsstudie**

Die Machbarkeit der entwickelten Konzepte wird im Rahmen einer prototypischen Implementierung gezeigt. Sie demonstriert, dass bestehende Werkzeuge der Geschäftsprozessmodellierung erweitert werden können, um (1) semantische Annotationen in Form von ontologischen Konzepten zu nutzen und (2) als Kontext und Ausgangspunkt für eine (heterogene) semantische Service Discovery zu dienen. Weiterhin wird die Umsetzung des theoretischen Konzeptes einer heterogenen Service Discovery auf Basis bestehender Ontology Alignment Werkzeuge realisiert. Obwohl aufgrund der durchgeführten Auftragsforschung kein Quellcode für die Öffentlichkeit zur Verfügung gestellt werden kann, so ist doch für die beteiligten Unternehmen ein wertvoller Demonstrator entstanden. Dieser kann für die Produktisierung der Konzepte als Grundlage dienen und in der Zukunft um weitere Konzepte erweitert werden.

(4) **Fallstudien**

Ein wichtiger Beitrag zur Evaluierung der Konzepte und der Implementierung ist die Beschreibung der Erfahrungen aus zwei Industrieprojekten. Durch die Mitarbeit von drei internationalen Unternehmen können in der Arbeit zwei Fallstudien vorgestellt werden. Diese demonstrieren einmal mehr die Relevanz der durchgeführten Forschung. Außerdem wird gezeigt, wie durch iteratives Feedback die Validität der Konzepte verbessert werden konnte. Vereinbarungen zur Vertraulichkeit beschränken die Fallstudien in ihrer Detaillierung, trotzdem können generelle Aussagen und Vorgehen bei der Evaluierung wichtige Einblicke in den Einsatz in der Praxis liefern.

(5) **Testset und quantitative Auswertung**

Für die Evaluierung zeigte es sich als notwendig, ein Testset an semantisch annotierten Services und Anfragen zu erstellen. Auf Grundlage dieser kann eine Auswertung der Implementierung anhand mehrerer Metriken des Information Retrievals durchgeführt werden. Somit entstehen forschungsrelevante Erkenntnisse zur Durchführung und Analyse der Evaluierung einer heterogenen Service Discovery.

2 Definition grundlegender Begriffe

Das vorliegende Kapitel definiert eine Vielzahl grundlegender Begrifflichkeiten aus den Bereichen Serviceorientierte Architekturen, Semantic Web und Web Services, Ontology Matching und Alignment sowie Geschäftsprozessorientierung und -management. Das Verständnis dieser Definitionen ist die Basis für den folgenden Teil der Dissertation.

2.1 Serviceorientierte Architekturen

In den folgenden Abschnitten soll zunächst geklärt werden, welchen Ursprung das Konzept der Serviceorientierung hat und welche Bedeutung es heute besitzt. Weiterhin wird versucht, den oft unterschiedlich interpretierten Begriff der serviceorientierten Architektur für den Rahmen der geplanten Dissertation abzugrenzen. Abschließend erläutert ein kurzer Exkurs die technischen Grundlagen für die Serviceorientierung und die SOA.

2.1.1 Serviceorientierung

Die Wurzeln der Serviceorientierung liegen in unterschiedlichen Bereichen der Informationstechnologie. Dabei haben Programmiersprachen, Infrastrukturtechnologien für Fernzugriffe und Unternehmens-EDV gleichermaßen an der Entwicklung dieses Paradigmas mitgewirkt. Im Bereich der Programmiersprachen wurde das Servicekonzept vor allem durch Komponenten und Objekte sowie ihre Abstraktion, Kapselung und Granularität beeinflusst. Verteilte Anwendungen und Infrastrukturentwicklungen prägten maßgeblich die Nutzung von standardisierten Schnittstellen zur Veröffentlichung von Funktionalitäten. Die Unternehmensarchitekturen der EDV sind der dritte Treiber für die Serviceorientierung. Hier ist vor allem die Nachfrage nach Enterprise Application Integration zu nennen, welche auch das Servicekonzept beeinflusst hat. (Krafzig et al. (2005), S. 13ff)

Serviceorientierung definiert sich, wie der Name schon sagt, durch die Nutzung von Dienstleistungen beziehungsweise Funktionen. Services stellen Komponenten dar, welche durch die Aufteilung von bestimmten Aufgabenfeldern eigenständig nutzbare und abgegrenzte Funktionen erfüllen. Ein Service bietet unterschiedliche Operationen über eine definierte Schnittstelle an. Dabei ist allerdings nicht relevant, wie diese realisiert werden.¹ Services können unterschiedliche Ausprägungen ihres Dienstumfanges haben. (Oey et al. (2005), S. 205-206)

¹ Nutzung des sog. Black-Box-Prinzps.

Obwohl der Begriff Komponente gewählt wurde, sollte die Serviceorientierung nicht mit der Komponentenorientierung verwechselt werden. Zwischen den beiden Konzepten gibt es wichtige Unterschiede. Vor allem die Dynamik der Services bei Verwendung und Komposition sowie deren Verfügbarkeit in einer sich verändernden Umgebung sind hier zu nennen. (Cervantes und Hall (2004))

2.1.2 Unternehmensarchitekturen

Bevor der Begriff der Serviceorientierten Architektur geklärt werden kann, soll kurz auf die Unternehmensarchitektur eingegangen werden. Ursprünglich ist der Architekturbegriff aus der Baukunst und Stadtplanung abgeleitet (Aier (2007), S. 103ff). Im Folgenden soll das Verständnis aus Sicht der Wirtschaftsinformatik verwendet werden. Nach Aier (2007) lässt sich eine Architektur im Sinne der Wirtschaftsinformatik, im Folgenden als Unternehmensarchitektur bezeichnet, wie folgt definieren:

„Eine Unternehmensarchitektur ist die abstrakte, planende Beschreibung der Strukturen und Muster der Organisation und IT eines Unternehmens und deren Zusammenspiel. Das Ziel dieser Beschreibung ist es, einen nachhaltigen Wandlungsprozess eines Unternehmens zu unterstützen.“ (Aier (2007), S. 109)

Das Zusammenspiel von Organisationsarchitektur auf der einen und IT-Architektur auf der andere Seite ist in Abbildung 3 dargestellt.

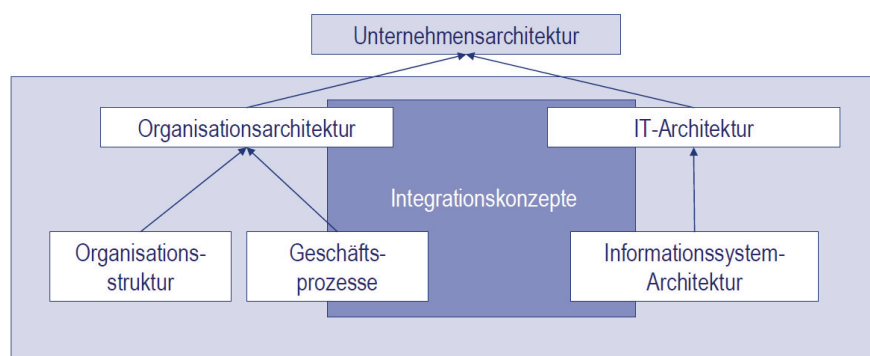


Abbildung 3: Bestandteile der Unternehmensarchitektur nach Aier (2007), S. 135.

Dieses Verständnis ist wichtig, da sich im Weiteren zeigen wird, dass eine Serviceorientierte Architektur bereits in ihrer Definition sowohl Komponenten der Organisation als auch der Informationstechnologie beinhaltet, obwohl manchmal nur von einer IT- oder IS-Architektur gesprochen wird. Vor allem die Verknüpfung von Geschäftsprozessen mit Software-Services spielt dabei eine entscheidende Rolle.

2.1.3 Begriffsdefinition SOA

Der „Hype“ um das Konzept von SOAs hat unter anderem dazu geführt, dass es die unterschiedlichsten Definitionen für sie gibt. Dabei sind die Unterschiede beim Verständnis durchaus stark. Oey et al. (2005) bemängeln vor allem die divergierende Gewichtung einzelner Aspekte und den Detaillierungsgrad der Definitionen. Gefordert wird eine Abgrenzung gegenüber anderen Architekturansätzen. Um das Verständnis des Begriffs zu verbessern, schlagen Oey et al. (2005) eine einheitliche Definition vor:

„Eine serviceorientierte Architektur ist ein Softwarearchitekturkonzept, in welchem Funktionen in Form von wiederverwendbaren, voneinander unabhängigen und lose gekoppelten Services implementiert werden. Services können unabhängig von zugrunde liegenden Implementierungen über wohldefinierte und veröffentlichte Serviceschnittstellen aufgerufen werden. Serviceinteraktion findet über eine dafür vorgesehene Kommunikationsinfrastruktur statt. Mit einer serviceorientierten Architektur werden insbesondere die Gestaltungsziele der Geschäftsprozessorientierung, der Flexibilität und der Wiederverwendbarkeit verbunden.“ (Oey et al. (2005), S. 203)

Auffällig ist an dieser Definition vor allem die Verknüpfung mit Zielen, welche diese zunächst allgemeine Definition stark an die Geschäftsprozessorientierung bindet. Ob dies die Verwendbarkeit der Definition erhöht ist aber fraglich.

Eine weitere, sehr umfassende Definition präsentiert Heutschi (2007):

„Eine SOA ist eine mehrschichtige, verteilte Informationssystem (IS)-Architektur, die Teile von Applikationen für eine vereinfachte Prozessintegration als geschäftsorientierte Services kapselt und dabei bestimmte Designprinzipien berücksichtigt.“ (Heutschi (2007), S. 22)

Hierbei ist zu beachten, dass auch die Geschäftsorientierung und eine Prozessintegration hervorgehoben werden. Fragwürdig ist, ob Services auch unabhängig von Applikationen oder nur als deren Kapselung existieren können. Die weitere Definition von Services im Sinne einer SOA entschärft diese Frage ein wenig:

„Ein Service stellt ein abstraktes Software-Element bzw. eine Schnittstelle dar, die anderen Applikationen über ein Netzwerk einen standardisierten Zugriff auf Anwendungsfunktionen anbietet.“ (Heutschi (2007), S. 22)

Wichtig ist die Feststellung, dass per Definition keine Technologie vorgeschrieben wird, um eine Serviceorientierte Architektur zu implementieren. Anwendungsfunktionen können auch unabhängig von bestehenden Applikationen und Applikationslandschaften durch Services angeboten und von anderen Services oder Applikationen verwendet werden. Gerade diese Vielfältigkeit an potentiellen Services in einem verteilten System ist einer der Gründe, warum die Suche nach Services mit passender Funktionalität eine große Herausforderung darstellt.

Dass sich unterschiedliche Sichten auf eine SOA durchaus vereinbaren lassen, ohne alle Aspekte in eine unflexible Definition zu zwingen, zeigt die Arbeit von Offermann et al. (2007). Auch hier erkennen die Autoren, dass es Verwirrung um die Begrifflichkeit ei-

ner SOA gibt. Es gibt verschiedene Blickwinkel auf eine SOA, wodurch bestimmte Zielarchitekturen entstehen. Problematisch wird es, wenn diese Zielarchitekturen unklar sind, wodurch Erstellung und Management der SOAs missglücken können. (Offermann et al. (2007))

Als Grundvoraussetzung ist anzunehmen, dass Services in einer SOA genutzt werden, um bestimmte Anwendungen zu kreieren. Die Unterschiede finden sich vor allem in der Art und Weise, wie die Services genutzt werden sollen. Aus diesen Überlegungen und eingehender Literaturrecherche definieren Offermann et al. (2007) sieben Haupt-sichten auf eine SOA:

- „Internet Application Development
- Application Development
- Enterprise Application Integration
- Workflow Support
- Business Process Management
- Collaborative Business Process Integration
- Grid Computing“.

Aus diesen unterschiedlichen Sichten auf eine SOA wurde von Offermann et al. (2007) ein Serviceschichten- und Serviceinteraktionsmodell entwickelt, die „holistic Service-Oriented Architecture“ (siehe Abbildung 4). Für die Gesamtheit der vorliegenden Arbeit wird dieses ganzheitliche Verständnis einer SOA verwendet. Die Einbeziehung der verschiedenen Sichten und Abstraktionsschichten erlaubt ein flexibleres Verständnis des SOA-Begriffs, als die Verwendung einer einzigen statischen, wenn auch vielleicht umfassenden Definition.

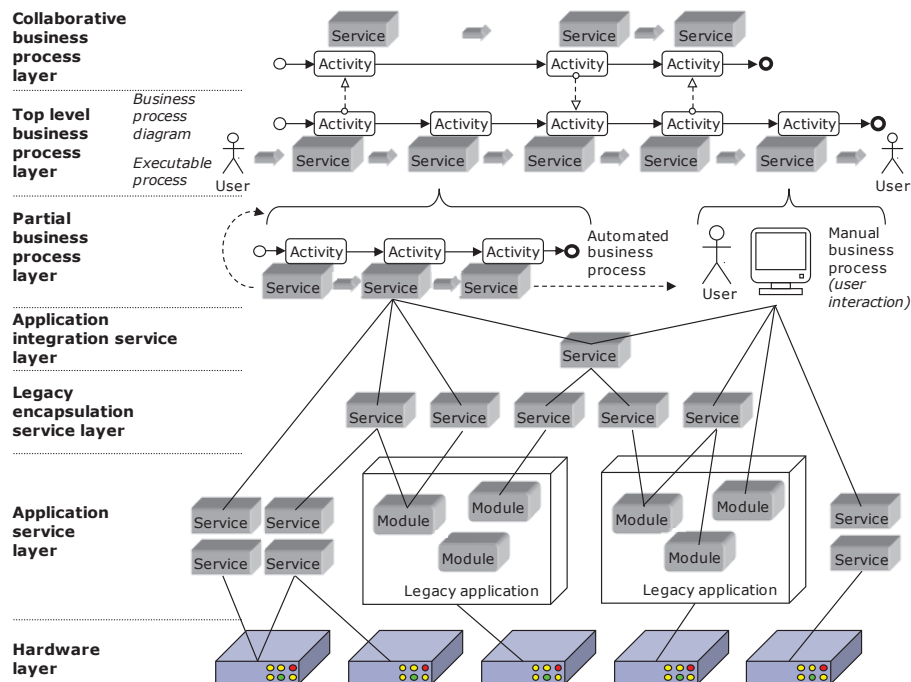


Abbildung 4: SL&I-Modell einer ganzheitlichen SOA nach Offermann et al. (2007)

2.1.4 Prinzipien der Serviceorientierung

Dem Paradigma der Serviceorientierung liegen, ähnlich wie der Objekt- oder Komponentenorientierung, bestimmte Konzepte und Prinzipien zugrunde (Erl (2005) S. 278). Dabei scheint es noch keine Einigkeit über die Grundprinzipien zu geben, welche auf jeden Fall im Rahmen des Servicedesigns Beachtung finden sollten (Legner und Heutschi (2007)). Zum besseren Verständnis der zugrunde liegenden Konzepte der Serviceorientierung geben die folgenden Abschnitte eine Übersicht über die Grund- und Zielprinzipien der Services in einer SOA.

2.1.4.1 Grundprinzipien

Im Folgenden werden neun Grundprinzipien der Serviceorientierung und ihre Beziehungen vorgestellt. Diese wurden durch eingehende Literaturrecherche identifiziert und anschließend miteinander in Verbindung gesetzt. Dabei wird davon ausgegangen, dass es ohne die Beachtung dieser Grundprinzipien nicht möglich sein wird, die Zielprinzipien - Komponierbarkeit, Interoperabilität, Wiederverwendbarkeit und Integration - vollständig zu erreichen. Die Beziehungen zu den Zielprinzipien werden in Abschnitt 2.1.4.2 vorgestellt.

Lose Kopplung (Loose Coupling)

Diese Eigenschaft wird von manchen Autoren als eines der Schlüsselprinzipien des Servicedesigns bezeichnet (Papazoglou et al. (2006), S. 22). Dabei beschreibt die Kopplung die Abhängigkeiten zwischen den Services, welche minimiert werden sollten (siehe zum Beispiel Legner und Heutschi (2007), S. 4 und Vinoski (2005), S. 72). Es existieren durchaus unterschiedliche Klassifizierungen der Kopplung. Sie kann abhängig von Ort, Plattform, Sprache, Vertrag, Format, Protokoll und Zeit sein (Bieberstein et al. (2006), S. 91). Diese detaillierte Auflistung beinhaltet viele Elemente der repräsentationalen Kopplung, welche davon ausgeht, dass Services unabhängig von Repräsentation, Implementierung oder Annahmen anderer Services sein sollten (Papazoglou und van den Heuvel (2006), S. 4). Im Rahmen der funktionalen Kopplung muss darauf geachtet werden, dass die Services eigenständig mit der Semantik der eingehenden Nachrichten umgehen und Nachrichtenformate notfalls umwandeln können (Lublinsky (2007)). Die Transaktionalität der Services wird durch eine lose Kopplung allerdings problematisch. Das Standardkonzept ACID², welches oft in verteilten Systemen verwendet wird, um die Verlässlichkeit dieser zu erhöhen, würde in einer SOA zu Problemen führen (Lublinsky (2007)).

Da die gesamte Interaktion über standardisierte Schnittstellen läuft, wird eine lose Kopplung der Services ermöglicht. Die Eliminierung von Abhängigkeiten unterstützt die Autonomie der Services. Lose gekoppelte Kommunikationsstrukturen begünstigen zudem die Zustandslosigkeit. (Erl (2005), S. 315)

Granularität (Granularity)

Die Art der Granularität der Services, ob fein oder grobgranular, wird oft diskutiert. Dabei ist zu beachten, welche Funktionalitäten durch die Services bereitgestellt werden und wie sie die Geschäftsaktivitäten unterstützen können (Legner und Heutschi (2007), S. 5). Zum einen wird vertreten, dass Schnittstellen eher grobgranular entworfen werden sollten, um einen Geschäftsprozess vollständig unterstützen zu können (Papazoglou und van den Heuvel (2006), S. 5 sowie Voß et al. (2006), S. 58). Andererseits können zu grobe Services die Wiederverwendbarkeit dieser einschränken. Daher sollte eine Balance zwischen einfachen Funktionalitäten eines Bottom-up-Ansatzes mit den eher grobgranularen Services aus einem Top-down-Ansatz gefunden werden (Bieberstein et al. (2006), S. 94)

Die Granularität beeinflusst die Kopplung, wobei sich Services mit einer bestimmten Anzahl an Operationen leichter von anderen entkoppeln lassen (Voß et al. (2006), S. 58).

² Atomicity, Consistency, Isolation und Durability

Auffindbarkeit (Discoverability)

Die dynamische Zusammenstellung und Nutzung von Services ist nur möglich, wenn diese auch auffindbar sind. Hiermit ist zunächst nicht die technische Realisierung, zum Beispiel mittels sogenannter „Service-Repositories“, gemeint. Beim Design sollte darauf geachtet werden, dass ausreichende Meta-Informationen für den Service formuliert werden, welche sowohl seine Aufgabe als auch die Funktionalität seiner Operationen beschreibt. (Erl (2005) S. 303-305)

Die Serviceverträge bilden die Basis für die Auffindbarkeit der Services. Dabei spielt eine Beschreibung der Schnittstellen eine wichtige Rolle für die Auswertung, ob ein Service von einer anfragenden Entität verwendet werden kann. (Erl (2005), S. 320)

Kapselung (Encapsulation)

Die Kapselung von Funktionen innerhalb eines Services stammt aus älteren Disziplinen der Softwareentwicklung, wie der Objekt- oder Komponentenorientierung (Krafzig et al. (2005), S. 16-18). Auch für die Serviceorientierung bietet dieses Prinzip des „Versteckens“ von Informationen diverse Vorteile. Es erlaubt einen Umgang mit der Komplexität der Funktionen einer SOA (Bieberstein et al. (2006), S. 90). Dabei werden die Details der Implementierung, zum Beispiel die Nutzung von Altsystemen, in einer „Black Box“ verborgen (Erl (2005), S. 289-299).

Die Kapselung geschieht mittels der Schnittstellen und hängt gleichzeitig stark mit dem Prinzip der Abstraktion zusammen (Erl (2005), S. 298-299 und Bieberstein et al. (2006), S. 90).

Abstraktion (Abstraction)

Um mit Komplexität umzugehen wird in vielen Disziplinen die Abstraktion eingesetzt. Auch in der Serviceorientierung findet sie Anwendung. So kann beispielsweise durch eine Kapselung von der zugrunde liegenden Altsystemimplementierung abstrahiert werden (Legner und Heutschi (2007), S. 4). Auch die Struktur einer SOA weist unterschiedliche Grade der Abstraktion auf. Es können unterschiedliche Schichten auf Unternehmens-, Prozess-, Service-, Komponenten- und Objekt-Ebene entstehen (Bieberstein et al. (2006), S. 86-89). Auch die hSOA präsentiert sich in verschiedenen Schichten der Abstraktion (siehe Abbildung 4). Zusätzlich wird bei den Schnittstellen die zugrunde liegende Logik durch verschiedene Operatoren abstrahiert (Erl (2005), S. 299).

Zustandslosigkeit (Statelessness)

Services benötigen einen gewissen Grad an Zustandslosigkeit. Die Operationen müssen so entworfen werden, dass sie Funktionalitäten bereitstellen, ohne einen bestimmten Zustand des Services vorauszusetzen. Eine Möglichkeit zur Umsetzung wäre die Nutzung dokumentenbasierter Nachrichten. (Erl (2005), S. 299)

Die zustandslose Interaktion der Services trägt unter anderem zur Entkopplung bei (Legner und Heutschi (2007), S. 4). Andererseits unterstützt eine bestehende Kommunikation auf Basis loser Kopplung auch den Entwurf von zustandslosen Services, da Zustandsinformationen innerhalb der Nachrichten gespeichert werden können. Ermöglicht wird die Vermeidung von Zuständen zudem durch eine Autonomie der Services. Dabei kann die zugrunde liegende Logik so angepasst werden, dass die Verwaltung der Zustände außerhalb der Grenzen des Service geschieht (Erl (2005), S. 319-320).

Standardisierte Schnittstellen (Standardized Interfaces)

Das Prinzip standardisierter Schnittstellen ist einer der Kernpunkte einer SOA und stammt unter anderem aus der Entwicklung von verteilten Anwendungen (Krafzig et al. (2005), S. 20). Dies geschieht meist in Form von sogenannten Serviceverträgen. Diese enthalten eine Beschreibung des Service, seiner Operationen und der Nachrichten, welche empfangen und gesendet werden können (Erl (2005), S. 295-296). Die Definition einer formalen, standardisierten Schnittstelle ist nicht nur für die Wiederverwendbarkeit wichtig, sondern auch für die Kontinuität des Unternehmens, welches die SOA einführt.

Ein Servicevertrag bietet die Basis der Auffindbarkeit und dient als Mittel der Abstraktion. Weiterhin wird eine lose Kopplung der Services ermöglicht, indem die Schnittstellen alle relevanten Informationen zur Interaktion bereitstellen. (Erl (2005), S. 313-314)

Kohäsion (Cohesion)

Das Prinzip der Kohäsion ist in der Serviceorientierung auf die Operationen der Services anzuwenden (Bieberstein et al. (2006), S. 92). Die Kohäsion ist als Grad der Stärke der funktionalen Abhängigkeit von Operationen zu verstehen. Durch hohe funktionale Kohäsion wird sichergestellt, dass ein Service eine bestimmte, problemorientierte Aufgabe ausführen kann und nur die dafür nötigen Operationen besitzt. Die kommunikationale Kohäsion erlaubt die Ausführung mehrerer Operationen innerhalb des Service durch die Nutzung der gleichen Eingangs- und Ausgangsnachrichten (Vinoski (2005), S. 73 und Papazoglou und van den Heuvel (2006), S. 4-5). In diesem Zusammenhang ist auch eine starke Kohäsion der Informationen der Serviceschnittstelle von Vorteil. Operationen sollten optimalerweise Typen, Struktur, Beziehungen und Logik teilen (Bieberstein et al. (2006), S. 93-94). Allerdings ist darauf zu achten, dass keine „schlechten“ Arten der Kohäsion verwendet werden. Beispielsweise sollten Operationen nicht zu einem Service zusammengefasst werden, nur weil sich die Art ihrer Implementierung ähnelt (Vinoski (2005), S. 74).

Kohäsion ist damit direkt an die Schnittstelle gebunden und verbessert bei richtiger Anwendung indirekt die lose Kopplung der Services. Nochmals ist darauf hinzuweisen, dass nicht die Kohäsion der Services einer SOA gemeint ist. Eine solche würde nämlich genau das Gegenteil einer losen Kopplung verursachen.

Autonomie (Autonomy)

Im Rahmen einer SOA wird die existierende Anwendungsarchitektur und -struktur in eine Vielzahl von kontrollierbaren autonomen Services aufgeteilt (Legner und Heutschi (2007), S. 4). Die Autonomie der Services erfordert, dass die verwendete Logik innerhalb bestimmter Grenzen existiert. So kann eine Kontrolle über die unterliegenden Ressourcen des Service erreicht werden. Das muss nicht heißen, dass diese Kontrolle exklusiv ist. Sie kann auch auf Basis von Service-Levels mit anderen geteilt werden. Während der Ausführungszeit muss die Autonomie allerdings auf jeden Fall gegeben sein (Erl (2005), S. 303-305).

Das Prinzip der Autonomie wird mittels der Entkopplung der Services durch das Entfernen unnötiger Abhängigkeiten ermöglicht. Je höher die Autonomie eines Service ist, desto einfacher kann dessen Zustandslosigkeit erreicht werden. (Erl (2005), S. 303-305)

2.1.4.2 Zielprinzipien

In der vorliegenden Arbeit sind die Zielprinzipien zur Entwicklung einer SOA Wiederverwendbarkeit, Komponierbarkeit, Interoperabilität und Integration. Im weiteren Verlauf werden sie beschrieben und ihre Beziehungen untereinander sowie die Unterstützung durch die Grundprinzipien dargestellt.

Die Zielprinzipien erhöhen direkt oder indirekt die Agilität, Flexibilität und Wartbarkeit (Krafzig et al. (2005), S. 6-7). Dadurch wird es der IT-Architektur möglich, sich den immer schneller ändernden Begebenheiten anzupassen und einem erhöhten Kostendruck standzuhalten.

Wiederverwendbarkeit (Reusability)

Um den erforderlichen Entwicklungs- und Wartungsaufwand der Architektur zu verringern und eine konsistente Umsetzung von Services zu gewährleisten, ist die Wiederverwendbarkeit eines der Gestaltungsziele einer SOA (Oey et al. (2005), S. 213). Sie erhöht, ob sofort oder in der Zukunft, die Flexibilität bei der Einführung oder Änderung von Prozessen. Durch maximale Wiederverwendbarkeit wird zudem die Redundanz der Services minimiert (Bieberstein et al. (2006), S. 89). Die Nutzung von Designstandards, welche jeden Service potentiell wiederverwendbar machen, ermöglicht es, zukünftigen Anforderungen mit geringerem Aufwand zu begegnen (Erl (2005), S. 292-293).

Über die Abstraktion wird die Wiederverwendbarkeit durch Bereitstellung generischer Schnittstellen unterstützt. Dabei ist die richtige Granularität der Schnittstelle entscheidend dafür, ob und wie ein Service mehrfach genutzt werden kann. Zustandslosigkeit, Autonomie und lose Kopplung ermöglichen den Entwurf unabhängiger Services, welche somit einfacher wiederzuverwenden sind. Services müssen zudem auffindbar sein, um sie einfach und dynamisch wiederverwenden zu können. (Erl (2005), S. 312-313)

Komponierbarkeit (Composability)

Bei der Entwicklung von Services ist darauf zu achten, dass sie sich mit anderen Services zu einer Komposition verbinden lassen. Im Rahmen einer SOA wird oft der Begriff Orchestrierung verwendet, welche sich auf die Komposition von Services zur Unterstützung eines Prozesses oder Prozessschrittes innerhalb des Unternehmens bezieht (Erl (2005), S. 301-302). Bei der Orchestrierung ist die Logik der Prozessausführung zentralisiert. Wenn diese Logik allerdings innerhalb der Nachrichten der Services enthalten ist, spricht man von einer Choreographie (Voß et al. (2006), S. 56). Durch eine implizite Koordinierung mittels einer Choreographie wird vor allem die Kollaboration von Services zwischen Unternehmen möglich (Offermann et al. (2007), S. 4).

Die Komponierbarkeit basiert hauptsächlich auf der Wiederverwendbarkeit. Zusätzlich sind standardisierte Schnittstellenoperatoren und passende Granularitäten nötig. Durch Zustandslosigkeit und Autonomie erhöht sich die Qualität der Komposition, da Abhängigkeitsprobleme vermieden werden. Lose Kopplung ermöglicht eine dynamische Kombination, weil keine Abhängigkeiten der Services beachtet werden müssen. (Erl (2005), S. 302 und 317)

Integration (Integration)

Die Integration ist kein neues, aber ein wichtiges Zielprinzip. Es stammt aus der Problematik, dass unterschiedliche Anwendungen innerhalb eines Unternehmens existieren, welche allerdings nicht oder nicht ausreichend miteinander verbunden sind. Um eine Integration der Anwendungen zu erreichen, kommt vor allem das Grundprinzip der Kapselung zum Tragen (Offermann et al. (2007), S. 3). Bei der Integration von Anwendungen durch Kapselung ist weiterhin auf alle anderen Grundprinzipien zu achten. Dabei fällt auf, dass sich Autonomie und Zustandslosigkeit, bedingt durch die Implementierungen der Altsysteme, nicht immer vollständig garantieren lassen. Die Nutzung von Service-Levels kann aber zumindest bei der Garantierung der Autonomie helfen (Erl (2005), S. 304-305).

Interoperabilität (Interoperability)

Die Interoperabilität ist ein wichtiges Prinzip zur Integration von Services und Anwendungen. Services können so entworfen werden, dass nicht nur momentane Lösungen interoperabel werden, sondern auch zukünftige Anwendungsintegration erleichtert und beschleunigt wird (Erl (2005), S. 45). Die Interoperabilität der Services wird oft durch offene und weit verbreitete technische und Geschäftsstandards garantiert (Legner und Heutschi (2007), S. 4). Doch auch die bereits beschriebenen Prinzipien besitzen einen gewissen Einfluss. Allen voran formen standardisierte Schnittstellen die Basis reibungsloser Interoperabilität. Eine lose Kopplung wird benötigt, um Services unabhängig voneinander zu verwenden und so in jeder möglichen Kombination interoperabel zu machen. Zudem erlaubt die Integration von bestehenden Anwendungen ihre unternehmensweite Kombination mittels Services.

Ähnlich der oben diskutierten Prinzipien und somit verknüpften Ziele schlägt Heutschi (2007) die Konsolidierung auf folgende Prinzipien vor (Heutschi (2007), S. 30-47):

- Schnittstellenorientierung
- Interoperabilität
- Autonomie und Modularität
- Bedarfsorientierung

Die Schnittstellenorientierung beinhaltet dabei eine Abstraktion der Services von der Serviceimplementierung, stabile und gemanagte Servicekontrakte (Servicevereinbarungen) sowie umfassende und einheitliche Servicespezifikationen. Die Interoperabilität soll gewährleistet werden durch technische und fachliche Standardisierung der Service-schnittstellen sowie der Verwendung offener und verbreiteter Industriestandards. Um Autonomie und Modularität gewährleisten zu können soll auf hohe Servicekohäsion und schwache logische Kopplung sowie auf eine lose gekoppelte Kommunikation zwischen den Services geachtet werden. Schließlich fordert Heutschi (2007) eine grobe Servicegranularität, welche an den Geschäftskonzepten ausgerichtet ist, sowie eine Generalisierung der Serviceleistung.³

2.1.5 Technische Grundlagen einer SOA

Im folgenden Unterabschnitt wird auf das Konzept der Web Services, deren Beschreibungssprache und die technischen Komponenten zur Realisierung einer serviceorientierten Architektur eingegangen.

2.1.5.1 Web Service und Web Service Description Language

Nachdem abstrakte Konzepte zu Service, Serviceorientierung und SOA diskutiert wurden, soll kurz auf eine Grundlage zur technischen Realisierung dieser Konzepte eingegangen werden: Web Services. Das *World Wide Web Consortium* (W3C) definiert einen Web Service wie folgt:

“A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.” (World Wide Web Consortium (2004))

³ Eine Übersicht über den Vergleich verschiedener Quellen zu Designprinzipien in einer SOA findet sich in Heutschi (2007), S. 47ff.

Die Beschreibung der Schnittstellen in Form von WSDL ermöglicht einen standardisierten Zugriff auf Services. Obwohl Services auch mittels anderer Technologien bereitgestellt werden können (Heutschi und Schemm (2005)), kristallisiert sich seit einigen Jahren die Nutzung von Web Services als de-facto Standard für die Realisierung von Services in einer SOA heraus. Daher liegt im Folgenden dem abstrakten Begriff Service die technische Realisierung in Form eines Web Service zu Grunde.

2.1.5.2 Technische Komponenten einer SOA

Die technische Infrastruktur zu Ausführung einer SOA wird nicht im Fokus dieser Arbeit stehen. Daher soll nachfolgend nur kurz auf mögliche Komponenten einer SOA eingegangen werden, um ein tieferes Verständnis des Gesamtkonzeptes zu erreichen.

Die Basiskomponenten einer SOA sind in Abbildung 5 skizziert. Die Services (in der Abbildung als Dienste bezeichnet) werden über den Service Bus von Dienst-Nutzern konsumiert. Dabei ist es unerheblich, ob die Dienste ihre Funktionalität autark implementieren, also als Service im Sinne einer SOA konzipiert wurden, oder ob sie Funktionalität von Altkomponenten kapseln. Die Serviceschnittstellen können mit der im vorangegangenen Abschnitt beschriebenen WSDL definiert sein. Es gibt allerdings auch SOAs welche auf andere Technologien setzen. Die Dienst-Nutzer oder auch Konsumenten des Services können unterschiedliche Funktionen haben. Möglich ist der Konsum eines Service um einen höheren Service durch eine Komposition bestehender Services zu kreieren. Die Präsentationsschicht einer Anwendung kann Services konsumieren und direkt dem menschlichen Nutzer anbieten, entweder zur Interaktion oder reinen Repräsentation. Im Bereich der Geschäftsprozessautomatisierung werden Services vor allem durch sogenannte Business Process Management Systems oder auch Business Process Execution Engines konsumiert, um Prozesse ganz oder teilweise zu automatisieren. (Schröpfer (2009), S. 26)

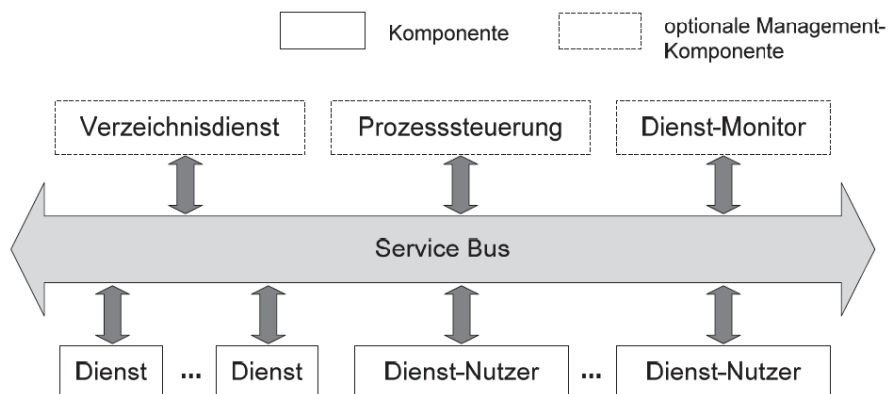


Abbildung 5: Mögliche Komponenten einer serviceorientierten Architektur, nach Schröpfer (2009), S. 27.

Der Service Bus, oft auch als Enterprise Service Bus (ESB) bezeichnet, bietet Zugriff auf die Serviceschnittstellen und vermittelt die Nachrichten zwischen Konsument und Dienst. Diese Form einer *Middleware* hat in vielen technischen Lösungen ihren Ursprung in der *Enterprise Application Integration* (EAI), welche vor dem Entstehen des SOA Paradigmas für die Integration zwischen verschiedenen Systemen eines Unternehmens genutzt wurde. Neben den grundlegenden Komponenten Dienst, Dienst-Nutzer und Service Bus existieren in vielen Lösungen optionale Komponenten. Beispiele sind, wie in Abbildung 5 dargestellt, Verzeichnisdienst, Prozesssteuerung und Dienst-Monitor. Nicht skizzierte Erweiterungen der technischen Architektur können Sicherheitsdienste, Rechteverwaltung und Regeldienste (*Rule Engines*) sein. (Schröpfer (2009), S. 27)

Hervorgehoben sei für die vorliegende Arbeit die Komponente des Verzeichnisdienstes, welche oft bereits einfache Formen der Servicesuche durch Schlüsselwörter oder Kategorisierungen beinhaltet. Viele Implementierungen basieren dabei auf dem Universal Description, Discovery and Integration (UDDI⁴) Standard der OASIS (Clement et al. (2004)).

2.2 Semantic Web und Semantic Web Services

Die formalen Sprachen und Definitionen zur Beschreibung von Web Services, zum Beispiel im Sinne des W3C, erlauben zwar Aussagen über Struktur und Regeln zum Ausdruck der Services, also deren Syntax. Es ist aber nicht möglich eine eindeutige Bedeutung und den Kontext der Services zu beschreiben. Diese Bedeutung und Einordnung in einen Kontext wird gemeinhin als Semantik bezeichnet.

Zu einem besseren Verständnis wird im Folgenden das semiotische Dreieck oder auch „Dreieck der Bedeutung“ (*engl.* Triangle of Reference) nach Ogden und Richards (1960)⁵ verwendet, zu sehen in Abbildung 6.

⁴ <http://uddi.xml.org/>

⁵ Erstausgabe 1923

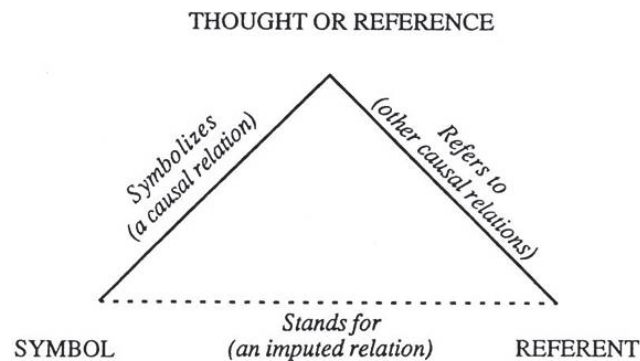


Abbildung 6: Semiotisches Dreieck nach Ogden und Richards (1960).

Der Ansatz des semiotischen Dreiecks kommt ursprünglich aus den Sprachwissenschaften und beschreibt die Beziehungen zwischen *Symbol*, *Referenz/ Konzept* und *Objekt/ Ding*. Ein Konzept oder ein Gedanke bezieht sich zum einen auf ein Objekt in der realen (oder auch in der fiktiven) Welt. Zum anderen kann ein solches Konzept durch ein Symbol beschrieben werden (zum Beispiel ein Wort in der natürlichen Sprache). Durch diese Symbolisierung entsteht implizit eine Verbindung zwischen Symbol und Objekt, dem Symbol wird also eine Verbindung zum Objekt „unterstellt“. Eine ähnliche Dreiteilung formulierten Autoren bereits vor OGDEN und RICHARDS. So beschrieb Frege (1892) die Beziehung zwischen *Sinn* und *Bedeutung* (von FREGE als *Bezeichnung* ausgedrückt).

Für die Beschreibung von Services im Allgemeinen und Web Services im Speziellen bedeutet dies, dass neben einer syntaktischen Sprache zusätzlich eine semantische Sprache benötigt wird, um die Bedeutung (also auch die Funktionalität) ausdrücken zu können. Deutlich wird diese Notwendigkeit zum einen durch Vieldeutigkeit der natürlichen Sprache, welche je nach Kontext oder auch Region unterschiedliche Bedeutungen annehmen kann. Zum anderen erlaubt die semantische Beschreibung die maschinelle Verarbeitung von Bedeutung im Sinne des Semantic Web nach Berners-Lee et al. (2001) (Siehe auch Abschnitt 2.2.2).

2.2.1 Ontologien

Als Grundlage der Repräsentation von semantischem Wissen, also der Bedeutung bestimmter Konzepte, werden größtenteils Ontologien verwendet. Der Begriff *Ontologie* stammt ursprünglich aus der Philosophie und beschreibt die „Theorie der existierenden Dinge“ (Ehrig (2007), S.11). Er ist Teil der Metaphysik (Grimm et al. (2007), S. 68.). In Informationssystemen werden Ontologien und Ontologiesprachen zur Repräsentation, Verteilung und Verarbeitung von Wissen genutzt. Dafür ist es nötig die Informationen für die maschinelle Verarbeitung bereitzustellen. (Grimm et al. (2007), S. 68)

2.2.1.1 Repräsentation von Wissen

Ein Forschungsbereich der künstlichen Intelligenz (engl.: Artificial Intelligence) ist die Darstellung von Wissen für die Verarbeitung durch Maschinen. Symbole werden in diesem Umfeld genutzt um Artefakte der realen Welt, meist einer bestimmten Domäne, darzustellen. Solche Artefakte können neben physischen Objekten auch Ereignisse oder Beziehungen sein. Auf diese Weise wird eine Wissensbasis aufgebaut, welche diese Symbole sowie bestimmte Abhängigkeiten und Regeln beinhaltet. (Grimm et al. (2007), S. 51)

2.2.1.2 Ontologien in der Informationsverarbeitung

Der Begriff der Ontologie in der Informatik beziehungsweise der Informationsverarbeitung und –systeme wird stark geprägt von einem Grundverständnis nach Gruber (Gruber (1993)). Dieser definiert Ontologien wie folgt:

„An ontology is an explicit specification of a conceptualization.“ (Gruber (1993), S. 199)

Hiernach definiert eine Ontologie also explizit bestimmte Konzepte einer Domäne (engl. *Universe of Discourse*). Innerhalb der Ontologie verbinden Definitionen die Entitäten der Domäne mit Beschreibungen zur Bedeutung und formalen Axiomen, welche die Interpretation der verwendeten Begriffe einschränkt (Gruber (1993), S. 199). Die Argumentation von Gruber führte zu einem verbreiteten Verständnis einer Ontologie als *formale, explizite Spezifikation einer gemeinsamen Konzeptionalisierung in einer Domäne von Interesse* (Grimm et al. (2007), S. 69, Ehrig (2007), S. 12.).

2.2.1.3 Formen von Ontologien

Die Definition von Ontologien sind, wie in 2.2.1.2 beschrieben, sehr allgemein gehalten. Daher ist es sinnvoll verschiedene Formen von Ontologien zu betrachten und voneinander abzugrenzen, um für den Verlauf der vorliegenden Arbeit ein gemeinsames Verständnis der zu behandelnden Ontologien und Ontologiesprachen zu erreichen. Generell können viele konzeptuelle Modelle der Informationsverarbeitung als Ontologien verstanden werden. Hierzu gehören Datenbank-Schemata, UML Modelle, Verzeichnisse, Thesauri oder Folksonomies (Euzenat und Shvaiko (2007), S. 29).

Beispiele für verschiedene Formen von Ontologien und deren Ordnung anhand ihrer Expressivität finden sich in Abbildung 7. Hiernach reichen die Varianten von Ontologien von „einfachen“ Glossaren und Verzeichnissen über strukturierte Glossare und formale Hierarchien in Taxonomien bis hin zu Datenmodellen und formalen Ontologien auf Basis von Beschreibungslogik.

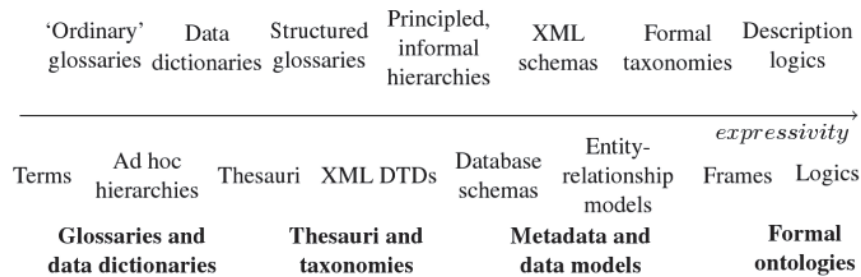


Abbildung 7: Formen von Ontologien nach Euzenat und Shvaiko (2007) in Anlehnung an Uchold und Gruninger (2004)

In der vorliegenden Arbeit werden vornehmlich sogenannte formale Ontologien und deren Repräsentation, also Sprachen, behandelt. Die Nutzung formaler Ontologien ist nötig um den Erfordernissen des *Semantic Web* gerecht zu werden, welches die Maschinenlesbarkeit und – verarbeitbarkeit fordert. Hierzu werden im folgenden Kapitel auch Beispiele unterschiedlicher Ontologiesprachen gegeben.

2.2.2 Semantic Web

Dem von Berners-Lee et al. (2001) oft zitierten „semantischen Web“ liegt der Gedanke zugrunde, semantische Informationen des World Wide Web für Maschinen lesbar und somit auch verarbeitbar zu machen. Um eine solche Vision umzusetzen, bedarf es der Anreicherung der bestehenden Daten mit zusätzlichen Informationen zu ihrer Bedeutung, sogenannten semantischen Annotationen.

2.2.2.1 Ontologiesprachen

Seit den Anfängen des Semantic Web haben verschiedene Gruppen unterschiedliche Sprachen zu Repräsentation von Wissen entwickelt. Im Rahmen der Forschung zu Semantic Web Service werden im Folgenden RDF(S), OWL sowie WSML vorgestellt.

RDF(S): Das Resource Description Framework (RDF) ist eine Sprache zur Repräsentation von Informationen über Ressourcen im World Wide Web. Seit 2004 ist RDF eine Empfehlung des W3C, um die Spezifikation zu verbreiten und somit die Interoperabilität und Funktionalität des Webs zu erhöhen. Ursprünglich ist RDF vor allem für die Repräsentation von Meta-Daten (Autor, Datum, Version, etc.) von Webseiten entwickelt worden. Basierend auf URIs (Uniform Resource Identifiers) beschreibt RDF in Graphen Attribute und Beziehungen zwischen Ressourcen. RDF verwendet 3-Tupel (Tripel) Aussagen (*engl.* Statements), bestehend aus Subjekt, Prädikat und Objekt, wobei Subjekt und Objekt die Knoten des Graphen sind. Eine Vielzahl dieser einzelnen Aussagen kann verwendet werden, um komplexe Zusammenhänge und einfache Bedeutungen auszudrücken sowie logische Schlussfolgerungen aus den Konzepten zu ziehen. Die Semantik von RDF wurde hierzu stark formalisiert. RDF erlaubt es zudem Aussa-

gen über Aussagen zu treffen, die sogenannte *Reification* (*dt.* – Konkretisierung). Weiterhin existiert die Möglichkeit der Serialisierung von RDF als XML. RDF wird erweitert durch das RDF Schema (RDFS), um ein Vokabular zur Verwendung in RDF zu erstellen, zum Beispiel in Form bestimmter Klassen und ihrer Hierarchien. Die Kombination von RDF und RDFS wird oftmals als RDF(S) bezeichnet. (Brickley und Guha (2004); Klyne und Carroll (2004); Manola und Miller (2004); Grimm et al. (2007), S.83-84)

Die Web Ontology Language (OWL) ist seit 2004 eine Empfehlung des W3C. OWL ist ein Nachfolger der DARPA Agent Markup Language⁶ (DAML) (Hendler und McGuinness (2000)) und dem Ontology Inference Layer (OIL) Connolly et al. (2001). OWL zielt auf die Verwertung von Informationen durch Maschinen ab, um das Semantic Web zu realisieren. Sie erlaubt es, die Bedeutung bestimmter Begriffe in einem Vokabular und die Beziehungen zwischen den Begriffen darzustellen. Technisch basiert OWL wie seine Vorgänger auf RDF(S). Zusätzlich zu den einfachen Beziehungen von RDF definiert OWL eine mächtigere Semantik in Form von Beziehungen zwischen Klassen (zum Beispiel Disjunktion), Kardinalität, Gleichheit, Charakteristika von Beziehungen (zum Beispiel Symmetrie) und andere. Um verschiedenen Anforderungen der Anwendung gerecht zu werden, vor allem Ausdrucksstärke und Skalierbarkeit von Schlussfolgerungen, stellt OWL drei Untersprachen zur Verfügung: OWL Lite, OWL DL und OWL Full. Die einfachsten Konzepte bietet OWL Lite, zum Beispiel Klassenhierarchien und einfache Kardinalität. OWL DL (Description Logic) basiert auf der sogenannten Beschreibungslogik als Form der Wissensrepräsentation (Baader et al. (2003)). Sie bietet eine maximale Form der Ausdrucksstärke, wobei gleichzeitig Vollständigkeit (*engl.* Completeness⁷) und Entscheidbarkeit (*engl.* Decidability⁸) der Entscheidungsberechnungen garantiert werden können. (McGuinness und van Harmelen (2004); Grimm et al. (2007), S. 87ff)

Seit 2009 ist OWL in der Version 2 verfügbar. Erweitert wurde die Ausdrucksstärke von OWL 1, unter anderem durch Eigenschaftsverkettung, zusätzliche Datentypen und – bereiche sowie Asymmetrie, Reflexivität und Disjunktion von Eigenschaften. (W3C OWL Working Group (2009))

Die Web Service Modeling Language (WSML) ist Teil des WSMO⁹ (Web Service Modeling Ontology) Frameworks zur ganzheitlichen Beschreibung von semantischen Web Services. Die Verwendung von WSMO für Web Services wird in Abschnitt 2.2.3.2 weiter detailliert. WSML wurde entwickelt, um alle Aspekte des WSMO Frameworks zu unterstützen und gleichzeitig verschiedene Arten der Formalisierung (wie Beschreibungslogik, Logische Programmierung und Prädikatenlogik) bereitzustellen (de Bruijn

⁶ www.daml.org

⁷ Alle Folgerungen können garantiert berechnet werden.

⁸ Alle Berechnungen können in begrenzter Zeit abgeschlossen werden.

⁹ www.wsmo.org

et al. (2006), S. 591). Da WSMML explizit zur Beschreibung von Web Services entwickelt wurde, ist sie keine reine Ontologiesprache. Sie besitzt SWS-spezifische Konzepte wie *goal*, *web service*, *interface*, *choreography* oder *capability*, um verschiedene Aspekte der Semantik von Web Services beschreiben zu können (Grimm et al. (2007), S. 98-99). Zur Realisierung der oben genannten Arten der Formalisierung bietet WSMML eine Vielzahl an Dialekten: WSMML-Core (basierend auf Horn-Logik Horn (1951) und Beschreibungslogik (Baader et al. (2003))), WSMML-Flight (basierend auf F(rame)-Logic (Kifer et al. (1995))), WSMML-Rule (Erweiterung von Flight durch Logische Programmierung), WSMML-DL (basierend auf reiner Beschreibungslogik) und WSMML-Full (Vereinigung von Rule und DL) (de Bruijn et al. (2006), S. 591-592). Abhängig vom verwendeten Dialekt können unterschiedliche Konzepte der Sprache genutzt werden. Die Syntax von WSMML beinhaltet sowohl eine konzeptionelle (Nutzung von Konzepten, Relationen und Instanzen) als auch eine logische Ebene (Nutzung von Axiomen/logischen Ausdrücken) (de Bruijn et al. (2006), S. 599). WSMML kann in Form von RDF und XML serialisiert werden.

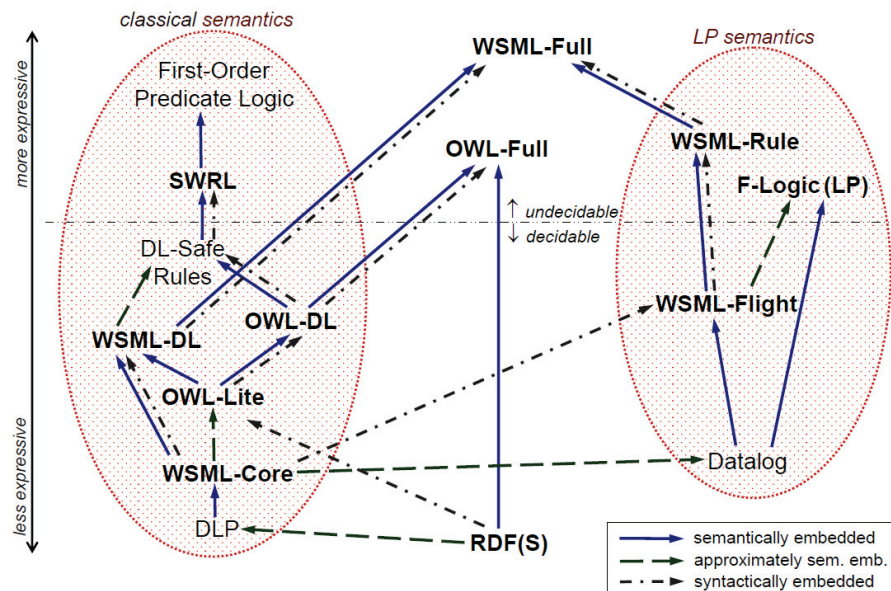


Abbildung 8: Übersicht aktueller Semantic Web Sprachen, nach Grimm et al. (2007), S. 80.

Basierend auf der Arbeit von Grimm et al. (2007) gibt Abbildung 8 eine Übersicht über die verschiedenen Ontologiesprachen und ordnet sie bezüglich ihrer Ausdrucksmöglichkeit, Entscheidbarkeit und Abhängigkeit untereinander an.

2.2.3 Semantic Web Services (SWS)

Im Zusammenhang mit Web Services ist es vor allem wünschenswert, die angebotenen Funktionalitäten der Schnittstellen zu beschreiben. Eine rein syntaktische Beschreibung in Form von WSDL-Dateien ist nicht ausreichend, um Informationen über beispielsweise Vorbedingungen und Effekte sowie andere Attribute wie Quality of Service zu vermitteln. Genau solche Informationen sind notwendig, um Services (halb-) automatisch auffinden, vergleichen und kombinieren zu können und so einen echten Mehrwert zu schaffen. (McIlraith et al. (2001), S. 46)

Ein Vergleich unterschiedlicher Initiativen zur semantischen Beschreibung von Web Services findet sich in Tabelle 1.

Annotation	OWL-S	WSMO	SAWSDL	SWSF	USDL
Herkunft	DAML-S/OWL	WSMF/DERI	WSDL-S/WSDL-ext	SWSL/SWSI	OWL
Ziel	“Describe properties & capabilities of Web services in unambiguous, computer-interpretable form”	“Describe all relevant aspects related to general services accessible through a WS interface [...] enabling automation of WS integration”	“Provide mechanisms for annotating WSDL documents referencing concepts of external semantic models”	“Provide language and ontological structure to support incorporation of machine understandable semantics into Web services”	“Formal description of semantics of web services to support discovering, deploying, composing & synthesizing”
Grad der Formalität	Hoch	Hoch	Mittel	Hoch	Mittel
Grad der Standardisierung	Mittel	Mittel	Mittel-Hoch	Niedrig	Sehr niedrig
Meta-Model	Ja	Ja	Nein	Ja	Unbekannt
Werkzeugunterstützung	- Editors - Matcher - Axis Plugin - Edipse Plugin - Annotations-hilfen - DL/XML mappings	- Editor - Edipse Plugin - Ausführungs-umgebung - Matchers - Reasoners	- Editors - UDDI - WSMO and OWL-S groundings - RDF mapping	- Begrenzte Integration in bestehende Werkzeuge	- Nicht öffentlich verfügbar

Tabelle 1: Übersicht von SWS Ansätzen

Die wichtigsten Annotationstechniken aus Tabelle 1 (OWL-S, WSMO und SAWSDL) werden in den folgenden Unterabschnitten kurz vorgestellt.

2.2.3.1 Web Ontology Language for Services (OWL-S)

Die *Web Ontology Language for Services* basiert auf der W3C Empfehlung OWL (Web Ontology Language) (McGuinness und Harmelen (2004)). OWL wurde für OWL-S verwendet, um eine sogenannte *Upper Ontology* für Services zu erstellen und so ein Meta-Modell zur Beschreibung von Services bereitzustellen. OWL-S ist nicht standardisiert, existiert aber als sogenannte *Member Submission*¹⁰ des W3C. Das Ziel von OWL-S ist die Entwicklung einer von Maschinen interpretierbare Beschreibung von Web Services (nicht gleichzusetzen mit dem Standard Web Service oder WSDL. (Martin et al. (2004))

OWL-S ist direkter Nachfolger von DAML-S, einer der ersten Ansätze zur Beschreibung von Semantik in Web Services. Das Grundkonzept der Nutzung von DAML als Annotation von Services wird in McIlraith et al. (2001) erläutert.

Die Motivation für die Nutzung von OWL-S liegt vor allem in drei Aspekten: Automatisches Auffinden von Web Services, automatisches Aufrufen von Web Services und automatisches Komposition und Zusammenwirken von Web Services. Für das Auffinden von Services stellt OWL-S Anzeigen (*engl.* Advertisements) über die Eigenschaften und Funktionalitäten der Services zur Verfügung. Zum automatischen Aufrufen können mit Hilfe von OWL-S deklarative APIs unter Zuhilfenahme von Domänenontologien definiert werden. Eine automatische Komposition wird unterstützt, indem mit OWL-S Vorbedingungen und Effekte der Services beschrieben werden können und gleichzeitig eine Sprache zur Beschreibung von Servicekompositionen und Datenflussinteraktionen definiert wird. (Martin et al. (2008))

Als Ontologie besteht OWL-S aus einem vordefinierten Vokabular und Axiomen, welche die Konzepte der Ontologie einschränken. Mittels der Axiome werden vor allem Hierarchien gebildet, Klassenbeschreibungen definiert und Beziehungen zwischen den Klassen beschrieben. Um die Entscheidbarkeit (siehe Abschnitt 2.2.2.1) zu garantieren, wurde OWL-S so entwickelt, dass sie in OWL-DL ausgedrückt werden kann. Zusätzliche Ausdrucksstärke wird durch die Möglichkeit der Nutzung anderer logischer Sprachen in den Vorbedingungen und Effekten ermöglicht. (Martin et al. (2007), S. 246)

¹⁰ Eine Einreichung durch Mitglieder des W3C ist jederzeit möglich. Das W3C Team entscheidet über eine Veröffentlichung auf der Webseite. Eine Einreichung ist kein Standard. Siehe auch www.w3.org/Submission/.

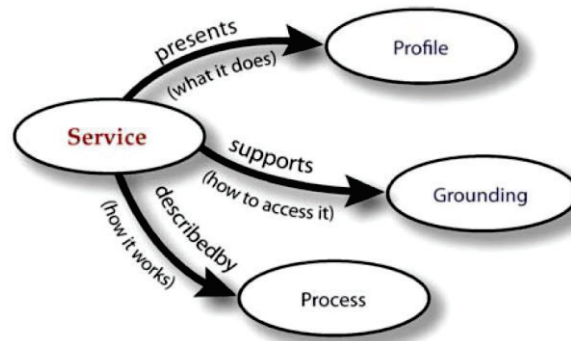


Abbildung 9: Oberste Ebene der Service Ontologie von OWL-S, nach Martin et al. (2007), S. 247

OWL-S besteht aus drei Unterontologien (*engl.* Sub-Ontologies): *Profile*, *Process Model* und *Grounding* (siehe Abbildung 9). *Profile* liefert Informationen über die Funktionalität eines Service, bezieht sich dabei aber hauptsächlich auf die nach außen sichtbare Schnittstelle desselben und ermöglicht dadurch unter anderem auch den Vergleich von Services. Somit wird beschrieben, „was der Service tut“. *Process Model* beschreibt wie der Service intern arbeitet und basiert dabei, wie der Name schon sagt, auf einem abstrakten Prozessmodell. Das *Process Model* ist die Beschreibung, „wie der Service genutzt werden kann“. Der eigentliche Aufruf eines real existierenden Service wird durch das *Grounding* ermöglicht, welches dabei Informationen zu technischen Details gibt, die zum Beispiel die WSDL-Beschreibungen noch erweitern. Das *Grounding* ist notwendig, um zu wissen „wie mit dem Service interagiert werden kann“. Jeder Service wird in OWL-S durch eine Instanz der Klasse *service* repräsentiert, welche durch seine Eigenschaften mit einem *Process Model* (Instanz der Klasse *process*), einem oder mehreren *Groundings* (Instanzen der Klasse *grounding*) und optional mit einem oder mehreren *Profiles* (Instanzen der Klasse *profile*) verbunden ist. Mehrere *Profiles* sind erlaubt, um Anzeigen des Service in unterschiedlichen Kontexten zu ermöglichen. Mehrere *Groundings* sind nötig, da ein abstrakter Service durchaus von mehreren Serviceimplementierungen realisiert werden kann. (Martin et al. (2007), S. 247-248)

OWL-S schreibt nicht vor, welcher Mechanismus der Verankerung (*engl.* Grounding) zur Verknüpfung mit Web Service Implementierungen verwendet werden soll. Allerdings stellt OWL-S eine vordefinierte Verankerung für WSDL zur Verfügung. Dabei werden einzelne Konzepte der Servicebeschreibung auf Elemente einer WSDL Schnittstelle abgebildet (zum Beispiel *atomic process* auf eine WSDL Operation, *input* und *output* auf *input* und *output message parts*). (Lausen et al. (2007), S. 205)

Die umfangreichste Möglichkeit einer Beschreibung des Service wird somit durch das *Profile* möglich, welches speziell *Inputs*, *Outputs* sowie *Preconditions* und *Results* (IOPR) formalisiert. Eingaben sind die Objektbeschreibungen, mit denen der Service arbeitet. Ausgaben sind die Objektbeschreibungen, welche der Service generiert. Diese Werte

sind allerdings nicht spezifisch zu definieren, sondern als Typen in Form von OWL-Klassen. Eine Vorbedingung ist eine Aussage, welche erfüllt („wahr“) sein muss, damit der Service wirksam ausgeführt werden kann. Resultate oder Effekte sind Aussagen, welche erfüllt werden, wenn der Service (erfolgreich) ausgeführt wurde. Meist sind diese Aussagen nicht bezogen auf die Werte der Eingaben und Ausgaben, sondern über die Entitäten auf welche durch die Eingaben und Ausgaben referenziert wird. (Martin et al. (2007), S. 248)

Die Erstellung von Servicebeschreibungen auf Basis von OWL-S wird unterstützt durch ein Plug-In¹¹ für den Semantic Web Editor Protegé¹² der Universität Stanford (Elenius et al. (2005)). Hierzu wird ein bestehendes Plug-In für OWL verwendet (Knublauch et al. (2004)).

2.2.3.2 Web Service Modeling Ontology (WSMO)

Die *Web Service Modeling Ontology*¹³ ist ein Meta-Model zur Beschreibung von unterschiedlichen Aspekten semantischer Web Services, einschließlich Definitionen der Struktur und Semantik von Meta-Daten. Auf Basis von vier Konzepten – *Ontologies*, *Web Services*, *Mediators* und *Goals* – zielt WSMO darauf ab die Automatisierung der Web Service Integration, bestehend unter anderem aus Auffinden, Komposition, Mediation und Ausführung von Services, zu erreichen. Dabei stellen die genannten Konzepte ähnliche Funktionalitäten zur Verfügung. Sie besitzen Beschreibungen nichtfunktionaler Eigenschaften, können externe Ontologien importieren sowie Mediatoren verwenden. Zudem wird jedes Datenmodell in WSMO als eine Ontologie repräsentiert. Im Sinne von semantischen Annotationen beschreibt WSMO Web Services als abstrakte Entitäten, welche zusätzlich zu den bereits erwähnten Konzepten vor allem durch *Capabilities* (Fähigkeiten) und *Interfaces* (Schnittstellen) beschrieben werden. (Roman et al. (2005), S. 78ff)

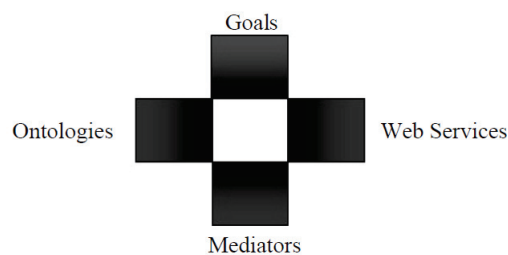


Abbildung 10: Kernelemente von WSMO, nach Roman et al. (2005), S. 80.

¹¹ owlseditor.semwebcentral.org/download.shtml

¹² protege.stanford.edu, aktuell in der Version 3.4.5 verfügbar.

¹³ www.wsmo.org

WSMO basiert konzeptionell auf dem Web Service Modeling Framework (WSMF), erweitert dieses und stellt eine formale Ontologie sowie notwendige Sprachen zur Verfügung (Lausen et al. (2007), S. 182). Die Elemente der obersten Ebene des WSMO Modells sind in Abbildung 10 dargestellt. Diese Elemente werden im Folgenden kurz erläutert.

Ontologien (Konzept *ontologies*) liefern die Terminologie, welche von den anderen Elementen von WSMO verwendet werden kann, um relevante Aspekte einer bestimmten Domäne auszudrücken (Roman et al. (2005), S. 79). Obwohl die Entwickler von WSMO auf bereits bestehende Sprachen hinweisen, gibt es ihrer Meinung nach keine Sprache mit ausreichender Expressivität zur Beschreibung von Web Services auf einer ausreichenden Ebene der Granularität (Roman et al. (2005), S. 82). Eine Diskussion über etwaige Mängel oder eine Referenz auf eine solche findet sich in der vorliegenden Publikation aber leider nicht. Wie bereits erwähnt, verwendet WSMO als Sprache WSML und kann somit aus verschiedenen Dialekten mit unterschiedlicher Expressivität wählen (siehe hierzu auch Abbildung 8).

Web Services bestehen aus den Fähigkeiten (Konzept *capability*), Schnittstellen (Konzept *interface*) und dem internen Ablauf des Service. Diese Aspekte werden mit Hilfe der Konzepte der Ontologien beschrieben (Roman et al. (2005), S. 79). Es wird ein Meta-Model zur Verfügung gestellt um einheitlich und explizit alle Aspekte eines Web Service zu beschreiben. Dabei unterstützt WSMO WSDL, beschränkt sich aber nicht auf diese Form der Interaktion (Roman et al. (2005), S. 88).

Die Fähigkeiten eines Web Service werden primär durch seine *Pre-* und *Postconditions* (Vor- und Nachbedingungen) sowie *Assumptions* und *Effects* (Annahmen und Effekte) weiter detailliert. Vorbedingungen beschreiben, welche Informationen vom Web Service benötigt werden, um den gewünschten Wert zu generieren. Annahmen beschreiben den Zustand der Welt vor der Ausführung des Service. Sind die Annahmen nicht erfüllt, kann nicht garantiert werden, dass der Web Service erfolgreich ausgeführt werden kann. Nachbedingungen beschreiben den Zustand des Informationsraums nach Beendigung des Web Service. Weiterhin kann beschrieben werden, welche Beziehung zwischen den initialen Informationen und den Resultaten des Web Service besteht. Die Effekte erlauben schließlich den Zustand der Welt nach der Ausführung, gesetzt den Fall, dass alle Vorbedingungen und Annahmen erfüllt wurden. Alle diese Aspekte werden dabei formal als Axiome in WSML (siehe Abschnitt 2.2.2.1) definiert, wodurch die Nutzung automatisierter Schlussfolgerungen (sog. Reasoning) ermöglicht wird. Dadurch entsteht auch die Möglichkeit neben einfacher textbasierter Suche nach Services auch komplexe semantische Abfragen durchzuführen. (Keller et al. (2007), S. 289ff)

Ziele (Konzept *goals*) beschreiben die Wünsche der Anwender nach einer gesuchten Funktionalität. Domänenspezifische Konzepte in Ontologien können verwendet werden, um diese Ziele zu definieren. Ziele werden ähnlich den Web Services formuliert, jedoch aus Sicht des Service-Nutzers (Roman et al. (2005), S. 79-80). Hierzu nutzen Ziele auch die Beschreibung von Fähigkeiten. Dabei muss keine komplette Beschreibung von Annahmen, Vorbedingungen, Effekten und Nachbedingungen vorhanden

sein. So kann zum Beispiel nur eine Nachbedingung beschrieben werden, entsprechende Services können nach dem Auffinden zusätzliche Anforderungen zum Aufruf besitzen. Weiterhin kann mit dem Ziel definiert werden, welche Form der Interaktion der Anwender über eine Schnittstelle wünscht. (Roman et al. (2005), S. 93)

Mediatoren (Konzept *mediators*) sind Elemente zum Umgang mit der Interoperabilität zwischen den verschiedenen WSMO Konzepten. Probleme der Interoperabilität können mit Hilfe der Mediatoren auf Daten-, Prozess- und Protokollebene behoben werden. (Roman et al. (2005), S. 80)

2.2.3.3 Semantic Annotations for WSDL and XML Schema (SAWSDL)

SAWSDL ist eine Weiterentwicklung der WSDL-S (WSDL-Semantics) Initiative, welche auch als Mitgliedseinreichung bei der W3C vorliegt. Das Ziel von WSDL-S ist die Bereitstellung eines leichtgewichtigen Ansatzes zur Erweiterung von Web Service Beschreibungen um Semantik. Dabei wird keine neue Beschreibungssprache definiert, sondern auf die Erweiterungsmöglichkeiten von WSDL gesetzt, indem Referenzen auf Inputs, Outputs sowie Preconditions und Effects eingeführt werden. Die referenzierten Modelle befinden sich jedoch außerhalb der WSDL Beschreibungen, wodurch ein Ontologie-agnostischer Ansatz entsteht. (Akkiraju et al. (2005))

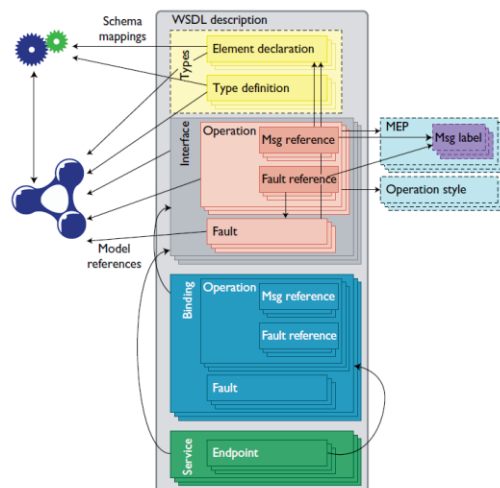


Abbildung 11: SAWSDL Konzept nach Kopecký et al. (2007), S. 61.

Den Ansatz von WSDL-S erweitert SAWSDL um die Möglichkeit neben semantischen Modellen auch Schema-Mappings zu referenzieren, wodurch Daten auf eine Ontologie-Ebene angehoben (*Lifting*) oder von einer solchen abgesenkt (*Lowering*) werden können. Hierdurch soll vor allem die Interoperabilität von Services erhöht werden. Generell können die erwähnten Referenzen Operationen, Fehlermeldungen und Schnittstellen innerhalb einer WSDL-Beschreibung detaillieren und liefern somit eine weitere Möglichkeit, die Funktionalität eines Service zu beschreiben. (Farrell und Lausen (2007))

2.3 Ontology Matching und Alignment

Wie in Abschnitt 2.2.1 erläutert gibt es verschiedene Darstellungen und Formen von Expressivität für Ontologien. Außerdem existieren eine Vielzahl unterschiedlicher Sprachen, um Ontologien auszudrücken (siehe Abschnitt 2.2.2.1). Daher ist es nicht verwunderlich, dass es in der Verwendung von Ontologien untereinander zu Problemen durch verschiedene Arten von Heterogenität kommen kann. Euzenat und Shvaiko (2007), S. 41-42 unterscheiden hierzu vier Formen der Heterogenität von Ontologien:

Syntaktische Heterogenität lässt sich zurückführen auf die mangelnde Interoperabilität zweier Ontologien in unterschiedlichen Ontologiesprachen. Ein Beispiel ist der Vergleich eines einfachen Verzeichnisses mit formalen konzeptuellen Modellen. Zusätzlich treten Probleme auch auf der Ebene von Modellen gleicher oder ähnlicher Expressivität auf. Hier gibt es das Problem unterschiedlichen Formalismen, wie zum Beispiel OWL und F-Logic. Um dies zu bewältigen wird oft auf die Ebene der jeweiligen formalen Sprache zurückgegriffen und die Konstrukte dieser miteinander in Einklang gebracht.

Terminologische Heterogenität basiert auf dem Fall von Unterschieden in der Benennung von denselben Entitäten in unterschiedlichen Ontologien. Dies kann durch die Nutzung unterschiedlicher Sprachen passieren, z.B. *Artikel* und *Paper*, oder auch durch die Verwendung von Synonymen.

Konzeptionelle Heterogenität, von Klein (2001) auch als logische Diskrepanz (engl. *Logical mismatch*) bezeichnet, entsteht bei Unterschieden in der Modellierung der gleichen Domäne. Unterschiede können sowohl in der Konzeptionalisierung, also dem Ausdruck unterschiedlicher Konzepte innerhalb der Domäne, als auch in der Repräsentation dieser liegen. Um vor allem den Umstand der Unterschiede in der Konzeptionalisierung zu verstehen seien im Folgenden drei Kategorien dieser Unterschiede definiert:

- *Unterschiede in der Abdeckung*: Zwei Ontologien beschreiben verschiedene, wenn auch möglicherweise überlappende Bereiche einer bestimmten Domäne, zum Beispiel geographische Regionen.
- *Unterschiede in der Granularität*: Zwei Ontologien beschreiben die gleichen Bereiche einer Domäne, allerdings auf verschiedenen Ebenen der Detaillierung.
- *Unterschiede in der Perspektive*: Zwei Ontologien beschreiben die gleichen Bereiche einer Domäne, auf gleicher Ebene der Detaillierung, aber aus unterschiedlicher Perspektive. Ein Beispiel ist der Unterschied zwischen geographischen und sozio-politischen Karten einer Region.

Semiotische Heterogenität bezeichnet den Umgang mit unterschiedlicher Interpretation gleicher Entitäten durch verschiedene Personen. Entitäten mit ursprünglich gleicher semantischer Interpretation werden oft durch Unterschiede im Kontext oder ihrer Nutzung anders wahrgenommen. Durch die Lücke zwischen maschineller Interpretation

und realer Welt beziehungsweise Kontext ist diese Form der Heterogenität schwer zu lösen und wird somit in den meisten Lösungsansätzen nicht behandelt.

Diese Formen der Heterogenität führen im Umgang mit Ontologien zu Herausforderungen des Abgleich zwischen diesen um eine Interoperabilität möglich oder zumindest teilweise möglich zu machen. Nur wenn eine Interoperabilität teilweise möglich ist können in Zukunft Services unterschiedlicher Anbieter, Domänen und Unternehmen automatisiert aufgefunden und verwendet werden.

2.3.1 Ontology Matching

Der Begriff des Matchings soll im Folgenden als der „Prozess des Auffindens von Beziehungen oder Übereinstimmungen zwischen Entitäten unterschiedlicher Ontologien“ (Euzenat und Shvaiko (2007), S. 42) verstanden werden. Die Definition dieser Beziehungen oder Übereinstimmungen hängt hierbei sowohl von der Form der Heterogenität der Ontologien ab (siehe vorheriger Abschnitt) als auch von der Nutzung bestimmter Algorithmen zur Festlegung dieser. Entitäten sind daher unter anderem Klassen, Individuen, Eigenschaften und Axiome der Ontologien (weitere Formen von Entitäten können existieren). Diese Definition grenzt auch von dem Terminus Mapping ab, welcher sowohl einen Prozess als auch das Produkt dieses Prozesses bezeichnen kann.

Das Set an Übereinstimmungen zwischen zwei oder mehr Ontologien wird als Alignment bezeichnet (Euzenat und Shvaiko (2007), S. 42). Das Alignment ist das Produkt des Matching-Prozesses, kann aber auch wieder als Eingangsparameter für einen neuen Matching-Prozess dienen.

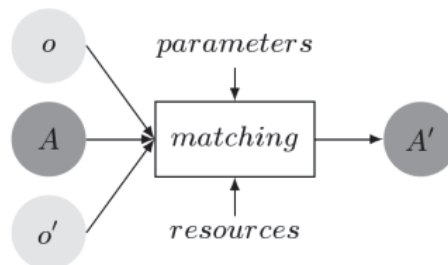


Abbildung 12: Der Matching-Prozess nach Euzenat und Shvaiko (2007), S. 44

Eine schematische Darstellung des Matching-Prozesses ist in Abbildung 12 zu sehen. Danach wird ein Matching durchgeführt über den Ontologien *o* und *o'* unter der Berücksichtigung eines Ausgangs-Alignment *A* (welches auch leer sein kann). Zur Durchführung stehen ein Set an Parametern und Ressourcen zur Verfügung. Das Endprodukt ist das Alignment *A'* zwischen den beiden Ontologien. Diese Definition lässt sich zu-

dem erweitern auf das Matching mehrerer Ontologien, wobei ein Set an Ontologien $\{o_1, \dots, o_n\}$ als Ausgangsbasis dient.

Das Matching unter Nutzung von Ressourcen und Parametern kann auch als Operator bezeichnet werden, ein sogenannter *Match*, welcher als Funktion über zwei Ontologien eine Abbildung (auch Mapping oder Alignment) zwischen diesen erstellt. (Rahm und Bernstein (2001), 336)

Zum besseren Verständnis unterschiedlicher Techniken des Ontology Matchings ist es hilfreich diese zu klassifizieren. Euzenat und Shvaiko (2007) bieten hierzu eine gute Klassifizierung anhand mehrerer Dimensionen, offensichtlich inspiriert von der Arbeit von Rahm und Bernstein (2001) (S. 338).

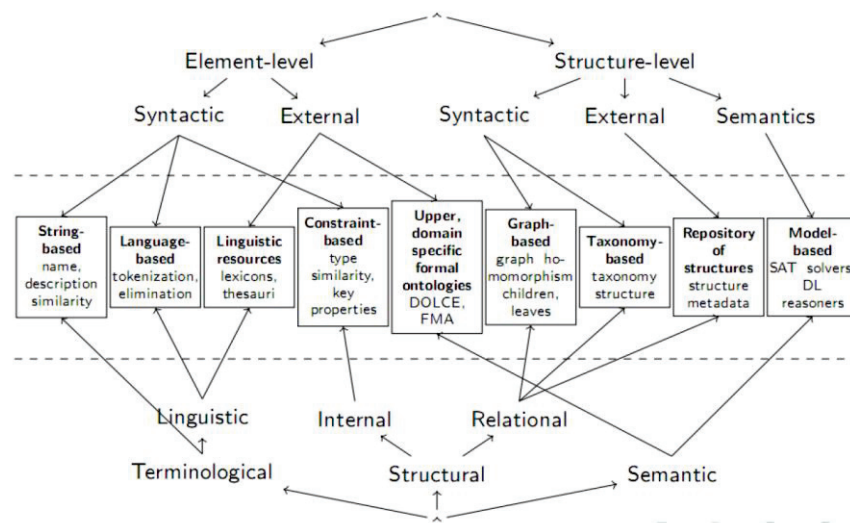


Abbildung 13: Klassifizierung von Matching-Techniken nach Euzenat und Shvaiko (2007), S. 65

Die Basis für die Klassifikation bilden zwei Klassifikationen (siehe Euzenat und Shvaiko (2007), S. 65):

- *Granularität/Input Interpretation*: Diese Form der Klassifikation basiert sowohl auf der Granularität der Eingangswerte (Input), das heißt Elemente oder Strukturen, als auch auf der generellen Interpretation des Inputs durch die jeweilige Matching-Technik (Syntaktisch vs. Semantisch vs. Extern).
- *Art des Inputs*: Diese Klassifikation basiert auf der Form der Eingangswerte für die elementaren Matching-Techniken.

Zur Vollständigkeit soll kurz auf die Unterscheidung von elementaren Matching-Techniken eingegangen werden:

- *Elemente vs. Struktur*: Matching-Techniken analysieren Ontologien zum einen auf Elementebene, zum Beispiel Entitäten und Instanzen, aber ohne die Be-

rücksichtigung von Beziehungen untereinander. Zum anderen kann analysiert werden, in welchem Zusammenhang diese Elemente stehen und inwieweit Strukturen mehrerer Ontologien Ähnlichkeiten aufweisen. Diese elementaren Matching-Techniken werden im eigentlichen Matching-Werkzeug oft kombiniert.

- *Syntaktisch vs. Semantisch vs. Extern*: Der Kern syntaktischer Matching-Algorithmen liegt in der Analyse der Struktur der Ontologien. Im Gegensatz dazu analysieren semantische Techniken die Bedeutung der Entitäten und können somit Ähnlichkeiten feststellen. Externe Techniken greifen auf zusätzliche Ressourcen zu, um die Ontologien zu analysieren. Zum Einsatz kommen hier unter anderem Domänenwissen oder semantische Netze (zum Beispiel WordNet (Miller (1995))).

Auf die einzelnen Techniken aus der vorgestellten Klassifikation (siehe Abbildung 13) soll noch nicht im Detail eingegangen werden. Die Klassifikation wird jedoch zum Verständnis der eingesetzten Matcher im Bereich der prototypischen Implementierung beitragen. Nachdem nun der Matching-Prozess und dessen Arten geklärt sind, wird im Folgenden auf das Ergebnis dieses Prozesses, das *Alignment*, eingegangen.

2.3.2 Ontology Alignment

Zunächst ist der Unterschied zwischen Ontology Alignment und Ontology Matching zu klären, da diese in der Fachwelt sowohl synonym als auch unterschiedlich verwendet werden. So definiert Klein (2001) den Prozess zum Erreichen eines Alignments ähnlich dem des Matchings nach Euzenat und Shvaiko (2007): Zwei oder mehr Ontologien werden zu einer gemeinsamen Übereinstimmung gebracht und somit konsistent und kohärent.

Im Weiteren wird der Begriff des Alignments vornehmlich als Ergebnis des Matching-Prozesses verstanden und als Repräsentation dessen. Dieses Verständnis basiert vornehmlich auf Euzenat (siehe unter anderem Euzenat und Shvaiko (2007); Euzenat (2008)). Formal kann man ein Alignment zwischen zwei Ontologien als folgendes Quadrupel ansehen (nach Euzenat et al. (2004)):

$$\text{Alignment } A: \langle e, f, r, l \rangle,$$

wobei e und f zwei korrespondierende Entitäten sind, r die Relation zwischen diesen Entitäten und l das Level der Sicherheit über der Relation. Dabei ist zu beachten, dass der Umfang eines Alignments abhängig ist von den Fähigkeiten des genutzten Matching-Algorithmus. So können zusätzliche Informationen wie etwa Kardinalität genutzt werden (siehe Euzenat und Shvaiko (2007), S. 226).

Für die vorliegende Arbeit reicht die Definition eines solchen „einfachen“ Alignments aus, um den Umgang mit der Heterogenität der aufgestellten Anwendungsfälle zu gewährleisten. Für die Diskussion komplexerer Alignments sei auf Zimmermann et al. (2006) verwiesen.

Zur Speicherung und Wiederverwendung von Alignments wird eine geeignete Repräsentation benötigt. Hierzu gibt es verschiedene Ansätze. Zum einen bringen die meisten Matching-Werkzeuge eine interne Repräsentation und eine entsprechende Serialisierung in meist proprietäre Formate mit sich. Zum anderen gibt es Bemühungen zur Standardisierung und somit zur Vergleichbarkeit von Alignments.

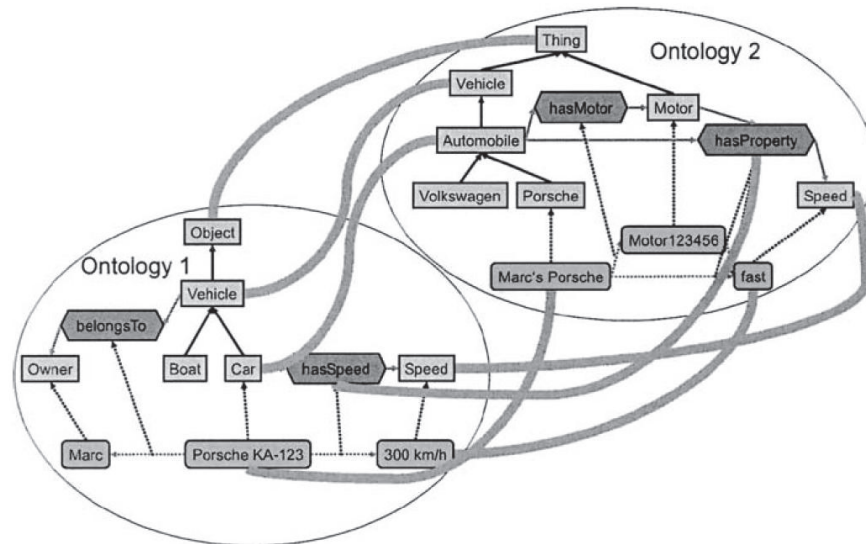


Abbildung 14: Beispiel für das Alignment zweier Ontologien, nach Ehrig (2007), S. 21

Es ist möglich zur Darstellung von Alignments bereits spezifizierte und standardisierte Formate zu verwenden, wie OWL. Hierbei kann man auf bestehende Konstrukte dieser Sprachen wie Konzepte und Relationen zurückgreifen, zum Beispiel zur Darstellung der Gleichheit mittels *owl:equivalentClass*. Der Vorteil liegt in einer ausgereiften Sprache mit der Möglichkeit der direkten Interpretation durch bestehende OWL Reasoner. Der Nachteil liegt vor allem in den begrenzten Möglichkeiten verschiedene Arten der Ähnlichkeit darzustellen, da in OWL meist von einer strikten Gleichheit ausgegangen wird. (Ehrig (2007))

Ein Ansatz für die standardisierte Repräsentation von Alignments wurde durch die *Ontology Alignment Evaluation Initiative*¹⁴ (OAEI) ins Leben gerufen. Das Ziel der Initiative ist die Auswertung der ständig wachsenden Anzahl an *Ontology Alignment* Werkzeugen auf Basis von vordefinierten Benchmarks mit dem Ziel der Vergleichbarkeit. Diese Vergleichbarkeit wird ermöglicht durch die Definition eines standardisierten Alignments Formats auf Basis von RDF und XML.

¹⁴ <http://oaei.ontologymatching.org/>

Das Format wurde im Rahmen der Ontology Alignment API entwickelt und besteht aus den *Referenzen* zu den verglichenen Ontologien, einem *Set an Übereinstimmungen* mit der Darstellung der Relation zwischen den Entitäten, einem *Level* zur Charakterisierung der Übereinstimmung sowie der *Kardinalität*. Unter Verwendung von RDF können weitere Metadaten genutzt werden wie der verwendete Algorithmus und das Erstellungsdatum (Beispielcode siehe Abbildung 15). (McIlraith et al. (2004); Euzenat und Shvaiko (2007))

```
<rdf:RDF>
<Alignment>
  <xml>yes</xml>
  <level>0</level>
  <type>11</type>
  <onto1>ontology1.owl</onto1>
  <onto2>ontology2.owl</onto2>
  <uri1>http://aifb.uni-karlsruhe.de/ontology1.owl</uri1>
  <uri2>http://aifb.uni-karlsruhe.de/ontology2.owl</uri2>
  <map>
    <Cell>
      <ent ity1 rdf:resource='ontology1.owl#object' />
      <ent ity2 rdf:resource='ontology2.owl#thing' />
      <measure rdf:datatype='XMLSchema#float'>1.0</measure>
      <relation>=</relation>
    </Cell>
    <Cell>
      <entity1 rdf:resource='ontology1.owl#vehicle' />
      <entity2 rdf:resource='ontology2.owl#vehicle' />
      <measure rdf:datatype='XMLSchema#float'>1.0</measure>
      <relation>=</relation>
    </Cell>
  </map>
</Alignment>
</rdf:RDF>
```

Abbildung 15: Beispielcode für ein Ontology Alignment nach McIlraith et al. (2004)

Die vorliegende Repräsentation ist unabhängig von Matcher und Ontologiesprache und entspricht somit dem Level 0 des Ontology Alignments Formats nach Euzenat und Shvaiko (2007). Die Darstellung nach Level 1 ist weiterhin unabhängig von einer Ontologiesprache, ersetzt aber die Paare der Entitäten mit Listen von Entitäten, um diese einfacher parsen zu können. Abschließend gibt es die Möglichkeit mit Level 2 sprachenspezifische Übereinstimmungen auszudrücken, zum Beispiel durch die Nutzung einer Logik-basierten Sprache.

Die Einsatzgebiete des Ontology Alignments sind vielschichtig. Ehrig (2007) gibt einige Beispiele:

Web Service Komposition Die Beschreibung von Web Services, vor allem ihrer Funktionalität und Ziele, kann stark variieren. Um in heterogenen Umgebungen trotzdem eine Kollaboration zwischen den Services zu erreichen, können Ontology Alignments eingesetzt werden. Die Anforderungen an ein solches Alignment sind Schnelligkeit, Zuverlässigkeit und Korrektheit, da ansonsten gerade in geschäftskritischen Anwendungen Probleme entstehen und kaskadieren können.

Datenintegration Der Umgang mit Heterogenität von Daten in Unternehmen ist weiterhin ein wichtiges Forschungsthema. Bestehende relationale Datenbanken beziehungsweise deren Schemata können bereits vollautomatisch in Ontologiesprachen über-

setzt (*engl.* lift) werden. Dies ermöglicht die Angleichung und Integration mit Ontology Alignment Mechanismen. Obwohl eine vollautomatische Lösung auf diese Weise noch nicht erreicht werden kann, reduziert die Nutzung von Alignments einen Teil der Integration deutlich.

Ontologie Evolution Bestehende Ontologien werden und sollen sich mit der Zeit verändern. Die Versionierung von Ontologien zum Umgang mit dieser Evolution liegt als Lösung auf der Hand. Um mit verschiedenen Versionen einer Ontologie umgehen zu können ermöglichen Alignments den einfachen Transfer von einer alten in eine neue Version oder umgekehrt. (hierzu auch Shvaiko und Euzenat (2008), S. 1178)

Reasoning Die Nutzung von Informationen des zukünftigen Semantic Web (siehe Abschnitt 2.2.2) oder in anderen Bereichen mit einer Vielzahl von heterogenen und verteilten Daten für Schlussfolgerungen (*engl.* Reasoning) wird gebremst durch die fehlende Interoperabilität dieser Daten. Die Nutzung von Alignments vieler verschiedener Ontologien verspricht ein Reasoning über diesen Ontologien ohne eine explizite Integration. Auch hier wird es in naher Zukunft keine vollautomatischen Lösungen geben. Allerdings können Resultate durch den Nutzer vorausgewertet werden, um eventuelle Inkonsistenzen zu vermeiden. (Hierzu auch Shvaiko und Euzenat (2008), S. 1177)

Merging Abschließend ist eine Abgrenzung des Ontology Alignments zum Ontology Merging notwendig. Nach Klein (2001) bezeichnet das Merging (*dt.* Zusammenführen) von Ontologien „das Erstellen einer neuen Ontologie durch Nutzung zwei oder mehrerer bestehender Ontologien mit überschneidenden Bestandteilen“. Das Alignment kann in diesem Bereich als Vorstufe zum Zusammenführen genutzt werden. Somit wird ein Teil des Prozesses automatisiert und die Veränderungen dokumentiert. Dies ist wichtig, da in der neu entstehenden Ontologie keine Hinweise mehr zu finden sind, welche Teile zu welcher ursprünglichen Ontologie gehören.

2.4 Geschäftsprozessorientierung und -management

Die Begriffe der Orientierung an Geschäftsprozessen und das Management dieser werden nicht nur aus einer organisatorischen Sicht heraus betrachtet, sondern vermehrt direkt als Gestaltungsmittel für Informationssysteme verwendet. Die Prozessorientierung wurde als praktische Organisationsgestaltung bereits zu Beginn der 1990er Jahre durch die Autoren Hammer und Champy (1994) und Davenport (1993) etabliert (siehe Krallmann et al. (2007), S. 30). Die Geschäftsprozessorientierung soll in der vorliegenden Arbeit genutzt werden, um bestehende Informationen in Form von Geschäftsprozessmodellen sowie die Modellierung von Geschäftsprozessen als Kontext für die Suche nach Services zu ermöglichen. Daher ist die Klärung dieser Begriffe im Folgenden unabdingbar.

2.4.1 (Geschäfts-)Prozessorientierung

Das Verständnis des Unternehmens anhand seiner Abläufe beziehungsweise Prozesse weicht von der klassischen Sicht auf den Aufbau des Unternehmens ab. Die Orientierung am Aufbau einer Organisation und ihrer Arbeitssysteme führt zu einer Aufteilung anhand von (oft hierarchischen) Teilsystemen wie Abteilungen oder Stellen. Die Orientierung an Prozessen erlaubt dagegen die Betrachtung von Abläufen über Teilsystemgrenzen hinaus. (Becker und Kahn (2005), S.6 ff)

Die Orientierung am Aufbau kann auch als funktionale Orientierung bezeichnet werden. Die funktionale Organisation zeigt Schwächen unter anderem im Umgang mit schnellen Änderungen des wirtschaftlichen Umfelds, kürzeren Produktlebenszyklen, differenzierten Kundenwünschen und dem Wandel von Anbieter- zu Käufermärkten (Krallmann et al. (2007), S. 30).

Die Prozessorientierung versucht bestehende Nachteile, welche aus den oben genannten Schwächen entstehen können, zu beseitigen. Zu diesen Nachteilen zählen mangelnde Flexibilität, hoher Aufwand der Koordination von einzelnen Vorgängen sowie eine Vielzahl an Schnittstellen im Informationsfluss (Kurz (1993), S. 6 nach Krallmann et al. (2007), S. 30).

Für den Verlauf der Arbeit soll folgende Definition von *Prozess* und *Geschäftsprozess* nach Becker und Kahn (2005) gelten:

Ein *Prozess* ist die inhaltlich abgeschlossene, zeitliche und sachlogische Folge von Aktivitäten, die zur Bearbeitung eines prozessprägenden betriebswirtschaftlichen Objektes notwendig sind.

Ein Geschäftsprozess ist ein spezieller Prozess, der durch die obersten Ziele der Unternehmung (Geschäftsziele) geprägt wird. Wesentliche Merkmale eines Geschäftsprozesses sind die Schnittstellen des Prozesses zu den Marktpartnern des Unternehmens (z.B. Kunden, Lieferanten).

Die Orientierung eines Unternehmens anhand seiner Geschäftsprozesse erlaubt die Etablierung einer Prozessorganisation. Eine solche kann aus verschiedenen Perspektiven betrachtet werden. Zum einen die Unterscheidung von Ablauf und Aufbau eines Unternehmens, wobei in der Prozessorientierung die Aufbauorganisation von der Ablauforganisation bestimmt wird. Weiterhin kann die Prozessorientierung als Mittel zur Koordination verwendet werden, um den Leistungs- und Informationsaustausch zwischen den Bereichen durch definierte Schnittstellen und verbindliche Vereinbarungen zu verbessern. Eine zusätzliche Perspektive versucht „ganzheitliche und schnittstellenfreie Prozesse“ (Krallmann et al. (2007), S. 32) zu nutzen, um die Arbeitsteilung zwischen den Funktionen zu ersetzen (Gaitanides (2004), S. 1209 nach Krallmann et al. (2007), S. 32).

Die Prozessorientierung zielt auf eine höhere Leistungsfähigkeit der Organisation, eine Erhöhung der Flexibilität bei erwarteten und unerwarteten Änderungen und eine besse-

re Beherrschbarkeit der Prozesse ab. Die Leistungsfähigkeit soll vor allem zur verbesserten Erfüllung von Kundenwünschen, auch individuellen, erhöht werden und schlägt sich in der Qualität und der Kundenzufriedenheit mit den Produkten nieder sowie bei der Senkung von Kosten und Zeit zur Erstellung. Erhöhte Flexibilität soll Änderungen einfacher, zeitsparender und mit geringeren Kosten erlauben. Diese Erhöhung der Flexibilität ist auch in der Nutzung von Prozessen in Informationssystemen, zum Beispiel durch Workflow-Engines, ein wichtiges Merkmal. Ein weiteres Merkmal, welches sowohl für die organisatorische als auch die informationstechnische Prozessorientierung gilt, ist die Verbesserung der Beherrschbarkeit der Prozesse (und damit auch deren IT-Unterstützung). Dies soll unter anderem erreicht werden, indem Prozesse anhand bestimmter Metriken messbar gemacht werden. Es können sogenannte Prozesskennzahlen definiert und gemessen werden, um Prozesse transparent und nachvollziehbar zu machen. (Maurer (1996), S. 5-6)

Der sinnvolle Einsatz der Geschäftsprozessorientierung in einem Unternehmen, sei es auf organisatorischer und/oder auf informationstechnischer Ebene, ist allerdings nur möglich, wenn die Prozesse aktiv überwacht und angepasst werden. Um dies zu ermöglichen, ist das Geschäftsprozessmanagement unabdingbar.

2.4.2 Geschäftsprozessmanagement

Einfach ausgedrückt kann ein Geschäftsprozessmanagement beschrieben werden als das Set aller Management-Aktivitäten, welche den Geschäftsprozessen zugehörig sind (Mendling (2007), S. 5). Eine erweiterte Definition präsentieren van der Aalst et al. (2003) (S. 4). Laut den Autoren ist das Ziel des Geschäftsprozessmanagements:

“[To support] business processes using methods, techniques, and software to design, enact, control, and analyze operational processes involving humans, organizations, applications, documents and other sources of information”

Somit wird neben den Management-Aktivitäten nach Mendling auch explizit Software als Unterstützungswerkzeug für die Geschäftsprozesse und deren Management genannt. Abzugrenzen von der Software zur Unterstützung der Geschäftsprozesse und der abstrakten Prozesse an sich ist zusätzlich der Begriff des Workflows, welcher seit den 1990er Jahren vor allem durch die Etablierung von Workflow-Management-Systemen verwendet wird. Die Workflow Management Coalition (WfMC) definiert einen Workflow als die Automation eines Geschäftsprozesses, im Ganzen oder teilweise, wobei Dokumente, Informationen oder Aufgaben von einem Teilnehmer zu einem anderen Teilnehmer weitergeleitet werden, mit dem Ziel einer Handlung anhand festgelegter prozeduralen Regeln (Workflow Management Coalition (1999), S. 8). Hier ist vor allem das Stichwort der Automation hervorzuheben, welche in der vorliegenden Arbeit letztlich durch die Orchestrierung verschiedener Services erreicht werden soll. Realisiert und kontrolliert werden automatisierte Geschäftsprozesse in Form von Workflows durch sogenannte Workflow Management Systeme (WMS). Ein WMS definiert, erstellt und verwaltet die Ausführung der Workflows mittels Software, welche auf einer oder mehreren Workflow Engines ausgeführt wird und sowohl die Prozessdefinitionen

interpretiert als auch mit den Teilnehmern des Workflows interagiert und wenn nötig weitere IT Applikationen aufrufen kann (Workflow Management Coalition (1999), S. 9).

Für ein besseres Verständnis des Geschäftsprozessmanagements wird im Folgenden ein managementorientierter Prozesslebenszyklus nach zur Mühlen (2004) verwendet. Dieser besteht aus sieben Haupt- und Nebenphasen: Analyse, Prozessdesign, Prozessimplementierung, Prozessdurchführung, Prozessevaluierung, Prozessüberwachung sowie Animation und Simulation der Prozesse (siehe Abbildung 16). Der Lebenszyklus bedient sich dabei mehrerer bestehender Lebenszyklen, integriert diese und erweitert sie vor allem um Artefakte zwischen den einzelnen Phasen.

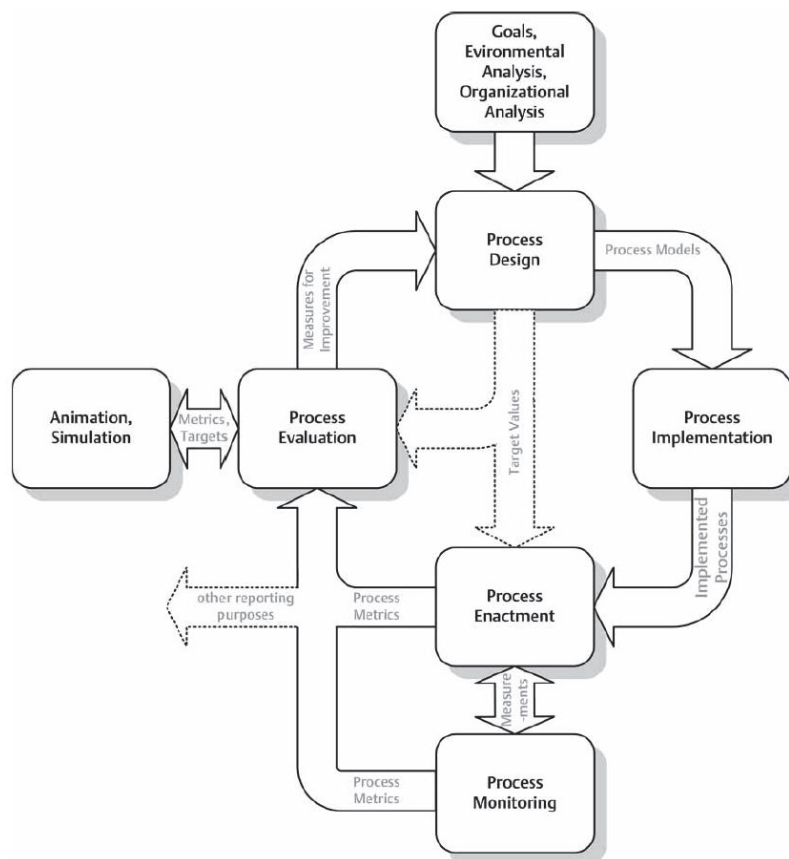


Abbildung 16: Prozesslebenszyklus nach zur Mühlen (2004), S. 86.

Die initiale Analyse der Projektziele, der Organisationsstrukturen und der Applikations- sowie Systemumgebung steht zu Beginn des Lebenszyklus. Anschließend wird im Prozessdesign die gesamtheitliche Struktur erstellt, die einzelnen Prozessmodelle entworfen und die benötigten Ressourcen für die Ausführung sowie die Verantwortlichen dieser

definiert. Die somit erstellten Prozessmodelle können als Artefakte in die Prozessimplementierung übergeben werden. Somit kann die Infrastruktur der Geschäftsprozessunterstützung entworfen und implementiert werden. Weiterhin wird die Lösung, wenn nötig, in bestehende Informationssysteme integriert. Während der Prozessdurchführung können aus den definierten Prozessmodellen einzelne Instanzen abgeleitet und instanziiert werden. Somit ist es möglich Prozesse automatisch oder manuell anzustoßen und zu koordinieren. Vordefinierte Teilnehmer des Prozesses werden benachrichtigt und zu Handlungen aufgefordert. Gleichzeitig werden in der Prozessüberwachung administrative Prozesse ausgeführt, welche die Performanz und weitere Prozesskennzahlen wie Durchlaufzeit, Wartezeit oder eingesetzte Ressourcen messen. Die Prozessevaluierung beendet den Hauptzyklus des Geschäftsprozessmanagement. Auf Basis bestehender Daten (zum Beispiel Log-Dateien) kann ex-post die Vielzahl an durchgeführten Prozessen einzeln oder kumuliert analysiert werden. Auf Basis solcher Analyse kann im Anschluss der Lebenszyklus neu angestoßen werden und ein überarbeitetes Design die bestehenden Prozesse verbessern. Als Unterstützung der Prozessevaluierung können Animationen und Simulationen ein besseres Verständnis zur Interpretation der Daten schaffen und Änderungen an den Prozessen können hinsichtlich Ressourcenverbrauch sowie weiterer Kennzahlen getestet werden. (zur Mühlen (2004), S. 85-87)

2.4.3 Geschäftsprozessmodellierung

Die Nutzung der Geschäftsprozessorientierung und das aktive Management der Prozesse setzen voraus, dass diese in einer sinnvoll nutzbaren Form repräsentiert und persistent gemacht werden können. In der Praxis werden hierzu Prozessmodelle verwendet. Die Nutzung einer rein textuellen Beschreibung der Modelle ist zwar möglich und auch nötig, allerdings ist bedingt durch die kognitive Fähigkeit der Menschen eine grafische Repräsentation einfacher und sinnvoller. (siehe hierzu zum Beispiel Mayer (1997) und Scaife und Rogers (1996))

Geschäftsprozessmodelle sind in der vorliegenden Arbeit ein wichtiger Faktor zur Kontextualisierung und Informationsgewinnung bei der Generierung von Suchanfragen sowie zur Speicherung und Weiterverwendung von Suchergebnissen. Daher soll in diesem Abschnitt zunächst der allgemeine Modellbegriff geklärt werden, um aufbauend darauf Geschäftsprozessmodelle sowie deren Ziele und Merkmale zu definieren. Anschließend werden die vorherrschenden Modellierungsformen beziehungsweise –sprachen vorgestellt, welche teilweise im weiteren Verlauf der Arbeit für den entwickelten Ansatz verwendet werden können.

2.4.3.1 Modellbegriff

Ein allgemeines Verständnis von Modell wird von Stachowiak (1973) definiert. Darin ist ein Modell eine Abbildung eines bestehenden Systems oder Objektes. Herausgestellt werden die Abstraktion und die Zweckorientierung des Modells. Weiterhin wird ein Modell durch das Verkürzungsmerkmal, das Abbildungsmerkmal und das pragmatisches Merkmal charakterisiert. Das Abbildungsmerkmal definiert, dass ein Modell auf

einem Original basiert, welches allerdings auch ein Modell sein kann. Die Verkürzung oder auch Reduktion führt zu einer Repräsentation eines für den Modellierer relevanten Teils des Originals. Zusätzlich ist das Modell pragmatisch zu erstellen bezüglich der Nutzung an Stelle des Originals, zu einem bestimmten Zweck und für eine bestimmte Zielgruppe an Modellnutzern. (Stachowiak (1973), S. 131ff und Kühne (2006), S. 3)

2.4.3.2 Definition von Geschäftsprozessmodellen, -modellierung und -modellierungssprachen

Nach einer ausführlichen Literaturrecherche und Diskussion verschiedener Definitionen beschreibt Mendling (2007) *Geschäftsprozessmodelle*, *Geschäftsprozessmodellierung* und *Geschäftsprozessmodellierungssprachen* wie folgt (Mendling (2007), S. 11-12):

- Ein *Geschäftsprozessmodell* ist das Resultat der Abbildung (siehe Abschnitt 2.4.3.1) eines Geschäftsprozesses. Dieser Geschäftsprozess kann ein realer Prozess eines Unternehmens sein, wie er durch den Modellierer verstanden wird. Alternativ kann es sich um einen konzeptionellen Geschäftsprozess handeln.
- Die *Geschäftsprozessmodellierung* ist die Aktivität der Erstellung eines Geschäftsprozessmodells durch den Modellierer. Die Geschäftsprozessmodellierung beinhaltet die Abstraktion vom realen Geschäftsprozess unter Beachtung eines bestimmten Modellierungszwecks beziehungsweise -ziels. Nur die für den Zweck der Modellierung relevanten Aspekte des Prozesses werden in das Modell übernommen.
- *Geschäftsprozessmodellierungssprachen* bieten ein vordefiniertes Set an Elementen und deren Beziehungen von Geschäftsprozessen zur Nutzung in der Geschäftsprozessmodellierung. Ein solches kontrolliertes Vokabular kann die Modellierung teilweise leiten. Eine Geschäftsprozessmodellierungssprache kann durch ein Meta-Model spezifiziert werden. Zusammen mit einer vordefinierten Vorgehensweise der Modellierung bietet die Modellierungssprache eine Geschäftsprozessmodellierungstechnik.

2.4.3.3 Modellierungssprachen

Im Laufe der Zeit haben sich unterschiedliche Sprachen zur Modellierung von Geschäftsprozessen entwickelt. Der folgende Abschnitt geht auf die vorherrschenden Sprachen in Wissenschaft und Praxis zum gegenwärtigen Zeitpunkt ein. Zurzeit werden von der Praxis vor allem ereignisgesteuerte Prozessketten und Prozessdiagramme der BPMN (Business Process Model and Notation) eingesetzt. Zusätzlich wird vor allem in der Forschung die Darstellung mit Petri-Netzen verwendet.

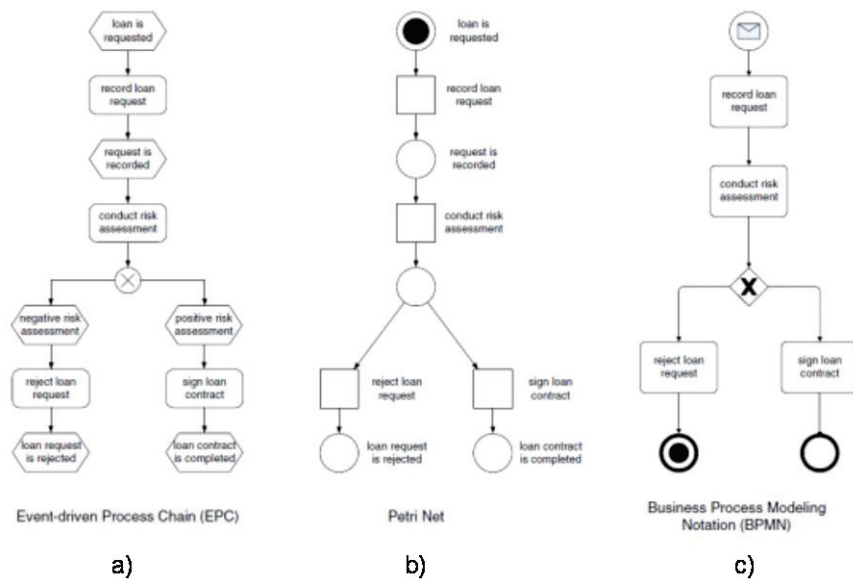


Abbildung 17: Beispiele verschiedener Geschäftsprozessmodellierungssprachen, nach Mendling (2007), S. 9.

Eine vor allem im deutschsprachigen Raum etablierte, aber auch international verbreitete, Geschäftsprozessmodellierungssprache ist die ereignisgesteuerte Prozesskette. Sie wurde von Prof. August-Wilhelm Scheer in Kooperation mit der SAP AG entwickelt (Keller et al. (1992)). Das entstandene „semantische“ Prozessmodell besteht aus Informationsobjekten, Funktionen und Ereignissen (siehe Abbildung 17a, Darstellung ohne Informationsobjekte). Der Ablauf eines Prozesses wird somit definiert von aktiven Funktionen des Informationssystems und passiven Ereignissen, welche den Prozessfluss beeinflussen können (zum Beispiel in Form von betriebswirtschaftlichen Bedingungen, Keller et al. (1992), S. 6-7). Neben den Grundelementen haben sich Erweiterungen zur EPK gebildet. Diese erweiterte EPK (eEPK) kann neben den Grundelementen der Prozessmodellierung Informationen zur Organisation (Einheiten, Stellen, Personen), (IT-) Anwendungen (Systeme, Module, DV-Funktionen), Daten (Geschäftsobjekte, Entitätstypen, Attribute), Informationsträger (Dateien, Listen, Formulare) und Sachmittel (Telefone, Drucker, Ordner, Karteien) abbilden (Krallmann et al. (2007), S. 99 und Seidlmeier (2010), S. 21). An der EPK wird kritisiert, dass sie kein standardisiertes Metamodel besitzt und somit formale Analysen problematisch sind sowie Fehler bei der Modellierung schwer überprüft werden können¹⁵. Allerdings ist die Formalisierung und Verifizierung durch eine Transformation in andere Modelle, zum Beispiel Petri-Netze,

¹⁵ Für eine ausführliche Diskussion von Fehlern in EPK-Modellen sei auf Mendling (2007) verwiesen.

möglich (van der Aalst (1999)). Weiterhin ist ohne formales Meta-Model eine Überführung der Modelle in ausführbare Prozesse nur mit Einschränkungen in der Modellierung beziehungsweise manueller Nacharbeit möglich (Ziemann und Mendling (2005)).

Ein internationales Konsortium von (Software-)Unternehmen, die Business Process Management Initiative (BPML), hat unter der Führung von IBM 2004 die ursprüngliche erste Version der Business Process Model and Notation (BPMN) als grafische Notation der Business Process Modeling Language (BPML) veröffentlicht (Allweyer (2009)). Ursprünglich stand das Akronym noch für Business Process Modeling Notation. Seit 2006 ist die Standardisierungsorganisation Object Management Group (OMG¹⁶) verantwortlich für die Veröffentlichung von BPMN, welche erstmals 2006 in der Version 1.0 verabschiedet wurde. Nach verbesserten Versionen 1.1 (2008) und 1.2 (2009) folgte im Januar 2011 die Version 2.0. Nach Vorschlägen von Konsortien großer internationaler Unternehmen (unter anderem SAP, Software AG, IBM, TIBCO Systems) wurde auf diese Weise eine Modellierungssprache auf Basis eines vordefinierten Metamodells geschaffen. Mittlerweile unterstützen eine große Anzahl von Prozessmodellierungswerkzeugen die BPMN in Version 1.2 oder bereits 2.0. Im Gegensatz zur EPK ist die BPMN vor allem außerhalb des deutschsprachigen Raums weit verbreitet. Das Ziel der BPMN ist die Bereitstellung einer Modellierungssprache, welche sowohl von den meisten Fachverantwortlichen als auch den technischen Entwicklern eines Unternehmens verstanden wird, um auf diese Weise eine Brücke zwischen der Geschäftsprozessmodellierung und der Geschäftsprozessimplementierung zu schlagen. Zusätzlich sollen vor allem XML-basierte Sprachen für die Ausführung von Geschäftsprozessen, wie zum Beispiel die Web Services Business Process Execution Language (WSBPEL), mit BPMN visualisiert werden können (OMG (2011), S. 1). Das endgültige Ziel ist die Standardisierung zur Beschreibung von Geschäftsprozessen über Unternehmens- und Werkzeuggrenzen hinaus. Hierzu definiert die Version 2.0 auch ein entsprechendes standardisiertes Austauschformat. Ein Beispielprozess ist in Abbildung 17c in Form eines Prozessdiagramms der BPMN zu sehen.

In Wissenschaft und Forschung wird zusätzlich zu EPK und BPMN die Repräsentation von Geschäftsprozessen mittels sogenannter Petri-Netze genutzt. Ursprünglich wurde die Repräsentation von Carl Adam Petri zur Darstellung diskreter und verteilter Systeme entwickelt (Petri (1962)). Vor allem die Möglichkeit der Verifikation der Petri-Netze hat zur Verwendung in verschiedenen Disziplinen geführt. Bereits Mitte der 1990er Jahre haben Van der Aalst und Van Hee (1996) vorgeschlagen, Petri-Netze zur Modellierung und Analyse von Geschäftsprozessen zu verwenden. Van der Aalst und Van Hee (1996) haben die ursprünglichen Petri-Netzwerke um die Dimensionen Zeit, Farbe und Hierarchie erweitert, um komplexe Geschäftsprozesszusammenhänge darstellen zu können. Die Farbe steht hierbei für einen Wert eines Token in einem Netz (Van der

¹⁶ www.omg.org

Aalst und Van Hee (1996), S. 19). Bestehende Forschungen haben vor allem die formale Verifikation von Geschäftsprozessen möglich gemacht, wodurch in der Zukunft Softwarehersteller entsprechende Unterstützung in ihren Werkzeugen realisieren können (Wynn et al. (2009)). Ein einfacher Beispielprozess in Form eines Petri-Netzes ist in Abbildung 17 zu sehen.

2.4.3.4 Ausführbarkeit von Geschäftsprozessen

Obwohl die Ausführung von Geschäftsprozessen nicht im Fokus dieser Arbeit liegt, erscheint ein Exkurs in die vorherrschenden Ansätze sinnvoll. Vor allem das Konzept der Weiterverwendung von gefundenen Services durch direkte oder indirekte Transformation der Geschäftsprozessmodelle benötigt diese Grundlagen in der Ausführbarkeit.

Die Ausführbarkeit von Geschäftsprozessen wird bereits seit Beginn der 1990er Jahre durch dedizierte Workflow Execution Software beziehungsweise Workflow Management Systems ermöglicht (Hollingsworth (1993)). Die Prozesse wurden zu Beginn direkt in den unterstützten Programmiersprachen umgesetzt, wodurch flexible Änderungen durch Modellierung nicht möglich waren. Mit dem Beginn der Nutzung von Web Services für die Orchestrierung von Prozessen wurden Initiativen zur Beschreibung und Ausführung durch formale Sprachen gestartet. IBM definierte die Web Service Flow Language (WSFL), welche auf gerichteten Graphen basiert (Leymann (2001)). Microsoft definierte mit dem gleichen Ziel XLANG, welches die WSDL als Fundament verwendet (Thatte (2001)). Aus den konkurrierenden Standards entstand im Zuge des Aufkommens des modernen Geschäftsprozessmanagements (im Gegensatz zum „alten“ Workflowmanagement) im Juli 2002 die Web Service Business Process Execution Language (zunächst BPEL4WS, zurzeit WS-BPEL) (Alves et al. (2007)).

WS-BPEL wurde von der OASIS¹⁷ (Organization for the Advancement of Structured Information Standards) standardisiert und weiter gepflegt. Die Sprache ist blockbasiert und erlaubt die Definition von Prozessen und Interaktion mit externen (Geschäfts-) Partnern mittels sogenannter PartnerLinks. Die PartnerLinks erlauben die Nutzung von definierten (Web-)Service Schnittstellen. (Alves et al. (2007))

Mit der Version 2.0 der BPMN wird explizit die Ausführungssemantik definiert und auf diese Weise die direkte Ausführung von Geschäftsprozessen unterstützt. Weiterhin ist eine Transformation von BPMN Modellen nach WS-BPEL möglich. (Allweyer (2009), S. 13-14)

Als weitere Möglichkeit kann die durch unterschiedliche Universitäten (allen voran die Technische Universität Eindhoven) definierte und gepflegte YAWL¹⁸ (Yet Another Workflow Language) genutzt werden. YAWL basiert auf sogenannten Patterns von

¹⁷ www.oasis-open.org

¹⁸ www.yawlfoundation.org

Workflows und zielt auf die Verbreitung als herstellerunabhängige Sprache ab. (van der Aalst und ter Hofstede (2005))

3 Stand der Forschung

In diesem Kapitel wird der aktuelle Stand der Forschung in relevanten Bereichen des entwickelten Ansatzes dargestellt. Diese Bereiche sind die *Service Discovery*, das *semantische Geschäftsprozessmanagement*, *Ontology Matching*, *Alignment* und *Integration* sowie die *Heterogene Service Discovery*. Somit wird eine Entwicklung anhand der neuesten Erkenntnisse und Implementierungen möglich sowie eine Abgrenzung zu bestehenden Ansätzen.

3.1 Service Discovery

Das Auffinden von Services, im Weiteren mit dem englischen Begriff *Service Discovery* bezeichnet, nimmt eine sehr wichtige Position im Rahmen von serviceorientierten Architekturen ein. Gerade um eine Wiederverwendung von bestehenden Services zu ermöglichen, müssen Service-Beschreibungen existieren, welche im Rahmen der *Service Discovery* mit den Anforderungen des Service-Konsumenten abgeglichen werden können. *Service Discovery* ist jedoch nicht unbedingt gleichzusetzen mit *Web Service Discovery*, wie in der folgenden Betrachtung eines abstrakten *Service Discovery* Prozesses (siehe Abbildung 18) zu sehen ist.

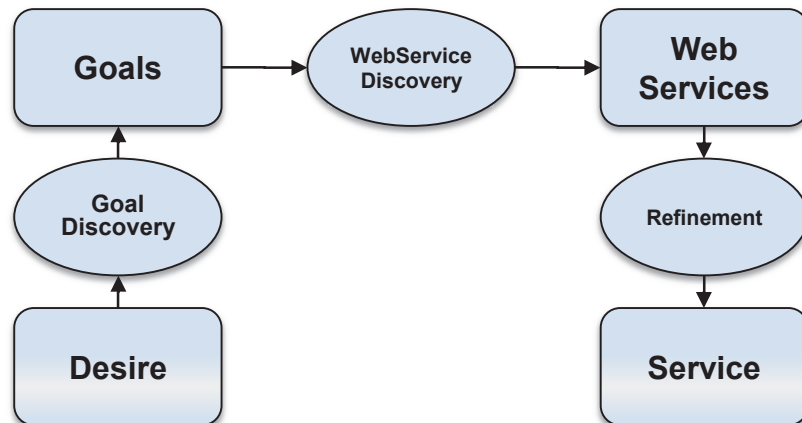


Abbildung 18: Service Discovery Prozess nach Keller et al. (2007), S. 295

Zu Beginn des Prozesses steht der Wunsch einen Service zu finden, welcher bestimmte Bedingungen (zum Beispiel Funktionalitäten) erfüllt. Dieser Wunsch muss allerdings im Folgenden in Form eines Ziels definiert werden, wobei es Unterschiede in den Möglichkeiten und Grenzen einer solchen Zielformulierung geben kann. Vom eigentlichen Wunsch wird abstrahiert und nur bestimmte Charakteristika fließen so in die eigentli-

che Service Discovery ein. Nachdem ein Ziel oder auch mehrere formuliert wurden, können Web Services gesucht werden, welche diese ganz oder teilweise erfüllen. Je nach Art und Umfang der Suchergebnisse kann es nötig sein, die Ergebnisse weiter zu manipulieren (zum Beispiel zu kombinieren), um einen Service zu schaffen, welcher den eigentlichen Wunsch am Anfang des Prozesses erfüllt. (Keller et al. (2007), S. 295)

3.1.1 Ebenen der Service Discovery

Um Services auffindbar zu machen, ist es zudem nötig, bestimmte Grade der Abstraktion einzuführen. Dabei unterscheiden sich diese Grade vor allem in Bezug auf ihre Komplexität und die Genauigkeit ihrer Ergebnisse, wobei sich diese beiden Eigenschaften meist entgegenstehen. Drei Ebenen nach Keller et al. (2007) (S. 289 und 298ff) werden im Folgenden unterschieden:

3.1.1.1 Keyword Discovery

Das Auffinden von Services auf Basis bestimmter Schlüsselwörter beziehungsweise durch die Analyse von Fließtextbeschreibungen der Services muss vor allem mit den Ambiguitäten der natürlichen Sprache umgehen können. Dies wirkt sich nicht nur auf die Formulierung der Suchbegriffe aus, sondern beeinflusst bereits den Prozess der Service-Annotation während der Design-Phase. So können bestimmte Namensvorgaben, kryptische Bezeichner oder maschinengenerierter Code zu Problemen bei der textuellen Suche führen. Dennoch ist diese Suche wichtig und beinhaltet meist keinen bis einen geringen Mehraufwand während des Servicedesign.

3.1.1.2 „Leichtgewichtige“ Semantische Discovery

Die Suche nach Services auf Basis von Schlüsselwörtern ist im Allgemeinen nicht formalisiert und bietet daher auch keine Möglichkeit, Schlussfolgerungen zu ziehen. Eine Lösung dies zu ermöglichen ist der Einsatz von kontrolliertem Vokabular, welches explizite und formale Semantik beinhaltet. Solch ein Vokabular ist beispielsweise mit einer Ontologie (und einer entsprechenden formalen Ontologiesprache) realisierbar. Unter diesen Voraussetzungen ist es möglich, sowohl die Funktionalitäten der Services als auch die Anforderungen (in Form von Zielen) zu formalisieren. Eine solche Beschreibung erlaubt es, den Zusammenhang zwischen Ziel und Service in Form unterschiedlicher Treffer zu beschreiben: Exakter, partieller, möglicher partieller und kein Treffer. Weiterhin können diese Informationen verwendet werden, um Kriterien zu formulieren, welche den Umfang der Funktionalität in Abhängigkeit von den Anforderungen beschreiben: *Intersection* (Überschneidung), *Subsume* (Subsumption), *Plugin* und *Identity* (Gleichheit).

In einem „leichtgewichtigen“ Ansatz sind die Services jedoch als atomare Objekte beschrieben, wodurch aus realen dynamischen Services abstrakte statische Servicebeschreibungen entstehen. Dabei bleibt allerdings meist offen, unter welchen Bedingungen die Services einsetzbar sind.

3.1.1.3 „Schwergewichtige“ Semantische Discovery

Diese Art von Discovery hat zum Ziel, ein hohes Maß an Automatisierung zu ermöglichen. Dabei setzt dieser Ansatz auf der im vorigen Abschnitt beschriebenen „leichtgewichtigen“ Variante auf und erweitert diesen um die Beschreibung der Funktionalität in Form von dynamischen Bedingungen. Auch dieser Ansatz benötigt eine formale Beschreibungssprache. Die Erweiterung der Beschreibung von dynamischen Aspekten kommt meist in Form von Abhängigkeiten der Ein- und Ausgaben des Service. Folglich müssen Service Beschreibungen sowohl Ein- und Ausgabeparameter enthalten als auch Vorbedingungen und Effekte.

Lösungen mit einem hohen Maß an semantischer Information stellen jedoch auch immer wieder sowohl seitens der Service Entwickler als auch seitens der Service Konsumenten entsprechend Ansprüche an den Benutzer.

3.1.2 Bestehende Ansätze

Bestehende Ansätze zur Discovery können vor allem durch die Nutzung der jeweiligen Beschreibungssprache unterschieden werden, welche unterschiedliche Vor- und Nachteile mit sich bringen und somit auch Auswirkungen auf die Art der Discovery haben. Im Folgenden werden daher Ansätze der vorherrschenden Sprachen zur Beschreibung von semantischen Web Services (siehe Abschnitt 2.2.3) diskutiert: OWL-S, WSMO und SAWSDL. Die Analyse der Ansätze untersucht sowohl das konzeptionelle Modell der Service Discovery (und somit meist auch der gewählten Servicebeschreibung) als auch die technische Implementierung.

3.1.2.1 Discovery mit OWL-S

Es existiert eine Vielzahl von Ansätzen zur Discovery von Services, welche mit Hilfe von OWL-S beschrieben werden. Dieser Abschnitt gibt eine Übersicht über den Stand der Forschung und stellt ausgewählte Ansätze vor. Es ist zu beachten, dass diese anfänglich auf diesem Gebiet auf dem Vorgänger von OWL-S, DAML-S, beruhen.

Hybride Service Discovery (Matchmaking) mit OWLS-MX

Am Deutschen Forschungszentrum für Künstliche Intelligenz in Saarbrücken wurde in Kooperation mit der Carnegie Mellon University, USA ein hybrider Matchmaker für OWL-S Services entwickelt. Konzept, Implementierung und Testergebnisse wurden in verschiedenen Artikeln publiziert. (Klusch et al. (2005); Kaufer und Klusch (2006a); Kaufer und Klusch (2006b); Klusch et al. (2009)).

OWLS-MX kombiniert das Matching von Services basierend auf logischen Sprachen - wie Prädikatenlogik (der ersten Stufe, *engl.* First Order Logic (FOL)), Regelsprachen oder logische Programmierung - mit nicht-logischem Matching. Hybrides Matching kann eingesetzt werden, wenn Logik-basiertes Matching zu keinen oder schlechten Ergebnissen führt. Der Einsatz eines solchen hybriden Matchings kann auch zur besseren

Skalierbarkeit beitragen, wenn zum Beispiel FOL-Reasoning nicht mehr effizient möglich ist (Fensel und van Harmelen (2007)).

Die Konzepte zur Servicebeschreibung können entweder in OWL-Lite oder OWL-DL formuliert werden. Zum Matching werden jedoch nur die Inputs und Outputs (I/O) des Serviceprofils verwendet (siehe Abschnitt 2.2.3.1). Der Matching Mechanismus von OWLS-MX basiert dafür auf fünf verschiedenen Filtern (oder auch Trefferkategorien): *exact*, *plugin*, *subsumes*, *subsumed-by* und *nearest neighbor*. Die ersten drei der Filter sind Logik-basiert, die letzten können als Hybrid bezeichnet werden, da sie zusätzlich syntaktische Ähnlichkeit berechnen (Klusch et al. (2006), S. 916). Ein *exact* Treffer entsteht, wenn die I/O Signatur des Service logisch exakt zur Anfrage passt. Falls der Service weniger Input benötigt und einen „detaillierteren“ Output erzeugt (welcher logisch exakt oder fast exakt die Anfrage trifft) als durch die Anfrage spezifiziert, entsteht ein *plugin* Treffer. Ein *plugin* Treffer kann gelockert werden, indem für das Output keine direkt Subsumption unterstellt wird. So entsteht ein *subsumes* Treffer. Ein *subsumed-by* Treffer liegt vor, wenn der Output des Service eine direkte Generalisierung des Outputs der Anfrage ist¹⁹. Weiterhin wird bei diesem Treffer auch ein syntaktisches Matching eingesetzt, welches einen gewissen Grenzwert überschreiten muss, um als Treffer gewertet werden zu können. Falls diese Logik-basierten Filter fehlschlagen (*logic-based fail*) kann mit Hilfe des syntaktischen Matchings noch ein *nearest neighbor* Treffer erzeugt werden, falls ein vordefinierter Wert α überschritten werden kann. Schlagen die Filter alle fehl entsteht kein Treffer (*fail*). Zum besseren Verständnis sind *plugin* und *nearest neighbor* Treffer zusammen mit der zugehörigen Domänenontologie beispielhaft in Abbildung 19 dargestellt. (Klusch et al. (2009), S. 5-7)

¹⁹ Anders formuliert: Output des Service subsumiert Output der Anfrage.

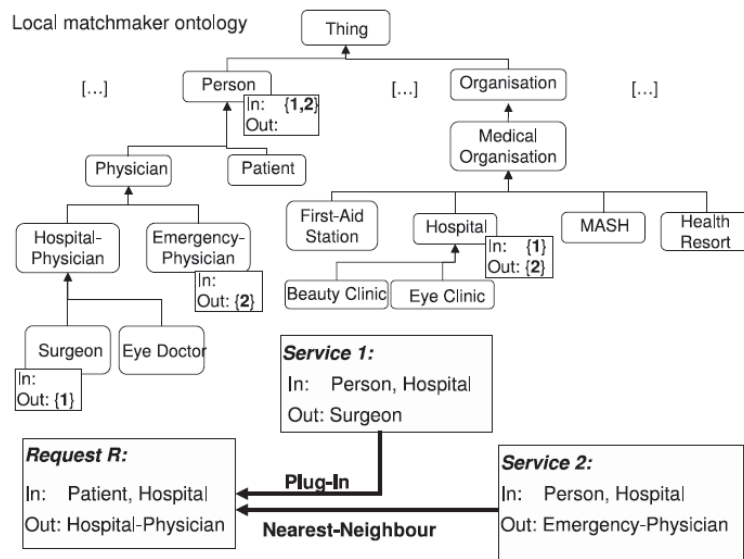


Abbildung 19: Beispiel von Treffern des Werkzeuges OWLS-MX, nach Klusch et al. (2009), S. 131.

Der Algorithmus von OWLS-MX verwendet als Anfrage eine OWL-S Servicebeschreibung. Das Ergebnis ist eine Liste aller relevanten Services, welche mit einem Treffergrad und einem Wert der syntaktischen Ähnlichkeit annotiert sind. Gewünschter Treffergrad und Grenzwert für die syntaktische Ähnlichkeit können vom Anwender definiert werden. Zur Suche nach passenden Services können verschiedene Varianten des Algorithmus eingesetzt werden: M0-M4. Variante M0 verwendet nur den Logik-basierten Matcher und liefert somit *exact*, *plugin* und *subsumes* Treffer sowie *subsumed-by* Treffer ohne syntaktische Ähnlichkeitsmessung. M1-M4 messen zusätzlich die syntaktische Ähnlichkeit, unterscheiden sich aber durch das verwendete Maß zur Ähnlichkeitsbestimmung (zum Beispiel Cosinus-Maß oder Jaccard-Koeffizient²⁰). (Klusch et al. (2006), S. 918-919)

Der OWLS-MX Algorithmus wurde in Java implementiert und verwendet die OWL-S API 1.1²¹, das Jena Semantic Web Framework²² und den OWL-DL Reasoner Pellet²³ (Klusch et al. (2009), S. 131). Die Suchanfragen und Ergebnisse werden über eine grafische Benutzeroberfläche dargestellt (zu sehen in Abbildung 20).

²⁰ Anzahl der gemeinsamen Mengen geteilt durch die Vereinigungsmenge.

²¹ www.mindswap.org/2004/owl-s/api/

²² jena.sourceforge.net

²³ clarkparsia.com/pellet

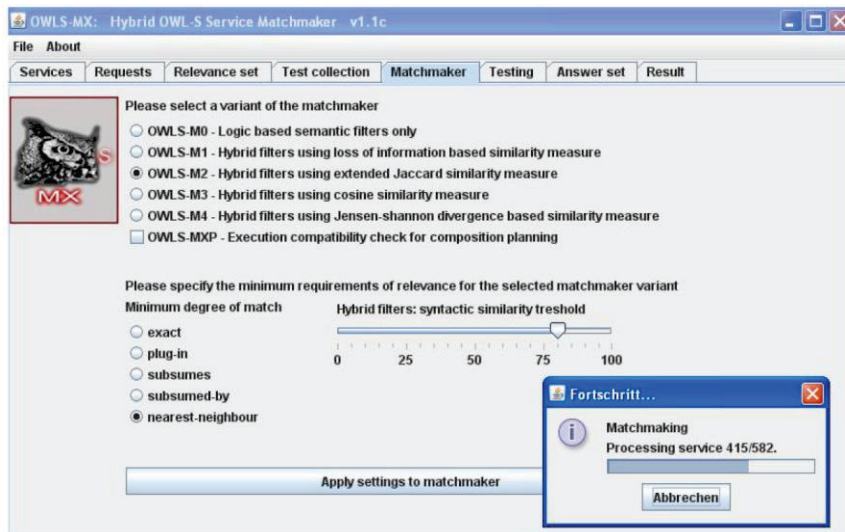


Abbildung 20: Benutzeroberfläche von OWLS-MX 1.1c, nach Klusch et al. (2009), S. 132.

Die Leistungsfähigkeit der Implementierung wird vor allem eingeschränkt durch den hohen Rechenaufwand zur initialen Befüllung der Matching-Ontologie. Die eigentlichen Anfragen auf der Ontologie benötigen nur noch einen Bruchteil dieser Zeit. Zur Evaluierung wurde eine Sammlung an Testservices und –anfragen²⁴ erzeugt. Unter Verwendung der Testanfragen wurden *Precision* und *Recall* gemessen. Durch die Ergebnisse konnten die Autoren zeigen, dass die Verwendung der zusätzlichen syntaktischen Ähnlichkeitsmessung im Durchschnitt sowohl *Precision* als auch *Recall* erhöhen konnte. (Klusch und Fries (2009), S. 51)

Letztlich wird auf mögliche Probleme bei der Nutzung hingewiesen. So ist zu beachten, dass vor allem falsche positive Ergebnisse (*engl.* false positives) bei der Suche entstehen können, wobei Services zur Nutzung vorgeschlagen werden, welche die ursprünglichen Anforderungen der Anfrage nicht erfüllen können. Solche Ergebnisse treten vermehrt bei der Logik-basierten Suche auf, zum Beispiel bei *plugin* Treffern, bei denen die Distanz zwischen gesuchtem und gefundenem Input-Konzept zu hoch ist, um noch als relevant erachtet zu werden. Hier spielt auch die Granularität der verwendeten Domänenontologie(n) eine Rolle. Weitere Probleme ergeben sich über die Verwendung von nur Input und Output Parametern, welche bei semantisch unterschiedlichen Services durchaus gleich sein können. (Klusch und Fries (2009), S. 52ff)

²⁴ OWL-S Testcollection (TC) 2.2 (Aktuelle Version 4.0), verfügbar unter projects.semwebcentral.org/projects/owls-tc/

DAML-S Discovery mittels UDDI (Universal Description, Discovery, and Integration)

Sycara et al. (2003) entwickelten einen Ansatz zur Suche, Interaktion und Komposition von semantischen Services auf Basis von UDDI und DAML-S. Dieser Ansatz soll im Folgenden vorgestellt werden, da er einer der prominentesten und ersten im Bereich der Web Service Discovery darstellt.

Der Ansatz kombiniert das von der OASIS standardisierte Register für Web Services – UDDI – und die Funktionalitätsbeschreibungen in Form von DAML-S. Wie auch bei Käufer und Klusch (2006b) werden Input und Output-Attribute des DAML-S *service profile* verwendet, um die Services und Serviceanfragen zu beschreiben. Zur Ermittlung von Treffern wird ein rein auf Logik (*Description Logic*) basierter Ansatz verwendet der auf der Arbeit der Autoren in Paolucci et al. (2002) aufbaut. Ein Treffer wird geliefert, falls ein angebotener Service an Stelle des gesuchten Service eingesetzt werden kann. Dies ist formal der Fall, wenn die Ausgabe des angebotenen Service gleich oder allgemeiner als die Ausgabe der Anfrage und gleichzeitig die Eingabe der Anfrage gleich oder allgemeiner als die Eingabe des Angebots ist. Diese Formalisierung erlaubt folgende Treffer: *exact output match* (Output der Anfrage $out_{Req} = \text{Output des angebotenen Service } out_{Ad}$), *exact input match* (Input der Anfrage $in_{Req} = \text{Input des angebotenen Service } in_{Ad}$), *plugIn match* ($out_{Ad} > out_{Req}$ oder $in_{Req} > in_{Ad}$) und *subsumed match* ($out_{Ad} < out_{Req}$ oder $in_{Req} < in_{Ad}$). Letzterer wird verwendet obwohl, unter Umständen weitere Daten für das Input zur Verfügung gestellt werden müssen und/ oder nicht alle gewünschten Daten vom angebotenen Service geliefert werden können. Laut Sycara et al. (2003) ist ein *plugIn* Treffer relevant, da andere Services genutzt werden können, um fehlende Daten zu liefern. Wird kein Treffer geliefert, so entsteht ein *fail*. Der Algorithmus ordnet die Treffer nach *exact* > *plugIn* > *subsumed* > *fail*, unter der Prämisse, dass Output Treffer eine höhere Relevanz als Input Treffer haben. Dahinter steht die Annahme, dass der Benutzer besser weiß was er von einem gesuchten Service als Output erwartet, als dass er genau beschreiben kann, was vom Anbieter an Input verlangt wird. (Sycara et al. (2003), S. 36)

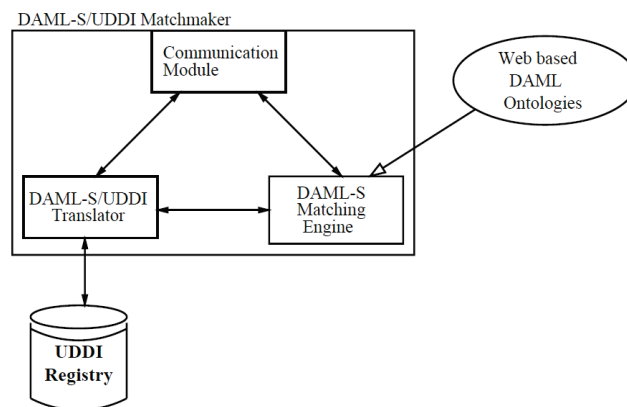


Abbildung 21: Architektur des DAML-S/UDDI Matchmaker nach Sycara et al. (2003), S. 38.

Die Kombination mit dem UDDI Register wird durch die Nutzung des sogenannten TModels²⁵ ermöglicht. Die UDDI Spezifikation sowie die meisten Implementierungen der Spezifikation sehen nur eine Schlüsselwort-basierte Suche vor. Daher ist es nötig die Beschreibungen der Funktionalität (*engl.* Capability) von DAML-S nach UDDI zur übersetzen. Allerdings ist es durch die fehlende Suchfunktionalität von UDDI notwendig, eine neue Schnittstelle für Suchanfragen zu schaffen. Eine Architektur zur Integration der einzelnen Module ist in Abbildung 21 zu sehen. Das Kommunikationsmodul empfängt sowohl Anfragen als auch Angebote von Services. Angebote werden in eine UDDI Beschreibung übersetzt, erhalten eine eindeutige Identifikation und die Beschreibung ihrer Funktionalität in DAML-S wird in der Matching Engine abgelegt. Anfragen werden durch das Kommunikationsmodul weitergeleitet an die Matching Engine, deren Ergebnis verwendet wird, um die UDDI Beschreibung des Service abzurufen. (Sycara et al. (2003), S. 36-37)

Die Skalierbarkeit des DAML-S/UDDI Matchmakers wurde von den Autoren nicht untersucht, sondern lediglich das Antwortverhalten bei einzelnen Anfragen gemessen. Eine Sammlung von Testservices und -anfragen kam nicht zum Einsatz, so kann auch keine Aussage zu *Precision* und *Recall* der Lösung gemacht werden.

Weitere Discovery-Ansätze mit OWL-S

Li und Horrocks (2004) präsentieren einen ähnlichen Ansatz des Subsumptions-Reasoning auf der DAML-S Beschreibungssprache. Zur Repräsentation von Regeln und Wissen wird auch auf die Beschreibungslogik zurückgegriffen. Die Autoren weisen auf das Problem hin, dass in einem DAML-S *profile* sehr viele Informationen enthalten sind, so viele, dass ein einzelner Ausdruck die Berechnungen semantischer Übereinstimmungen schwierig macht (Li und Horrocks (2004), S. 54). Um dieses Problem zu lösen, werden bestimmte Eigenschaften aus dem Profil extrahiert und direkt als Attribute an Serviceangebote und -anfragen beschrieben. Somit werden neben dem Eingangs- und Ausgangswert noch Anbieter und Nachfrager beschrieben. Zusätzlich zu den Treffern *Exact*, *PlugIn* und *Subsume* wird eine zusätzliche Trefferkategorie, *Intersection*, durch die Autoren hinzugefügt. Formal drückt diese jedoch nur aus, dass die Schnittmenge von Anfrage und Angebot erfüllbar ist. Hierbei wird von Bechhofer et al. (2004) explizit darauf hingewiesen, dass diese Unterscheidung von Paolucci et al. (2002) nicht gemacht wurde. Falls die genannten Treffer nicht gefunden werden, wird automatisch ein *Disjoint* generiert. Die Priorisierung der Treffer ist *Disjoint* < *Intersection* < *Subsume* < *PlugIn* < *Exact*, ohne eine Unterscheidung zwischen Eingangs- und Ausgangswerten. Die Implementierung des Matchmaking Algorithmus basiert auf einer Umgebung mit verschiedenen Software-Agenten. Die Schlussfolgerungen werden mittels RACER²⁶

²⁵ Technical Model, technische Beschreibung eines Service in UDDI mit eindeutiger Identifizierung.

²⁶ www.sts.tu-harburg.de/~r.f.moeller/racer/

(Haarslev und Müller (2001)) berechnet. Die Auswertung der Leistungsfähigkeit des Ansatzes zeigt vor allem, dass die Klassifizierung der Angebote durch RACER schlecht skaliert. (Li und Horrocks (2004))

Die vorgestellten Ansätze betrachten noch keine Nutzung der semantischen Beschreibung zur Komposition von Services zur Erfüllung eines Ziels. Erste Anstrengungen in diese Richtung unternahmen Benatallah et al. (2003), indem in einem Algorithmus untersucht wird, wie die Anfrage unter Berücksichtigung der bestehenden Services „umgeschrieben“ werden kann. Hiermit kann über den entstehenden Kandidaten die beste Variante ausgewählt werden. Erreicht werden soll dieses Ziel, indem eine Anfrage durch eine Kombination von Services beantwortet wird, wobei die größte Möglichkeit an Ausgangswerten der Anfrage geliefert und gleichzeitig die geringste Menge an Eingangswerten benötigt wird. Bedingt durch DAML-S wird auch in diesem Ansatz die Beschreibungslogik verwendet, um Schlussfolgerungen zu erlauben. Zur Berechnung des besten Sets an Services (von den Autoren *Profil* genannt) werden Hypergraphen verwendet (siehe hierzu Berge (1976)). Die Implementierung des Algorithmus wurde mit zufälligen Servicebeschreibungen und Ontologien getestet, bei denen allerdings eine maximale Anzahl von 660 Services getestet wurde. In den Tests konnten Antwortzeiten von 50-60 ms gezeigt werden. (Benatallah et al. (2003))

Ein weitreichender Vergleich dieser und anderer Ansätze findet sich in Lara et al. (2005). Es existieren auch aktuelle Ansätze, welche in diesem Vergleich noch nicht berücksichtigt wurden. Zu erwähnen ist hier vor allem die Verbindung von OWL-S Profilen mit UDDI-Verzeichnissen von Aguilera et al. (2007). Es sei anzumerken, dass hierbei keine Evaluierung der Komplexität durchgeführt wurde und zu erwarten ist, dass aufgrund der beschriebenen Rahmenbedingungen hohe Aufwände in der Berechnung entstehen.

3.1.2.2 Discovery mit WSDL-S/SAWSDL

Es existiert nur eine geringe Anzahl von Ansätzen zur Discovery von Services, welche mit Hilfe von SA-WSDL beschrieben werden. Dieser Abschnitt gibt eine Übersicht über den Stand der Forschung und stellt ausgewählte Ansätze vor. Es ist zu beachten, dass die anfänglichen Ansätze auf diesem Gebiet auf dem Vorgänger von SA-WSDL, WSDL-S, beruhen.

Lumina Implementierung in der METEOR-S Discovery and Publish Engine

Der prominenteste Ansatz in der Nutzung von WSDL-S und SAWSDL ist das METEOR-S Projekt²⁷. Die METEOR-S Web Services Discovery Infrastructure (MWSDI) realisiert die Suche nach Web Services basierend auf WSDL-S Beschreibungen mittels UDDI. Im Vergleich mit anderen Ansätzen verwendet MWSDI einen

²⁷ ldis.cs.uga.edu/projects/meteor-s/

„leichtgewichtigen“ Ansatz zur Service Discovery (siehe Abschnitt 3.1.1.2). Bestehende Technologien, in diesem Fall WSDL und UDDI, werden mit semantischen Informationen angereichert. Es wird somit kein zusätzlicher Aufwand in Form eines neuen Beschreibungsansatzes geschaffen. (Grimm (2007))

Zur Annotation von Services in der MWSDI werden die XML Bausteine von WSDL auf domänenspezifische Ontologien abgebildet, um somit die vom Service Anbieter unterstellte Bedeutung zu repräsentieren. Diese Verbindung kann im Weiteren zur Suche nach Services verwendet werden. Annotiert werden im METEOR-S Ansatz nur Input- und Outputelemente von WSDL. Dabei kann es mehrere In- und Outputs geben, welche jeweils eindeutig mit einem Konzept aus der Domänenontologie verbunden werden. Für die Verknüpfung zwischen WSDL Konzepten und Ontologie werden zwei Elemente von UDDI verwendet: *tModels* und *CategoryBags* (Sivashanmugam et al. (2003)). *tModels* können verwendet werden, um Entitäten in UDDI mit Konzepten von Taxonomien zu verknüpfen. Bei der Registrierung eines *tModel* in UDDI wird gleichzeitig eine eindeutige Identifikation generiert, auf welche verwiesen werden kann. Verschiedene *tModels* können mit Hilfe der *CategoryBags* kategorisiert werden. Zur Publikation von semantischen annotierten Services werden *tModels* für die Taxonomien von In- und Outputkonzepten erstellt. Diese können mittels des *overviewURL* Konzeptes des *tModels* mit der entsprechenden Domänenontologie verbunden werden. Die Kategorisierung von Services erfolgt danach durch die Verknüpfung von entsprechenden Input- oder Outputkonzepten mit der ID der verwendeten Ontologie. Diese Paare (Konzept, ID) können in UDDI mittels des *tModels* abgelegt und zur späteren Suche verwendet werden. (Verma et al. (2005), S. 31-33)

Die Suche nach Services beginnt entweder mit der bereits integrierten Suche in UDDI oder durch die Definition von Service Vorlagen (*engl.* Templates), welche die gewünschten ontologischen Konzepte bezüglich In- und Output des Service beschreiben. Der implementierte Algorithmus überprüft zunächst die Funktionalität der registrierten Web Services mit der Anfrage (Phase 1). Anschließend werden die Resultate der ersten Phase anhand ihrer semantischen Ähnlichkeit von In- und Output klassifiziert (Phase 2). Optional kann in der dritten Phase eine Überprüfung von Vorbedingungen und Effekten durchgeführt werden²⁸. (Sivashanmugam et al. (2003), S. 397)

²⁸ Diese Phase ist in der theoretischen Ausarbeitung vorhanden, findet sich aber in der Beschreibung der Implementierung des MWSDI nicht wieder.

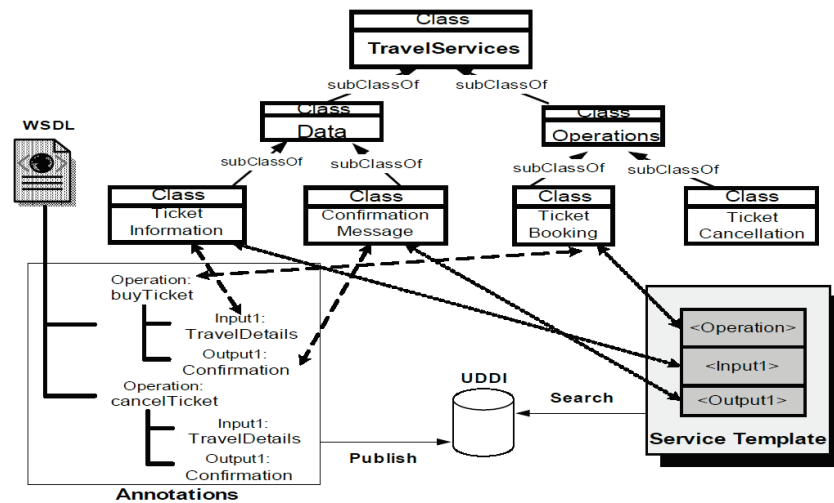


Abbildung 22: Semantische Annotation, Publikation und Discovery, nach Sivashanmugam et al. (2003), S. 401.

Der bestehende Ansatz von MWSDI wurde mit der Veröffentlichung von SAWSDL erweitert²⁹. Die Suche nach Services in Form von SAWSDL Beschreibungen wird durch ein Eclipse-Plugin (genannt *Lumina*³⁰) ermöglicht, welches als GUI für die MWSDI Implementierung verwendet wird. Weiterhin wird ein Werkzeug zur Erweiterung von WSDL Beschreibungen mit SAWSDL (und WSDL-S) Annotationen bereitgestellt (*Radian*³¹). Eine detaillierte Beschreibung der Implementierung von *Lumina* findet sich in Li (2005).

Die starke Kopplung an WSDL macht diesen Ansatz nicht ganz so flexibel wie andere, bietet dennoch eine hohe Standardkonformität und führt keine zusätzliche Schicht für die folgende Ausführung der Services ein. Eine Evaluierung der Qualität und Leistungsfähigkeit der Implementierung konnte nicht gefunden werden.

Semantische Service Auswahl mit SAWSDL-MX

Wie auch die Implementierung OWL-MX (siehe Abschnitt 3.1.2.1) basiert der SAWSDL-MX Algorithmus auf der Kombination von Logik-basierten und IR³²-basiertem Matching. Von Klusch und Kapahnke (2008) wird hier auf einen Nachteil der Ontologieagnostik von SAWSDL hingewiesen. Die Auswahl von Services muss sowohl mit der Heterogenität von Ontologiesprachen als auch mit der von Ontologien an

²⁹ lstdis.cs.uga.edu/projects/meteor-s/SAWSDL/

³⁰ lstdis.cs.uga.edu/projects/meteor-s/downloads/Lumina/

³¹ lstdis.cs.uga.edu/projects/meteor-s/downloads/OnlineDoc/

³² Information Retrieval, siehe auch Abschnitt 3.1

sich umgehen. Die Autoren umgehen diese Problematik zunächst mit der Adoption nur einer Ontologiesprache, nämlich OWL-S. Auf diese Weise ist, wie bei OWL-S MX, ein Subsumptions-Reasoning möglich. Weiterhin kann man mit SAWSDL verschiedene Ebenen der XML-Struktur annotieren. In der Spezifikation existiert keine Vorgabe, welche Ebene der Granularität zu wählen ist oder wie verschiedene Ebenen der Annotation zu kombinieren sind. Klusch und Kapahnke (2008) annotieren daher in der ersten Version von SAWSDL-MX nur die oberste Ebene der Daten einer Serviceoperation. (Klusch und Kapahnke (2008), S. 5-7)

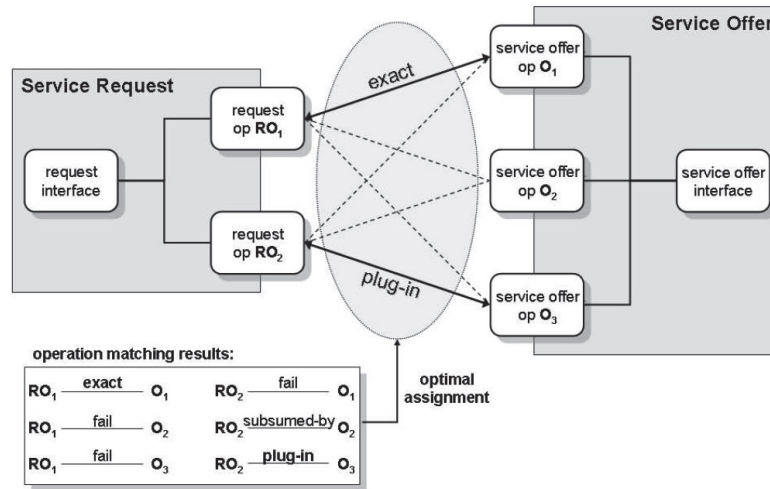


Abbildung 23: Schnittstellenbasiertes Matching mit SAWSDL-MX.

Das Matching des SASWDL-MX wird auf Schnittstellenebene (WSDL Interface) durchgeführt. Als Anfrage werden SAWSDL Dateien verwendet, welche genau eine Schnittstelle aber mehrere Operationen beinhalten können. Jede mögliche Kombination der Operationen wird im Rahmen des Algorithmus mit den Serviceangeboten verglichen. Dies geschieht unter Nutzung eines bipartiten Graphen, welcher als Knoten die Operationen von Anfrage und Angebot besitzt. Die Kanten zwischen den Knoten werden mit dem Grad der Ähnlichkeit beziehungsweise dem Grad des Treffers gewichtet (siehe Abbildung 23). Hierbei ist zu beachten, dass die Suche nur auf Angeboten mit einer Operation pro Schnittstelle beziehungsweise Service basiert. Die Komplexität mehrerer Operationen pro Service würde den Einsatz weiterer Planungsalgorithmen erfordern, welche in der aktuellen Version von SAWSDL-MX nicht vorhanden sind. Logik-basiertes und syntaktisches Matching der Operationen wird mit den gleichen Algorithmen wie in OWLS-MX (siehe Seite 53) durchgeführt. Eine Evaluierung der Implementierung erfolgt durch die Verwendung der OWL-S Testcollection, welche halbautomatisch in SAWSDL Definitionen überführt wurde (unter Verwendung von

OWLS2WSDL³³). Wie auch bereits bei OWLS-MX legen die Testergebnisse nahe, dass ein hybrides Matching bessere Ergebnisse als reines Logik-basiertes Matching liefert. Allerdings liegen die Antwortzeiten des hybriden Matchings sowohl über reinem Logik-basierten als auch IR-basiertem Matching. (Klusch und Kapahnke (2008), S. 7-9 und S. 13-15)

Kombination von SAWSDL, OWL-DL und UDDI in FUSION

Kourtesis und Paraskakis (2008) stellen einen Ansatz zur Kombination verschiedener Techniken der Service Discovery mittels SAWSDL vor. Hierzu wird UDDI zum Speichern und Abrufen von syntaktischen und semantischen Informationen, SAWSDL als Annotationsmechanismus für Serviceschnittstellen und OWL-S zur Modellierung von Serviceeigenschaften sowie zur Nutzung von DL-Reasoning verwendet. Services werden in FUSION durch funktionale Serviceprofile (*engl.* functional profile) repräsentiert. Diese beinhalten auch nichtfunktionale Eigenschaften. Dargestellt als OWL-Objekt, beinhaltet das *Functional Profile* eine Kategorisierung des Service in einer bereitgestellten Servicetaxonomie, eine Beschreibung des Inputs (in Form eines Sets an möglichen Input Daten, *InputDataSet*) und eine Beschreibung des Outputs (analog zur Inputbeschreibung). Die Suche nach Services kann sowohl über Schlüsselwörter als auch unter Verwendung eines *Request Functional Profile* (RFP) angestoßen werden. Basierend auf Subsumptions-Reasoning können relevante Services identifiziert und ausgegeben werden. Welche Arten von Treffern existieren (wie zum Beispiel bei Klusch und Kapahnke (2008)) wird durch die Autoren nicht erläutert. Die prototypische Implementierung von FUSION basiert auf UDDI4j³⁴, SAWSDL4j³⁵ / WSDL4j³⁶, Pellet und der OWL-API³⁷. (Kourtesis und Paraskakis (2008))

Semantische Service Discovery mit SAWSDL und SPARQL

Einen weiteren Ansatz zur Nutzung von SAWSDL für die Suche nach Services präsentieren Iqbal et al. (2008). Der Beitrag der Autoren besteht dabei vor allem in der Erweiterung der SAWSDL Beschreibungen um Vor- und Nachbedingungen in Form von SPARQL-Ausdrücken, einer Abfragesprache für RDF (Prud'hommeaux und Seaborne (2008)). Somit kann dem Umstand Rechnung getragen werden, dass semantische Servicebeschreibungen basierend auf In- und Outputdaten unter Umständen nicht ausreichen, um die Funktionalität des Service eindeutig zu beschreiben. Unter Verwendung der SAWSDL Modellreferenzen (*engl.* model references, siehe Abbildung 11) können die Serviceoperationen einer WSDL Definition mit Vor- und Nachbedingungen assozi-

³³ projects.semwebcentral.org/projects/owls2wsdl/

³⁴ uddi4j.sourceforge.net

³⁵ knoesis.wright.edu/research/srl/oss/sawSDL4j/

³⁶ sourceforge.net/projects/wsdl4j/

³⁷ owlapi.sourceforge.net/

iert werden. Die Servicesuche verwendet durch den Nutzer definierte Ziele in Form von SPARQL Anfragen und vergleicht diese mit den Beschreibungen der Serviceangebote. Falls keine Treffer durch diese Suche erzeugt werden, können nach und nach Vorbedingungen entfernt werden. Ob diese Möglichkeit auch für Nachbedingungen gilt, wird von den Autoren nicht erläutert. Der vorgestellte Ansatz wurde prototypisch auf Basis des Jena Semantic Web Toolkit³⁸, der ebXML Registry freebXML³⁹ (zur Speicherung der SAWSDL und SPARQL Beschreibungen) und der JAXR API⁴⁰ implementiert. Eine Evaluierung der Implementierung ist nicht publiziert. (Iqbal et al. (2008))

3.1.2.3 Discovery mit WSMO

Durch die hohe Spezifizierung der einzelnen Elemente von WSMO wird bereits teilweise vorgegeben, wie eine Discovery durchzuführen ist. Anfragen werden konform zu WSMO *Goals* formuliert, welche neben Pre- und Postcondition, Assumptions und Effects, sowie In- und Outputs auch nichtfunktionale Eigenschaften und interne Prozessabläufe der Services beinhalten können. Bestehende Ansätze decken jedoch bei weitem nicht alle diese Aspekte ab, sondern konzentrieren sich meist auf Subsumptions-Reasoning von Pre- und Postconditions (Keller et al. (2007)). Exemplarisch werden im Folgenden WSMO-MX und die Discovery im WSMX (Web Service Modeling eXecution environment) vorgestellt.

Hybride Service Discovery (Matchmaking) mit WSMO-MX

Wie auch die bereits vorgestellten Ansätze aus der *-MX Matchmaker Familie (OWL-S siehe Seite 61 und SAWSDL siehe Seite 53) zielt der WSMO-MX Ansatz auf die Kombination aus Logik-basiertem Matching und syntaktischer Ähnlichkeitsbestimmung. Im Gegensatz zu den bereits vorgestellten Ansätzen verwendet WSMO-MX allerdings eine Beschreibung der Services, welche nicht auf Beschreibungslogik basieren. Dazu wird der WSML-Rule Dialekt (siehe Abschnitt 2.2.2.1) genutzt. (Klusch und Kaufer (2009), S. 23)

Die Beschreibung der Services in WSMO-MX basiert neben den Eingangs- und Ausgangsparametern (wie bereits in den vorherigen *-MX Ansätzen) zusätzlich auf der Definition von Vorbedingungen und Effekten. Dies erlaubt eine genauere Beschreibung der Funktionalität, ist aber natürlich auch mit einem höheren Aufwand verbunden. Hierbei kommt als Basis WSML-Rule zum Einsatz, welche auf der sogenannten Frame Logic (F-Logic) basiert. F-Logic bietet eine Erweiterung der Prädikatenlogik erster Ordnung um Konzepte der Objektorientierung (Kifer und Lausen (1989)). Es erlaubt die logische Programmierung mittels Objektorientierung unter Nutzung komplexer Objekte, Hierarchien und Vererbung. (Klusch und Kaufer (2009), S. 24-25)

³⁸ jena.sourceforge.net/

³⁹ www.freebxml.org/

⁴⁰ java.sun.com/webservices/jaxr/

Kaufer und Klusch (2006b) erweitern WSMO-Rule zur Beschreibung der Services um die Möglichkeit, bestimmte Präferenzen und Einschränkungen oder Lockerungen bei der Suche modellieren zu können. Die angepasste WSMO-MX verwendet zusätzliche Derivat-Konzepte (*engl. derivatives*) zur Annotation von Meta-Informationen über Art und Weise, wie ein Matching der Konzepte durchgeführt werden kann. Somit können bestehende Konzepte einer Ontologie gekapselt und mit diesen Informationen erweitert werden. Der Aufbau eines Service-Derivats ist in Abbildung 24 zu sehen. (Klusch und Kaufer (2009), S. 25-26)

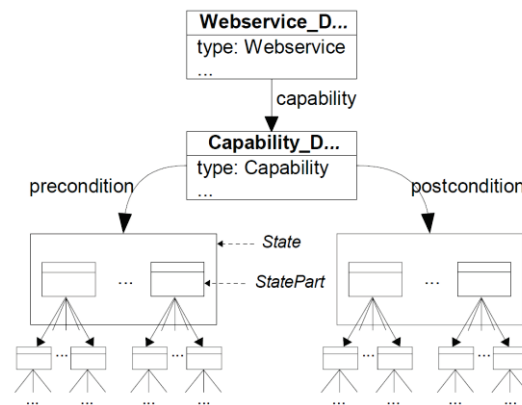


Abbildung 24: Service-Derivat in WSMO-MX, nach Klusch und Kaufer (2009), S. 26.

Analog zu Service-Derivaten existieren Derivate des *Goal* Konzeptes von WSMO. Somit können bereits bei der Anfrage Anforderungen an die Ergebnisse gestellt werden, beispielsweise, dass nur exakte und Subsumptions-Treffer oder solche mit einem bestimmten Schwellenwert für die syntaktische Ähnlichkeit generiert werden. Die Treffer-Kategorien sind hierbei ähnlich denen der bereits vorgestellten *-MX Ansätze: *equivalence*, *plug-in*, *inverse plugin (subsumption)*, *intersection* und *disjunction (fail)*. Weiterhin gibt es den Treffer *fuzzy similarity*, welcher die nicht-logikbasierten Ergebnisse auf Basis syntaktischer Ähnlichkeit beschreibt, und *neutral*, welcher weder einen Treffer noch keinen Treffer darstellt. Aus diesen Treffern wird ein Ergebnisvektor ($v = \pi_{\equiv}, \pi_{\sqsubseteq}, \pi_{\supseteq}, \pi_{\cap}, \pi_{\sim}, \pi_{\circ}, \pi_{\perp}$) mit $\pi_i \in [0,1]$ ($i \in \{\equiv, \sqsubseteq, \supseteq, \cap, \sim, \circ, \perp\}$). Die $\sum \pi_i$ drückt somit den Umfang der Übereinstimmung der Service- und Goal-Derivate aus. (Kaufer und Klusch (2006b), S. 166)

Durch die Servicederivate und deren Meta-Informationen ist es möglich, dem Nutzer detailliert Auskunft darüber zu geben, warum bestimmte Treffer entstanden sind. Somit können die Anwender ihre Suche einfacher verfeinern oder anpassen.

WSMO-MX bietet verschiedene Filter für das Matching der Ziele mit Services. *Type matching* arbeitet auf den Konzepten der Service- und Goal-derivate und bestimmt deren Ähnlichkeit auf Basis ihrer Relation innerhalb der Ontologie(n). So gibt es Gleichheit (*equivalent*), Unter- und Oberkonzepte (*sub- and supertype*), Geschwister (*sibling*, Konzepte mit gleichem direktem Vorfahren), Gatten (*spouse*, Konzepte mit gleichem direk-

tem Nachfahren), gemeinsame Vorfahren (*common ancestor*), gemeinsame Nachfahren (*common descendant*) sowie Verwandte (*relative*, es existiert ein Verbindungspfad in einer nicht-gerichteten Ontologie). Falls die Derivate komplex sind (das heißt nicht nur aus einem Konzept bestehen) werden ihre Beziehungen im *relation matching* verglichen. Dabei werden die Relationen des Ziels, wenn möglich, rekursiv mit den Relationen des Service verglichen. Auf Derivaten können wie erwähnt bestimmte Einschränkungen, sogenannte *constraints*, definiert werden. Dies sind Regeln in F-Logic, welche im Rahmen des *constraint matching* zwischen Service- und Goalderivat verglichen werden. Dabei können keine oder auch mehrere Regeln pro Derivat definiert werden. Zum Einsatz kommt ein *relative query containment*⁴¹ auf den in WSMO-MX enthaltenen Ontologien. Zusätzlich zu diesen logikbasierten Filtern erlaubt WSMO-MX die Berechnung der syntaktischen Ähnlichkeit zwischen den Derivaten. Die eingesetzten Ähnlichkeitsmetriken sind hierbei Kosinus-Maß, erweitertes Jaccard-Maß, loss-of-information (LOI) und gewichtetes LOI (siehe auch Seite 53ff und Klusch et al. (2005)). Dieser Filter beeinflusst den Ergebnisvektor im Grad der *fuzzy similarity* $\pi_{\sim}$ ab einem vordefinierten Schwellenwert α . Input- und Outputparameter der Derivate können separat annotiert werden. Diese Annotationen werden im Rahmen des *parameter matchings* überprüft. Eine Auswirkung auf den Ergebnisvektor entsteht nicht, allerdings werden Abweichungen zwischen den Parametern der Derivate als Annotation zurückgegeben. So kann der Nutzer eventuelle Anpassungen an der verwendeten Schnittstelle zum Service durchführen. Optional bietet WSMO-MX ein *intentional matching* nach Keller et al. (2005). Die Ergebnisse der vorgestellten Filter können durch voreingestellte oder personalisierte Gewichtung geordnet werden. (Klusch und Kaufer (2009), S. 31ff)

Die Implementierung von WSMO-MX⁴² basiert auf Apache-Tapestry⁴³ und Apache-HiveMind⁴⁴. Somit kann sie in einem kompatiblen JEE Container eingesetzt werden (zum Beispiel Apache-Tomcat⁴⁵). Dem Anwender wird eine webbasierte Benutzeroberfläche geboten, um Anfragen zu starten und eine Konfiguration des Matchmakers durchzuführen. Als Reasoner wird der kommerzielle OntoBroker⁴⁶ eingesetzt. (Klusch und Kaufer (2009), S. 35)

Zur Evaluierung der Implementierung wird eine Sammlung an Testservices und -goals verwendet, welche von den Autoren selbst entwickelt wurde. Die *Service Retrieval*

⁴¹ *Relative query containment* besagt: Regel A ist relativ in Regel B enthalten ($A \sqsubseteq_{\mathcal{KB}} B$) falls das Antwortset $Q_{\mathcal{KB}}(A)$ nach der Anfrage A auf Wissensbasis $\mathcal{KB}$ eine Untermenge von $Q_{\mathcal{KB}}(B)$ ist.

⁴² projects.semwebcentral.org/projects/wsmomx/

⁴³ tapestry.apache.org

⁴⁴ hivemind.apache.org

⁴⁵ tomcat.apache.org

⁴⁶ www.ontoprise.de/de/produkte/ontobroker/

Testcollection WSM-TC2 beinhaltet dazu 97 Services und 22 Anfragen mit über 2200 Derivaten. Die Auswertung der Implementierung zeigt vor allem, dass ein hybrides Matchmaking (logisch und syntaktisch) die besten Ergebnisse erzielt, indem es vor allem viele *false positives* und *false negatives* der jeweils anderen Matchingstrategie eliminiert. (Klusch und Kaufer (2009), S. 36ff; Klusch et al. (2008), S. 405ff)

Discovery im Web Service Modeling Execution Environment

Zur Umsetzung der Konzepte von WSMO (siehe Abschnitt 2.2.3.2) wurde eine Referenzimplementierung entwickelt: das Web Service Modeling eXecution environment (WSMX). Bevor die Implementierung der Discovery Komponente im WSMX beschrieben wird, soll zunächst das konzeptionelle Modell der Web Service Discovery in WSMO dargestellt werden. Das Modell basiert dabei auf den drei Ebenen der Service Discovery: Keyword Discovery, „leichtgewichtige“ und „schwergewichtige“ semantische Discovery (siehe Abschnitt 3.1.1).

Die Suche nach Schlüsselworten liegt nicht im Fokus der Discovery von WSMO, wird aber als wichtige Technik bei der Suche anerkannt. Für die Umsetzung einer solchen Suche wird auf bestehende Ansätze verwiesen. (Keller et al. (2007), S 303)

Die Suche nach Schlüsselworten hat vor allem den Nachteil, dass die Semantik der Begriffe nicht explizit definiert ist. Die Discovery auf Basis einfacher semantischer Beschreibungen setzt genau an diesem Punkt an. Zum Einsatz kommt ein kontrolliertes Vokabular in Form von Ontologien. Einfache semantische Beschreibungen basieren auf einem Set von Konzepten dieser Ontologien und beschreiben somit was ein Service liefern kann, jedoch nicht unter welchen Umständen (im Sinne von Vorbedingungen und Annahmen). Services werden daher nur in Form von Output und Effekt beschrieben. Die Verbindung eines Sets an Objekten zur Beschreibung des Service mit einem Set an Objekten zur Beschreibung eines Ziels ist über die Analyse der Ontologien möglich, mit denen die verwendeten Konzepte verknüpft sind. (Keller et al. (2007), S. 303-305)

Ein wichtiger Aspekt bei dieser Form der Discovery ist die Möglichkeit des Modellierers frei zu entscheiden, wie das Set an Konzepten interpretiert werden kann. Zum einen kann gewünscht sein, dass *alle* Elemente des Ziels in dem zu findenden Service enthalten sein sollen. Zum anderen kann es dem Modellierer ausreichen, dass nur *einige* der Elemente in der Servicebeschreibung enthalten sind. Diese *Intention* beeinflusst die Suche nach relevanten Services deutlich und es ist daher sinnvoll, die *Intention* bei der Beschreibung des Ziels oder zu Beginn der Suche explizit zu definieren. Anfragen und Services werden dann in Form von Paaren ausgedrückt: $\mathcal{D} = (\mathcal{R}_{\mathcal{D}}, \mathcal{I}_{\mathcal{D}})$, wobei $\mathcal{R}_{\mathcal{D}}$ das Set an Objekten zur Funktionalitätsbeschreibung und $\mathcal{I}_{\mathcal{D}} \in \{\forall, \exists\}$ die entsprechende universelle ($\forall$) oder existenzielle ($\exists$) *Intention* ist.

Intention of $\mathcal{G}/\mathcal{A}$	$I_{\mathcal{A}} = \forall$		$I_{\mathcal{A}} = \exists$	
$I_{\mathcal{G}} = \forall$	$R_{\mathcal{G}} = R_{\mathcal{A}}$	Match	$R_{\mathcal{G}} = R_{\mathcal{A}}$	PossMatch
	$R_{\mathcal{G}} \subseteq R_{\mathcal{A}}$	Match	$R_{\mathcal{G}} \subseteq R_{\mathcal{A}}$	PossMatch
	$R_{\mathcal{G}} \supseteq R_{\mathcal{A}}$	ParMatch	$R_{\mathcal{G}} \supseteq R_{\mathcal{A}}$	ParMatch
	$R_{\mathcal{G}} \cap R_{\mathcal{A}} \neq \emptyset$	ParMatch	$R_{\mathcal{G}} \cap R_{\mathcal{A}} \neq \emptyset$	PossParMatch
	$R_{\mathcal{G}} \cap R_{\mathcal{A}} = \emptyset$	NoMatch	$R_{\mathcal{G}} \cap R_{\mathcal{A}} = \emptyset$	NoMatch
$I_{\mathcal{G}} = \exists$	$R_{\mathcal{G}} = R_{\mathcal{A}}$	Match	$R_{\mathcal{G}} = R_{\mathcal{A}}$	Match
	$R_{\mathcal{G}} \subseteq R_{\mathcal{A}}$	Match	$R_{\mathcal{G}} \subseteq R_{\mathcal{A}}$	PossMatch
	$R_{\mathcal{G}} \supseteq R_{\mathcal{A}}$	Match	$R_{\mathcal{G}} \supseteq R_{\mathcal{A}}$	Match
	$R_{\mathcal{G}} \cap R_{\mathcal{A}} \neq \emptyset$	Match	$R_{\mathcal{G}} \cap R_{\mathcal{A}} \neq \emptyset$	PossMatch
	$R_{\mathcal{G}} \cap R_{\mathcal{A}} = \emptyset$	NoMatch	$R_{\mathcal{G}} \cap R_{\mathcal{A}} = \emptyset$	NoMatch

Abbildung 25: Interaktion zwischen set-theoretischen Kriterien, Intentionen und intuitiven Verständnis des Matchings, nach Keller et al. (2007), S. 307.

Durch die Verknüpfung der Intention von Ziel ($\mathcal{G}$) und Service ($\mathcal{A}$) ergibt sich ein Matriz, welche Treffer (*Match*), partielle Treffer (*ParMatch*), mögliche Treffer (*PossMatch*), mögliche partielle Treffer (*PossParMatch*) und keine Treffer (*NoMatch*) darstellt (siehe Abbildung 25). Beispielhafte Treffer nach Keller et al. (2007) (S. 307ff) sind:

- (1) Treffer - $I_{\mathcal{G}} = \forall, I_{\mathcal{A}} = \forall, R_{\mathcal{G}} \subseteq R_{\mathcal{A}}$: Die Anfrage definiert, dass alle definierten Elemente relevant sind. Der Anbieter kann in diesem Fall alle Objekte in der Servicebeschreibung $R_{\mathcal{A}}$ liefern. In diesem Fall wird die Anfrage komplett durch den angebotenen Service abgedeckt, da alle Objekte der Anfrage $R_{\mathcal{G}}$ durch $\mathcal{A}$ abgedeckt werden.
- (2) Partieller Treffer - $I_{\mathcal{G}} = \forall, I_{\mathcal{A}} = \forall, R_{\mathcal{G}} \cap R_{\mathcal{A}} \neq \emptyset$: Wie auch in (1) sind alle Objekte der Anfrage relevant und alle Objekte der Servicebeschreibung können durch den Anbieter geliefert werden. Allerdings gibt es nur eine Schnittmenge der definierten Objekte aus $R_{\mathcal{G}}$ und $R_{\mathcal{A}}$. Da der angebotene Service nur eine Teilmenge der angefragten Objekte bereitstellen kann und somit allerhöchstens verbunden mit einem (oder mehreren) weiteren Service(s) die Anfrage erfüllen kann, handelt es sich um einen partiellen Treffer.
- (3) Möglicher Treffer - $I_{\mathcal{G}} = \forall, I_{\mathcal{A}} = \exists, R_{\mathcal{G}} \subseteq R_{\mathcal{A}}$: Wieder definiert die Anfrage alle geforderten Objekte als relevant, der Anbieter kann aber nur einen Teil der in $R_{\mathcal{A}}$ beschriebenen Objekte liefern. Zusätzlich bilden die angefragten Objekte eine Teilmenge der angebotenen Objekte. Durch die existenzielle Intention des Anbieters kann jedoch nicht sicher gesagt werden, ob es sich wirklich um einen Treffer handelt. Durch die Prüfung einer detaillierteren Beschreibung der möglichen Treffer kann während der Service Discovery daher aber trotzdem ein relevanter Service gefunden werden.
- (4) Möglicher partieller Treffer - $I_{\mathcal{G}} = \forall, I_{\mathcal{A}} = \exists, R_{\mathcal{G}} \cap R_{\mathcal{A}} \neq \emptyset$: Wie auch in (3) sind alle angefragten Objekte relevant und der Anbieter kann nur einen Teil der in $R_{\mathcal{A}}$ beschriebenen Objekte liefern. Zudem existiert wie in (2) eine Schnittmenge der Anfrage und des Angebots. Bestenfalls kann der Anbieter

die Anfrage also nur teilweise erfüllen. Wieder ist nicht genau bekannt, welcher Teil der Anfrage möglicherweise erfüllt werden kann. Wie auch in (3) können weitere Informationen während des Service Discovery Prozesses eingeholt werden. Insgesamt handelt es sich um einen möglichen partiellen Treffer.

- (5) Kein Treffer - $I_G = \exists, I_A = \forall, R_G \cap R_A = \emptyset$: Die Anfrage versucht einen Teil der in R_G als relevant spezifizierten Objekte zu erhalten. Das Angebot kann alle in R_A definierten Objekte liefern. Die beiden Sets an Objekten besitzen aber keine Überschneidungen, wodurch die Anfrage nicht (auch nicht teilweise) durch das Angebot erfüllt werden kann. Es handelt sich um keinen Treffer.

Die von den Keller et al. (2007) vorgeschlagene mögliche existenzielle Intention des Anbieters $I_A = \exists$ kann aus Sicht des Autors zu Problemen führen, wie auch in Abbildung 25, rechte Spalte ersichtlich wird. Auf diese Weise kann es zu vielen „möglichen“ Treffern kommen, welche weitere detaillierte Informationen für eine Entscheidung benötigen. Daher sollten diese existenziellen Intentionen auf Seite des Anbieters vermieden werden.

Keller et al. (2007) bezeichnen die vorgestellte Art der Servicedefinition wie bereits erwähnt als einfache semantische Beschreibung. Diese einfache Form kann erweitert werden, um zum Beispiel höhere Formen der Automatisierung bei der Selektion und Ausführung von Services zu erreichen. Hierfür können zum einen die zur Beschreibung verwendeten Konzepte weiter detailliert und spezifiziert werden. Keller et al. (2007) schlagen hierzu die Nutzung einer Sprache der Prädikatenlogik (erster Ordnung, FOL) oder ein Subset dieser vor. Eine erweiterte Form der Servicebeschreibung sollte zum anderen den Zusammenhang zwischen Annahmen und Inputs sowie Outputs, Effekten und Nachbedingungen beinhalten. (Keller et al. (2007), S. 314)

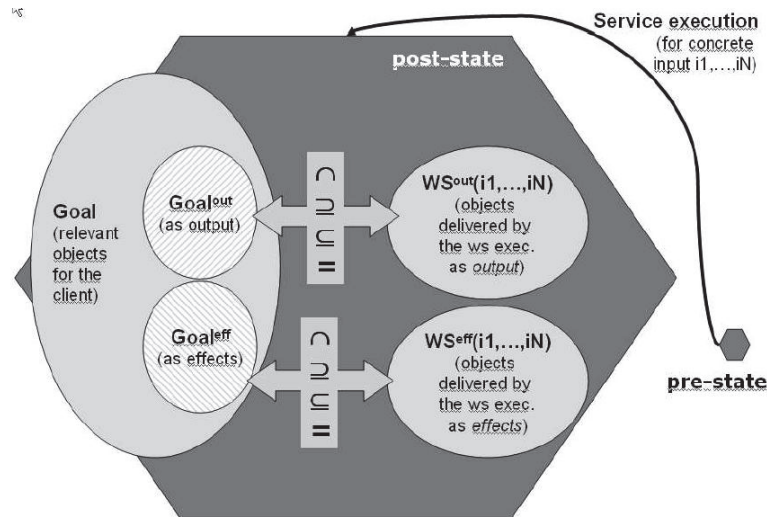


Abbildung 26: Erweitertes Servicemodell, nach Keller et al. (2007), S 316.

Dieses erweiterte Servicemodell ist in Abbildung 26 zu sehen. Ein Web Service wird mit einem konkreten Set an Eingabewerten $(i_1, \dots, i_N)$ aufgerufen, welches in einem konkreten Zustand der Welt existiert (*pre-state*). Die Ausführung des Web Service führt zu einem anderen Zustand der Welt (*post-state*), in welchem vom Service ein Set an Ausgabewerten $(ws^{out}(i_1, \dots, i_N))$ und ein Set von Effekten $(ws^{eff}(i_1, \dots, i_N))$ geliefert wird. Da in WSMO das Ziel aus einem Set an gewünschten Informationen ($goal^{inf}$) und einem Set an gewünschten Effekten ($goal^{eff}$) besteht, können im Matchingverfahren die Zusammenhänge zwischen Outputs $(ws^{out}(i_1, \dots, i_N))$ und $goal^{inf}$ und Effekten $(ws^{eff}(i_1, \dots, i_N))$ und $goal^{eff}$ untersucht werden. (Keller et al. (2007), S. 315)

Obwohl Keller et al. (2007) Annahmen und Vorbedingungen als Teil der Servicebeschreibung benennen, werden diese weder formal definiert noch in der Definition von Zielen verwendet. Mögliche Formen der Definition bieten allerdings die Autoren in früheren Publikationen (Roman et al. (2005); Lausen et al. (2007)).

Die Umsetzung der vorgestellten Konzepte der Service Discovery mittels WSMO wurde (zumindest teilweise) in WSMX realisiert. In Abbildung 27 ist das *Service Discovery Interface* und die entsprechende Implementierung der Komponente konzeptionell dargestellt. Zur Durchführung des Matchings kann nach Bedarf auf unterschiedliche Reasoner zugegriffen werden. Ermöglicht wird dies durch das WSM2Reasoner Framework¹⁷. Für die Verarbeitung der Ziele und Services wird WSMO4j¹⁸ eingesetzt. Weitere Informationen und eine Erklärung der gesamten WSMX Architektur finden sich in Haller et al. (2005) und Bussler et al. (2005).

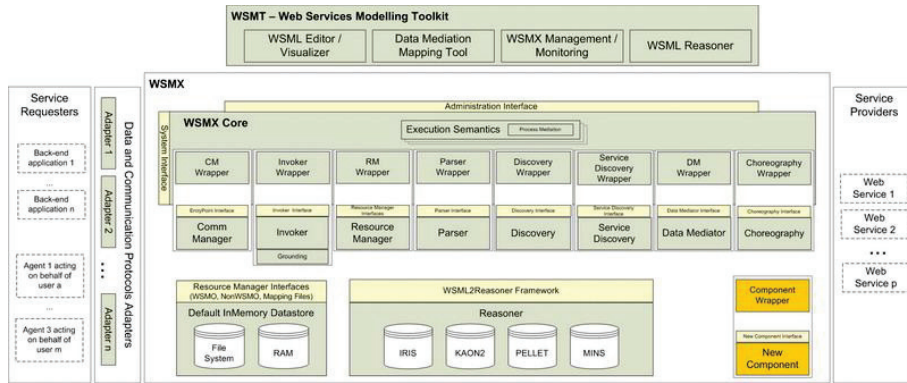


Abbildung 27: WSMX Architektur, siehe www.wsmx.org.

¹⁷ tools.sti-innsbruck.at/wsm2reasoner

¹⁸ wsmo4j.sourceforge.net

Weitere Ansätze zur Discovery auf Basis des WSMO Frameworks finden sich in Si und Zhefeng (2009) sowie in Zaremba et al. (2009).

3.2 Semantisches Geschäftsprozessmanagement

Im Umfeld des Geschäftsprozessmanagements bietet vor allem die Automatisierung der Prozesse Vorteile für die Unternehmen, seien es die Beschleunigung manueller Prozesse oder auch die Kontrolle von Performanz durch die Überwachung verschiedener Indikatoren. Die Automatisierung der Prozesse bringt aber zusätzliche Herausforderungen mit sich, vor allem in Zeiten stetigen Wandels und einer Vielzahl bestehender Systeme und Prozesse. Für Unternehmen existieren unter anderem drei Dimensionen, in welchen sie (bezüglich der Automatisierung mittels BPM) mit anderen konkurrieren: (1) Kosten pro Prozessausführung, (2) Kosten pro Prozesseinrichtung und (3) Verzögerung in der Prozesseinrichtung. Somit müssen Unternehmen vor allem *effizient* (geringe Kosten pro Transaktion im operationellen Geschäft) und *agil* (geringe Verzögerung bei der Implementierung neuer und Änderung bestehender Prozesse) sein sowie *in kleinen Iterationen ihre Prozesswelt erweitern können*. Letzteres ist vor allem nötig, um auch einfache Verbesserungen schnell implementieren und kleinere Geschäftsmöglichkeiten wahrnehmen zu können. (Hepp et al. (2005), S. 536)

Die Automatisierung von Geschäftsprozessen und deren Überwachung im Sinne des Geschäftsprozessmanagements beinhaltet vor allem zwei Sichten: Geschäftsprozessexperten betrachten Prozesse aus Sicht der Fachseite und IT-Experten aus Sicht der Implementierung. Die bestehende Kluft zwischen diesen zwei Seiten konnte bisher nicht ausreichend durch Automatisierung behoben werden und somit entsteht ein Flaschenhals durch die manuelle Überbrückung dieser. Ein Beispiel solcher manueller Arbeiten ist die Implementierung von bereits modellierten Prozessen (zum Beispiel in Form von EPK oder BPMN) mittels Ausführungssprachen wie BPEL oder proprietären Lösungen eines bestimmten Herstellers. Aufwände entstehen zum einen durch fehlende oder lückenhafte automatische Konvertierung der Sprachen und zum anderen durch Missverständnisse oder fehlende Informationen zu der Bedeutung der Prozesse und Prozessbestandteile. Auch die Suche nach bestehenden Prozessen auf Basis bestimmter Merkmale oder Kennzahlen ist meist mit manuellem Aufwand verbunden. Diese Kluft zwischen Fachseite und IT ist in Abbildung 28 skizziert.

The Critical IT / Process Divide

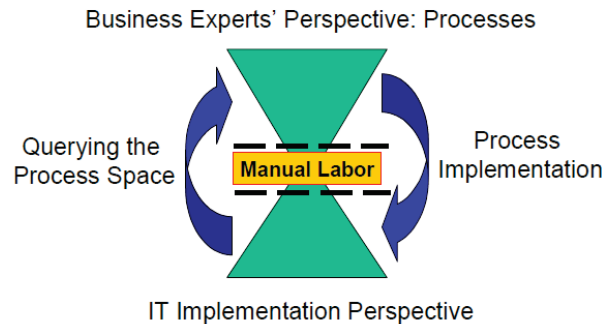


Abbildung 28: Die bi-direktionale Kluft zwischen Fachseite und IT, nach Hepp et al. (2005), S. 537.

Bestehende Werkzeuge des Geschäftsprozessmanagements versuchen bereits diese Kluft zu verringern, zum Beispiel durch die Verwendung von Geschäftsprozesssammlungen (*engl.* Repositories) oder gemeinsame Modellierungssprachen für Fach- und IT-Seite. Auf diese Weise kann zumindest die Syntax (zum Beispiel bei der Modellierung) harmonisiert werden, dies garantiert aber nicht eine gemeinsame Semantik. Hepp et al. (2005) schlagen daher eine explizite Beschreibung der Semantik von Geschäftsprozessen und deren Implementierung auf Basis der Erkenntnisse im Bereich semantischer Webservices vor. Die Kombination von Geschäftsprozessmanagement und semantischen Webservices wird als semantisches Geschäftsprozessmanagement (*engl.* semantic Business Process Management – sBPM) bezeichnet.

Nach Hepp et al. (2005) (S. 537) benötigt ein solches semantisches Geschäftsprozessmanagement:

- (1) die Darstellung und Beschreibung aller Prozesse als semantische Webservices (atomar und zusammengesetzt) einer Organisation in einem geeigneten Repository,
- (2) die Erfassung der gesamten IT-Landschaft (u.a. Hardware, Betriebssysteme, Fertigungssysteme) in Form einer Ontologie,
- (3) die Erfassung von Domänenwissen (u.a. Geschäftsregeln und technische Randbedingungen) sowie dessen Speicherung in Form von Axiomen einer Regelsprache (zum Beispiel als Teil einer Ontologiesprache),
- (4) die Abbildung von transaktionalen Daten der bestehenden Systeme (zum Beispiel des Enterprise Resource Planning (ERP)) und Speicherung als Instanzen,
- (5) die Darstellung von Anfragen in einer Ontologiesprache (query language),
- (6) die Modellierung von Anforderungen der Fachseite (Geschäftsprozessexperten) als formalisierte Ziele

- (7) und den Einsatz einer Ausführungsumgebung für semantische Webservices, welche auch die Mediation zwischen Zielen und Anfragen der Fachseite im Hinblick auf die bestehende Prozesswelt erlaubt.

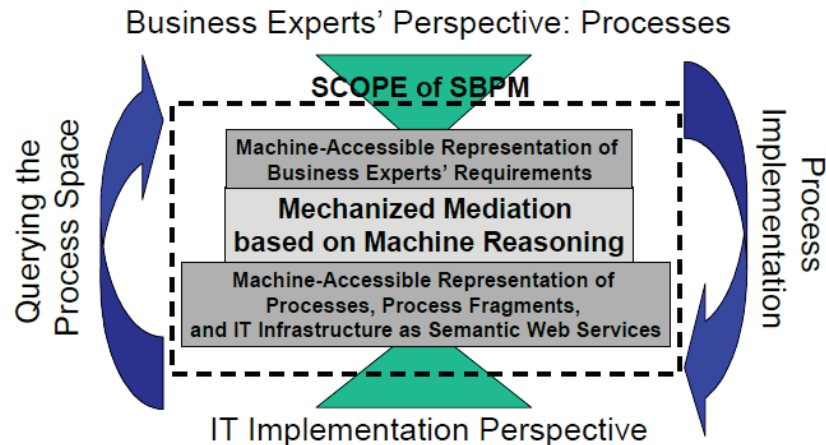


Abbildung 29: Semantisches Geschäftsprozessmanagement, nach Hepp et al. (2005), S. 538.

Der Umfang eines solchen semantischen Geschäftsprozessmanagements ist in Abbildung 29 skizziert. Somit soll zum einen die Suche innerhalb der Prozesswelt automatisiert werden, indem Anfragen auf Basis von Ontologiekonzepten und Einschränkungen formal definiert werden können (zum Beispiel die Darstellung aller Systeme abhängig von System X). Zum anderen sollen formalisierte Ziele zur Implementierung neuer Geschäftsprozesse auf Basis bestehender Services genutzt werden. Durch die starke Formalisierung ist es weiterhin möglich Referenzprozesse zu definieren, welche durch Adaption in verschiedenen Unternehmen oder Abteilungen eingesetzt werden können.

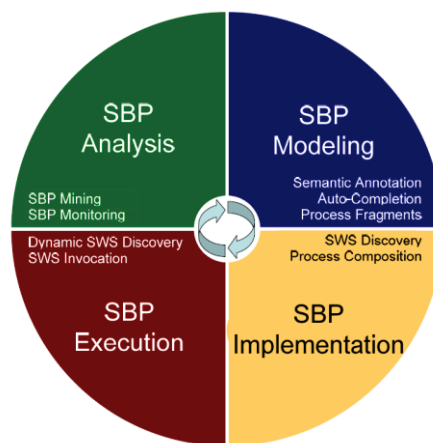


Abbildung 30: Der Semantic Business Process Modeling Lifecycle, nach Wetzstein et al. (2007).

Die Anforderungen an ein solches semantisches Geschäftsprozessmanagement sind vielschichtig. Für eine bessere Analyse dieser Anforderungen haben Wetzstein et al. (2007) ein Lebenszyklusmodell des Geschäftsprozessmanagements herangezogen. Dieses ist nahezu deckungsgleich mit dem Hauptzyklus des Geschäftsprozessmanagements in Abbildung 16 (siehe Abschnitt 2.4.2, S. 44). Es besteht aus Analyse, Modellierung, Implementierung und Ausführung von semantischen Geschäftsprozessen (*engl.* Semantic Business Processes – SPB). Jede Phase beinhaltet mehrere Teilbereiche. Die Modellierung als erste Phase nutzt semantische Annotationen, Auto-Vervollständigung und Prozessfragmente zur Darstellung von SBPs. Für die Implementierung der Prozesse stehen das Auffinden von semantischen Web Services (Discovery) sowie die Komposition im Vordergrund. In der Ausführungsphase soll eine dynamische Discovery der Services genutzt werden (vollautomatisch). Sowohl vordefinierte als auch dynamisch gefundene Services werden mittels semantischer Technologien aufgerufen (SWS Invocation). Abschließend werden in der Analysephase semantische Geschäftsprozesse überwacht (Monitoring) und ausgewertet (Mining). Eine Übersicht über den Lebenszyklus des semantischen Geschäftsprozessmanagements gibt Abbildung 30.

Anhand des Lebenszyklus werden in den folgenden Abschnitten alle Phasen vorgestellt und bestehende Ansätze aus der Forschung in der jeweiligen Phase diskutiert. Eine Übersicht über die einzelnen Phasen des sBPM Lebenszyklus findet sich auch in Pedrinaci et al. (2008).

3.2.1 Modellierung semantischer Geschäftsprozesse

Die Modellierungsphase führt zu semantisch annotierten Geschäftsprozessmodellen. Ziel ist es, die Semantik sowohl der Aktivitäten als auch der Entscheidungen eines Prozessflusses zu definieren. Es soll explizit definiert werden, in Form von Referenzen auf Konzepte von Ontologien, was in den Aktivitäten des Prozesses erreicht werden soll. Dies erweitert die Beschreibung durch die natürliche Sprache. Die Annotation der Aktivitäten erlaubt im Weiteren vor allem die (halb-)automatische Suche nach semantischen Web Services zur Implementierung der Aktivitäten oder ganzer Prozessfragmente. (Wetzstein et al. (2007), S. 4-5)

Wetzstein et al. (2007) beschreiben die benötigten Funktionalitäten und Anwendungsfälle in der Modellierungsphase (Wetzstein et al. (2007), S. 5):

- *Semantische Annotation von Geschäftsprozessmodellen:* Die Modellierung von Prozessen wird, wie in der klassischen Geschäftsprozessmodellierung auch, durch bestehende Notationen wie BPMN oder (e)EPK realisiert. Zusätzlich wird dem Modellierer die Möglichkeit gegeben, die einzelnen Elemente des Prozesses mit Konzepten aus einer oder mehreren Ontologien zu annotieren. Hierbei können Referenzen auf verschiedene Arten von Ontologien zum Einsatz kommen. Eine organisatorische Ontologie erlaubt zum Beispiel die Referenzierung auf Abteilungen, deren Aufgaben ausgeführt werden. IT Services zur Implementierung der Aktivitäten können mittels SWS Ontologien

annotiert werden. Domänenontologien erlauben die Beschreibung der im Prozess verwendeten Daten. Weitere Beispiele finden sich zudem in Hepp und Roman (2007). Durch die Referenzen auf verschiedene Ontologien entsteht die gewünschte explizite Darstellung der Semantik, wodurch gleichzeitig eine Maschinenlesbarkeit erreicht wird.

- *Wiederverwendung von Geschäftsprozessfragmenten*: Prozessfragmente sind Teile eines Prozessmodells, welche potentiell wiederverwendbar sind. Teile eines semantischen Geschäftsprozesses können ausgewählt und zur späteren Wiederverwendung in einem Repository gespeichert werden. Existieren solche Fragmente bereits, so kann vor oder auch während der Modellierung nach existierenden Fragmenten gesucht werden. Die Annotationen der Fragmente erlauben die semantische Suche nach Prozessen, wodurch eine höhere Treffergenauigkeit im Vergleich zu bestehenden Ansätzen bei der Wiederverwendung von Prozessbestandteilen erreicht werden soll. Ein solcher Ansatz findet sich zum Beispiel in Thomas (2006). Der Modellierer oder Analyst kann zur Suche unter anderem die Domäne und die gewünschte Funktionalität des Fragments definieren. Auch nach Referenzen auf bestimmte Ontologiekonzepte kann gesucht werden.
- *Automatische Vervollständigung*: Die Funktionalität der automatischen Vervollständigung von Begriffen oder Funktionen ist bereits in Suchmaschinen oder integrierten Entwicklungsumgebungen weit verbreitet. Eine solche Vervollständigung soll für die semantische Geschäftsprozessmodellierung auch für die Suche nach Prozessfragmenten verwendet werden. Hierzu kann der Modellierer einen noch nicht vollständigen Teil des Prozesses auswählen und nach Vorschlägen zur Vervollständigung suchen. Hierbei wird auf das bereits erwähnte Repository an Prozessfragmenten zugegriffen.

Eine Grundvoraussetzung dieser Funktionalitäten und Anforderungen ist die Definition und Verwendung einer semantischen Geschäftsprozessmodellierungssprache. Bestehende Ansätze aus diesem Bereich werden im folgenden Abschnitt vorgestellt. Anschließend werden Beispiele zur Umsetzung der durch Wetzstein et al. (2007) genannten Punkte gegeben.

3.2.1.1 Semantische Geschäftsprozessmodellierungssprachen

Durch die große Akzeptanz bestehender Prozessmodellierungssprachen bauen die meisten Ansätze zur semantischen Geschäftsprozessmodellierung auf eine Erweiterung dieser Sprachen. Im Weiteren wird daher beispielhaft je ein Ansatz für die BPMN und EPK vorgestellt.

Thomas und Fellmann (2009) erläutern in ihrer Arbeit das Problem der Nutzung natürlicher Sprache zur Beschreibung von Prozessmodellen. Zum einen entstehen durch die natürliche Sprache Interpretationsspielräume wie Synonyme oder Homonyme. Zum anderen kann die Semantik ohne eine Formalisierung nicht Maschinen interpretiert werden. Zur Lösung dieser Probleme schlagen die Autoren eine Erweiterung des

semiotischen Dreiecks (siehe Abbildung 6, S. 23) vor. Dabei ist ein Modell das Symbol, welches für ein Objekt, in diesem Fall ein Prozess, steht. Das Modell referenziert auf eine Ontologie, in welcher die Konzepte der Symbole formalisiert sind (siehe Abbildung 31). Zur Realisierung dieser Erweiterung verwenden Thomas und Fellmann (2009) die EPK (siehe Abschnitt 2.4.3.3) als Modellierungssprache und OWL (siehe Abschnitt 2.2.2.1) als Ontologiesprache. (Thomas und Fellmann (2009), S. 507-509)

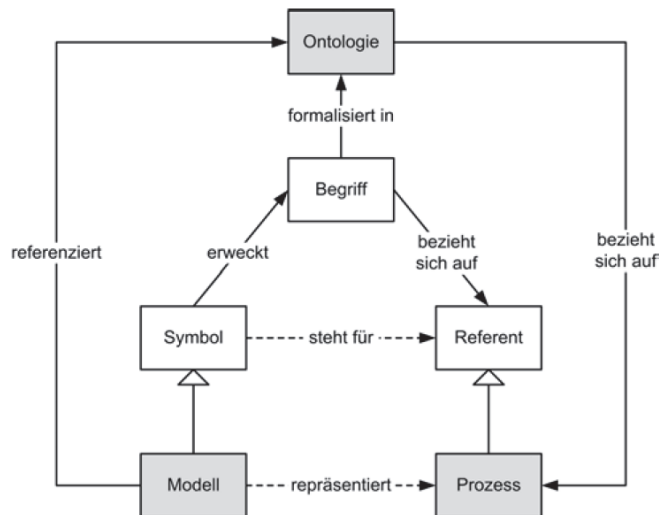


Abbildung 31: Erweitertes semiotisches Dreieck, nach Thomas und Fellmann (2009), S. 508.

Die Repräsentation der Prozesse innerhalb einer Ontologie erfolgt bei Thomas und Fellmann (2009) unter Nutzung der Graphentheorie. Die Aktivitäten, Ereignisse und Entscheidungen werden als Knoten in der Ontologie instanziiert und durch Eigenschaften dieser miteinander verbunden (Kanten). Die Instanzen sind entsprechend ihrer Art verschiedenen Klassen (EventNode, ActivityNode, GatewayNode) zugeordnet. Durch diese Darstellung können die einzelnen Instanzen innerhalb eines Prozesses durch weitere Ontologiekonzepte semantisch annotiert werden. Zur Annotation kann eine Domänenontologie eingesetzt werden. Diese beinhaltet Instanzen sowohl zu Aktivitäten, Ereignissen und Entscheidungen als auch weitergehenden Konzepten wie Organisation oder Serviceimplementierungen. Die Verbindung der Prozessinstanzen mit den Domäneninstanzen wird durch die Verwendung einer semantischen Äquivalenz realisiert (siehe Abbildung 32). Allerdings schlagen die Autoren die Nutzung reiner 1:1 Beziehungen zwischen Prozess- und Domäneninstanz vor, welches leicht zu einer Explosion der Konzepte innerhalb der Domänenontologie führen kann. (Thomas und Fellmann (2009), S. 510-511)

Obwohl die Arbeit von Thomas und Fellmann (2009) auch für die BPMN eingesetzt werden kann, soll ein weiterer Ansatz für die semantische Annotation von Geschäftsprozessmodellen vorgestellt werden.

Abramowicz et al. (2007) präsentieren ein Konzept zur semantischen Erweiterung der BPMN. Wie auch Thomas und Fellmann (2009) zielen die Autoren dabei auf die explizite Beschreibung der Bedeutung eines Prozesses und die Maschinenlesbarkeit dieser ab. Zur Formalisierung der Semantik wird WSML (siehe Abschnitt 2.2.2.1) als Ontologiesprache verwendet. Die sogenannte semantische BPMN (sBPMN) beschreibt die Bestandteile der Modellierungssprache in Form von Ontologiekonzepten. Für die oberste Ebene der Ontologie bedeutet das die Einführung von Konzepten für die Flussobjekte, Verbindungsobjekte, Artefakte und „Swimlanes“ (siehe Abbildung 33). Flußobjekte sind Ereignisse, Aktivitäten und Entscheidungen des Prozesses. Verbindungsobjekte sind der Sequenzfluss, Nachrichtenfluss und die Assoziation. Artefakte sind Datenobjekte, Gruppen und Annotationen. „Swimlanes“ beschreiben organisatorische Grenzen in Form von Abteilungen (Lanes) und Organisationen (Pools). Die Art und Weise der Verknüpfung der Prozessontologie mit etwaigen Domänenontologien wird von den Autoren nicht näher spezifiziert. (Abramowicz et al. (2007))

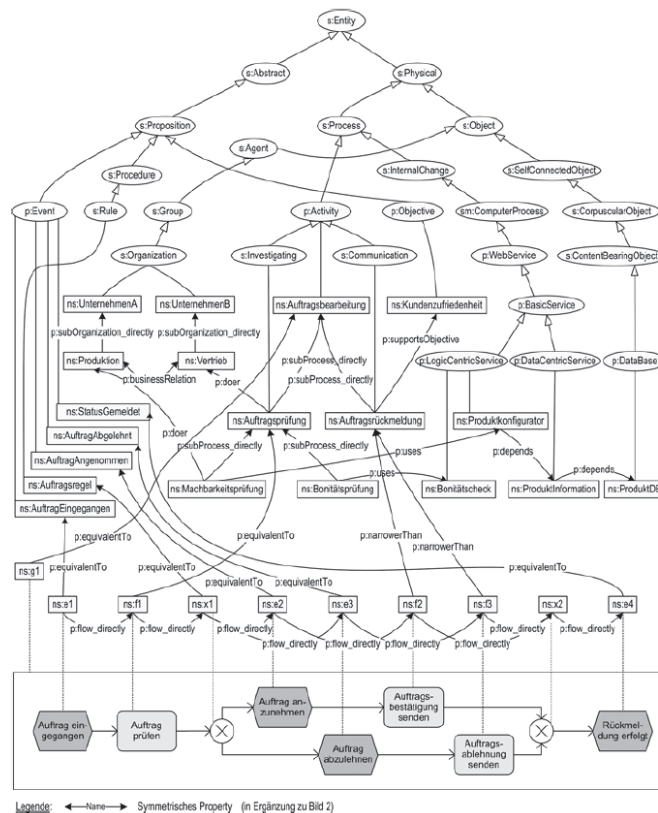


Abbildung 32: Beispielhafte semantische Annotation einer EPK, nach Thomas und Fellmann (2009), S. 511.

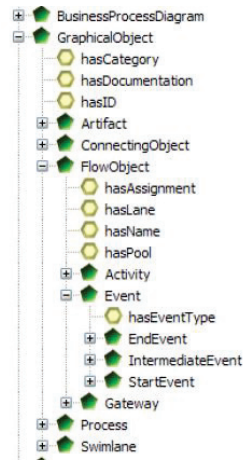


Abbildung 33: Bestandteile der sBPMN (nicht vollständig), nach Jędrzejczak et al. (2008), S. 9.

3.2.1.2 Umsetzung der Modellierung semantischer Geschäftsprozesse

Ein Ansatz zur semantischen Geschäftsprozessmodellierung und Implementierung wird von Born et al. (2007) präsentiert. Dieser wird im folgenden bezüglich der Anforderungen aus Wetzstein et al. (2007) für die erste Phase des sBPM Lebenszyklus diskutiert.

Bei der *semantischen Annotation von Geschäftsprozessmodellen* steht bei Born et al. (2007) vor allem die Benutzerfreundlichkeit im Vordergrund. Hierzu verwenden die Autoren bestehende Artefakte der BPMN: Datenobjekte und Assoziationen. Das Ziel ist die Annotation des Geschäftsprozesses während der Modellierung einfach für den Geschäftsprozessexperten zu halten. Hierzu sollen die Datenobjekte mit Konzepten aus Domänenontologien verbunden werden können. Zusätzlich zu den Objekten kann auch deren Status definiert werden. Dieser kann, je nach Objekt, verschiedene Transitionen durchlaufen (zum Beispiel *Angebot gesendet* folgt auf *Angebot erstellt*), welches auch eine spätere Suche nach fehlenden Prozessfragmenten erleichtern soll. Hierbei können bestimmte Transitionen mehrere Objekte beeinflussen (zum Beispiel kann ein *erstelltes Angebot* gleichzeitig einen *Auftrag* initiieren). Weiterhin können natürlichsprachliche Vor- und Nachbedingungen der Aktivitäten erstellt werden. Der Ansatz ist in Abbildung 34 dargestellt. (Born et al. (2007), S. 262-264)

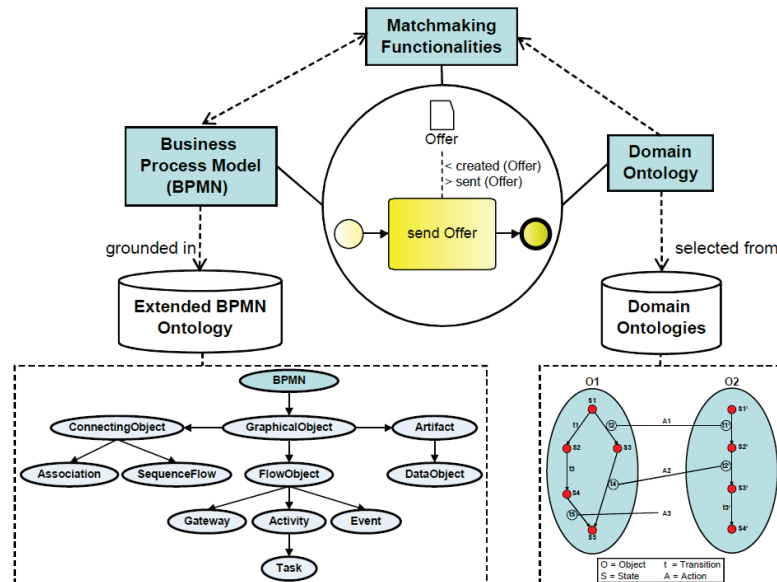


Abbildung 34: Realisierung semantischer Geschäftsprozessmodellierung, nach Born et al. (2007), S: 263.

Eine *automatische Vervollständigung* von Begriffen zur Annotation wird von Born et al. (2007) angedeutet, allerdings nur in Form von Suchfunktionalitäten umgesetzt. So ist es möglich nach den Begriffen innerhalb der Prozesselemente (Datenobjekte, Status, Aktivität) in den Domänenontologien zu suchen und mit den Ergebnissen zu verbinden. Weiterhin kann eine kontextbasierte Suche verwendet werden, welche auf Basis bereits annotierte Prozessbestandteile etwaige Nachfolger oder Vorgänger finden kann (zum Beispiel nachdem ein *Angebot gesendet* wurde, kann der *Auftrag ausgeführt* werden). Dies ist zudem bereits ein Schritt in die *automatische Vervollständigung* von ganzen Prozessfragmenten und der Wiederverwendung dieser, welche in Born et al. (2009) vertieft wird. (Born et al. (2007), S. 265-266)

Die *Wiederverwendung von Geschäftsprozessfragmenten* wird nicht explizit unterstützt. Implizit werden allerdings Fragmente der Prozesse in Form der Konzepte und Status der Domänenontologie verwendet. Für die Suche in Geschäftsfragmenten als Grundlage der Wiederverwendung sei hier auf Markovic und Pereira (2008) verwiesen. Eine prototypische Implementierung des vorgestellten Ansatzes wurde im „SAP-Research“ Modellierungstool „Maestro for BPMN“ realisiert. „Maestro“ ist allerdings nicht frei verfügbar. (Born et al. (2007), S. 266)

Als Alternative zur Modellierung semantischer Geschäftsprozesse in einer integrierten Entwicklungsumgebung präsentieren Dimitrov et al. (2007c) den BPMO Editor auf Basis des WSMO Studios⁴⁹. Die Anforderungen des Editors leiten sich direkt aus Hepp und Roman (2007) ab und umfassen die Möglichkeit der semantischen Annotation eines Geschäftsprozesses, die Speicherung der annotierten Prozessmodelle für spätere Suche und Wiederverwendung sowie die Transformation der Modelle in ausführbare Sprachen. Somit bietet der BPMO Editor eine Open Source Alternative, basierend auf Eclipse, zum oben erwähnten „Maestro“ Werkzeug. Der Editor (siehe Abbildung 35) bietet eine Modellierungsfläche für die erweiterte sBPMN Sprache. Wie in bestehenden Modellierungswerkzeugen können einzelne Elemente via Drag-and-Drop auf die Modellierungsfläche gezogen und miteinander verknüpft werden. Zusätzlich bietet die Oberfläche Zugriff auf den WSMO Explorer, welcher Konzepte der geladenen Ontologien zur Annotation des Prozesses zur Verfügung stellt. So können zum Beispiel Vor- und Nachbedingungen in Form von Axiomen an einzelne Aktivitäten annotiert werden. Es ist zudem möglich, vordefinierte Ziele für die spätere Servicesuche mit Elementen des Prozesses zu verknüpfen. (Dimitrov et al. (2007c))

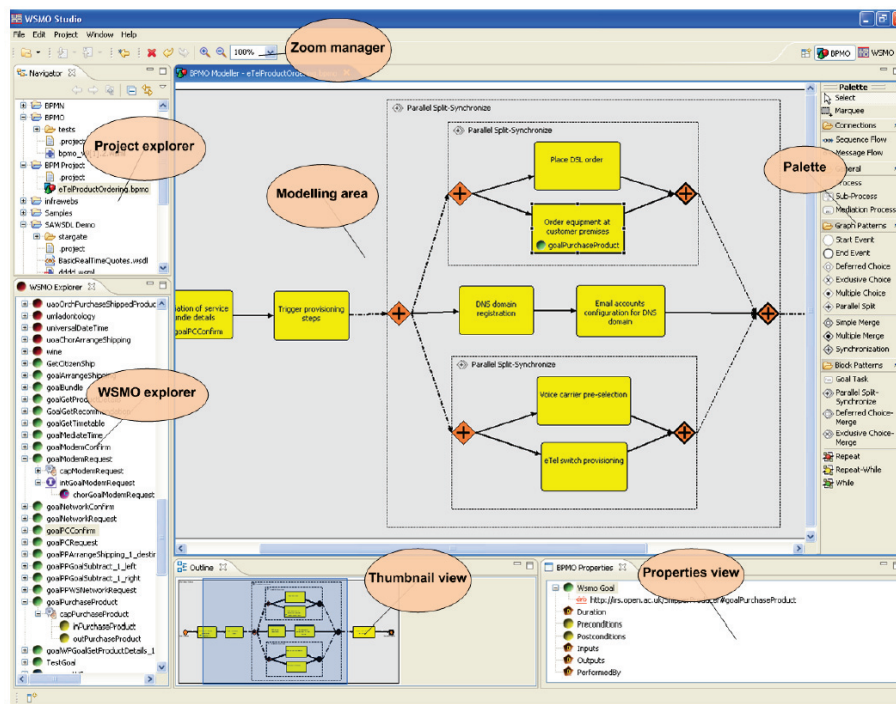


Abbildung 35: BPMO Editor Ansicht im WSMO Studio, nach Dimitrov et al. (2007c), S. 103.

⁴⁹ <http://www.wsmostudio.org>

Die Entwicklung des BPMO Editors wurde anscheinend eingestellt. Letzte Aktivitäten sind im Jahre 2008 zu verzeichnen⁵⁰.

Ein weiterer Ansatz zur Nutzung von semantischen Technologien im Bereich der Modellierung von Geschäftsprozessen präsentieren Di Francescomarino et al. (2009). Im Gegensatz zu Wetzstein et al. (2007) sehen die Autoren den Nutzen einer semantischen Annotation eines Geschäftsprozesses vor allem in der einfacheren Validierung desselben. Es wird vorgeschlagen, die Aktivitäten, Datenobjekte und Ereignisse eines BPMN Diagramms mit Konzepten einer OWL-Ontologie zu annotieren. Da die Konzepte der Ontologien bestimmten Klassen (entsprechend *Aktionen*, *Objekten*, *Ereignissen*) zugeordnet werden, ist es möglich die korrekte Annotation eines Diagramms durch Axiome zu validieren. Hierzu werden das Diagramm und die beinhalteten Annotationen in eine sogenannte BPMN-Ontologie überführt. Die entstehende Wissensbasis besteht somit aus den Instanzen der Geschäftsprozessdiagramme und Axiomen sowie Einschränkungen auf Basis von BPMN- und Domänenontologien. Auf diese Weise können strukturelle und inhaltliche Anforderungen an die Modellierung gestellt und durch eine formale Darstellung validiert werden. (Di Francescomarino et al. (2009))

3.2.2 Implementierung semantischer Geschäftsprozesse

In der Implementierungsphase sollen die semantischen Geschäftsprozesse aus der Modellierung in ausführbare Prozessmodelle überführt werden. Hierfür ist eine Transformation der Prozessbeschreibung nötig, da das bestehende semantische Prozessmodell nicht alle nötigen Informationen für die Ausführung beinhaltet. Eine solche Transformation besteht aus dem Auffinden passender semantischer Web Services zur Unterstützung der Aktivitäten, der Spezifizierung des Datenflusses und der Erstellung eines Prozessmodells, welche von der zu verwendenden Prozess-Engine verstanden wird (zum Beispiel BPEL). Die semantische Annotation des Prozesses erlaubt hier einen höheren Grad der Automation von der Modellierung zur Implementierung. Die Beschreibung der Aktivitäten kann verwendet werden, um passende semantische Web Services aufzufinden. Diese Suche kann durch Planungsalgorithmen unterstützt werden, um Aktivitäten durch eine Kombination von Services zu unterstützen. Ohne semantische Beschreibungen muss die Transformation in ein ausführbares Prozessmodell größtenteils manuell erfolgen. (Wetzstein et al. (2007), S. 6)

Wetzstein et al. (2007) beschreiben die benötigten Funktionalitäten und Anwendungsfälle in der Implementierungsphase (Wetzstein et al. (2007), S. 6-7):

- *Semantische Web Service Suche (Discovery)*: Ein Repository an semantischen Web Services hält Beschreibungen dieser vor. Die semantische Annotation einer (oder mehrerer) Aktivitäten des Prozesses wird als Basis für die Suche ge-

⁵⁰ Siehe <http://www.wsmstudio.org/multiproject/org.wsmstudio.bpmo>

- nutzt und mit den vorliegenden Beschreibungen verglichen. Zum Stand der Forschung auf diesem Gebiet sei auf Abschnitt 3.1 verwiesen.
- *Prozesskomposition*: Eine Prozesskomposition hat zum Ziel durch automatische Suche nach und eventuell Kombination von SWS eine Aktivität, ein Prozessfragment oder den gesamten Prozess zu implementieren. Nachdem die Modellierung des Prozesses in der ersten Phase abgeschlossen ist, kann während der Komposition eine Vielzahl von Services und bereits komponierten Prozessfragmenten durchsucht und zur Unterstützung des Prozesses eingesetzt werden. Wird die gewünschte Funktionalität erreicht so können die benötigten Informationen zur Ausführung in das Modell übernommen werden. Abschließend wird es auf seine Korrektheit überprüft und im Prozess-Repository gespeichert.
 - *Manuelle Verfeinerung*: Die vollständige Automation der Überführung eines modellierten in einen ausführbaren Prozess ist wünschenswert, aber nicht immer realisierbar. Teilweise müssen Prozesse verfeinert werden, zum Beispiel bezüglich bestimmter Sicherheitsaspekte oder Transaktionen. Eine solche manuelle Verfeinerung ist auch nötig, wenn die automatische Suche nach Services nicht die gewünschten (oder keine) Ergebnisse liefert oder die gefundenen Schnittstellen nicht mit den vorliegenden Datenstrukturen umgehen können. In solchen Fällen kommen Prozess- und Datenmediatoren zum Einsatz.
 - *Prozessimplementierung (Deployment)*: Nach der Überführung in ein ausführbares Modell kann der Prozess auf einer Prozess-Engine bereitgestellt werden. Zusätzlich wird eine semantische Service Beschreibung des gesamten Prozesses erstellt. Somit wird der Gesamtprozess als SWS zur weiteren Nutzung zur Verfügung gestellt und im SWS Repository abgelegt.

Wie bereits im vorigen Abschnitt sollen im Folgenden untersucht werden, welche Ansätze für die einzelnen Anforderungen von Wetzstein et al. (2007) im Bereich der Implementierung semantischer Geschäftsprozesse existieren. Der Stand der Forschung der *semantischen Web Service Suche (Discovery)* wurde bereits ausführlich diskutiert. Daher sei hier auf Abschnitt 3.1 verwiesen.

Ergänzend zur Suche nach einzelnen SWS bietet die Theorie der Komposition die Möglichkeit, ein gewünschtes Ziel durch die Kombinationen mehrerer Service zu realisieren. Hierbei ist zunächst die Komposition während der Modellierung und während der Ausführung der Prozesse zu unterscheiden. Die Komposition während der Modellierung erlaubt vor allem eine manuelle Überprüfung des Ergebnisses, welches das Problem fehlerhafter Kompositionen verringert. Auf eine Komposition zur Zeit der Ausführung wird in Abschnitt 3.2.3 eingegangen.

Die Grundlagen für die Komposition im Kontext der semantischen Geschäftsprozessmodellierung stellen Weber et al. (2007) dar. Neben einer Komposition zur Zeit der Modellierung gehen die Autoren dabei von einer einheitlichen Beschreibung der Artefakte (Daten) aus, da die Prozesse innerhalb einer geschlossenen Domäne existieren. Für

die Implementierung werden nach Weber et al. (2007) sechs Komponenten benötigt (siehe Abbildung 36): (1) eine Geschäftsprozessmodellierungsumgebung, (2) eine Kompositionskomponente, (3), eine „Discovery“-Komponente, (4) eine Mediationskomponente, (5) ein Repository für Mediatoren und (6) ein Repository für ausführbare Artefakte (zum Beispiel Services). Die Kompositionskomponente enthält neben einer Kernkomponente für die Durchführung des Kompositionsalgorithmus noch Möglichkeiten zum Einsatz verschiedener Validatoren (zum Beispiel für unternehmensinterne Regeln und Richtlinien).

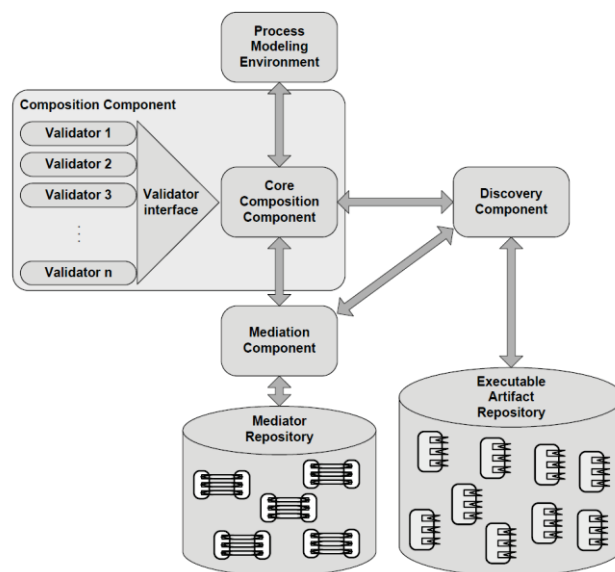


Abbildung 36: Komponentenarchitektur zur Komposition von semantischen Geschäftsprozessen, nach Weber et al. (2007), S. 58.

Ein möglicher Kompositionsalgorithmus beginnt mit der Identifizierung von bestehenden Prozessbestandteilen, welche unterstützt werden sollen. Anschließend kann es notwendig sein, die Prozessstruktur zu transformieren (zum Beispiel falls eine blockbasierte Sprache wie BPEL unterstützt werden soll, die Prozesse jedoch in Form von graphbasierten Sprachen vorliegen). Für die einzelnen Aktivitäten des Prozesses kann nun überprüft werden, ob sie mit Hilfe eines einzelnen Artefaktes (Service) unterstützt werden können. In diesem Schritt können bestehende Service Discovery Algorithmen adaptiert werden (siehe Abschnitt 3.2.2). Falls die Unterstützung durch ein einzelnes Artefakt nicht möglich ist, müssen zunächst Algorithmen zur Web Service Komposition eingesetzt werden. Die Autoren verweisen hier zum Beispiel auf Berardi et al. (2005). Etwaige Kompositionen müssen im Anschluss auf ihre Konsistenz überprüft werden. Im letzten Schritt werden die Implementierungen der Aktivitäten kombiniert und wiederum auf ihre Konsistenz überprüft. Falls keine Anpassungen durch den Nutzer nötig sind, das heißt eine *manuelle Verfeinerung*, ist die Prozesskomposition abgeschlossen. (Weber et al. (2007), S. 59-60)

Die Arbeiten von Born et al. (2007) und Weber et al. (2007) werden in Born et al. (2008) kombiniert. Der Fokus der Komposition liegt, wie bereits in Weber et al. (2007) dargestellt wurde, auf der Unterstützung eines komplexen Geschäftsprozesses und nicht nur in der Bereitstellung eines neuen, aus mehreren Web Services bestehenden Service. Für die Komposition wird auf die Annotation des Prozesses nach Born et al. (2007) zurückgegriffen. Die Beschreibung der Status vor und nach Aktivitäten durch Referenzen auf entsprechende Ontologiekonzepte erlaubt die Generierung von Zielbeschreibungen. Auf Basis dieser Zielbeschreibungen wird die Servicesuche und –komposition durchgeführt. Wie bereits in Born et al. (2007) wurde eine prototypische Implementierung in „Maestro“ integriert und somit die Machbarkeit und Skalierbarkeit des Ansatzes demonstriert. (Born et al. (2008))

Für die Implementierung und spätere Ausführung (siehe nächster Abschnitt) von semantischen Geschäftsprozessen ist eine Übersetzung in ein ausführbares Modell nötig. Bestehende Ansätze wie BPEL beschreiben zwar die Syntax der Prozesse, die Semantik wird allerdings nicht explizit ausgewiesen. Um diese Lücke zu schließen, präsentieren Nitzsche et al. (2007c) eine Ontologie auf Basis von BPEL, welche zusätzliche semantische Annotationen erlaubt. Das Konzept einer solchen Ontologie baut auf der Definition von BPEL in Form des XML Schemas der BPEL Spezifikation (Alves et al. (2007)) auf. Initial wird jedes XML Element aus BPEL in Form einer Ontologieklass in WSML abgebildet. Die Attribute der Elemente werden entsprechend als Attribute der WSML Klassen dargestellt. Auch die Kardinalitäten des XML Schemas werden soweit wie möglich übernommen. Die Verwendung von WSML macht es außerdem möglich Hierarchien einzuführen. Diese sind bereits in der textuellen Beschreibung vorgesehen, aber nicht im XML Schema definiert. Ein Beispiel hierfür ist das Konzept einer Aktivität (*Activity*), welche die Unterklassen einfache Aktivität (*BasicActivity*) und strukturierte Aktivität (*StructuredActivity*) beinhaltet. Für die Erweiterung der Semantik der Ontologie stehen Regeln in Form von Axiomen zur Verfügung. Diese können in der BPEL-Ontologie sinnvoll genutzt werden, zum Beispiel um zu überprüfen, ob bei der Schnittstellendefinition in Form sogenannter *partnerLinks* mindestens eine Rolle des Prozesspartners beschrieben wird. Zusätzlich wurden Konzepte zum Ausdruck der Reihenfolge von Elementen innerhalb eines Prozesses in WSML definiert, um einen strukturierten Ablauf des Prozesses garantieren zu können, wie er in BPEL vorgeschrieben wird. Die Umwandlung von BPEL nach WSML erlaubt weitere semantische Annotationen, zum einen aus der Modellierung, aber auch für die spätere Auswertung im Rahmen des semantischen Minings. (Nitzsche et al. (2007c))

Als Ergänzung einer BPEL Ontologie nach Nitzsche et al. (2007c) zur Nutzung von BPEL für den Aufruf von semantischen Web Services wurde BPEL4SWS (Nitzsche et al. (2007a)) entwickelt. Auf BPEL4SWS wird ihm Rahmen der Vorstellung einer Ausführungs-Engine für semantische Geschäftsprozessmodelle in Abschnitt 3.2.3 weiter eingegangen.

In den vorherigen Ansätzen wurde davon ausgegangen, dass die Implementierung der Funktionalität durch einen oder mehrere (Web) Services bereitgestellt wurde, welche während der Implementierungsphase gefunden und an die entsprechenden Prozesse ge-

bunden werden. Semantische Geschäftsprozessmodelle bieten allerdings auch die Möglichkeit, Services erst zur Laufzeit der Prozesse zu suchen und zu binden. Hierfür ist es nötig statt Services, entsprechend der gewünschten Funktionalität, Ziele an Aktivitäten oder Prozesse zu annotieren. Solche Ziele müssen präzise die gewünschte Funktionalität (zum Beispiel in Form von Vor- und Nachbedingungen sowie In- und Outputs) beschreiben, um während der Laufzeit des Prozesses vollautomatisch entsprechende Services finden zu können. Zusätzlich zu der Definition der Ziele muss eine passende Middleware zur Verfügung stehen, welche die Ziele interpretieren und die gefundenen Services an den Prozess binden kann. Ein solches Vorgehen wird als „späte Bindung“ (engl.: late binding) bezeichnet. (Pedrinaci et al. (2008), S. 549-550)

3.2.3 Ausführung semantischer Geschäftsprozesse

Nach der Implementierungsphase können semantische Geschäftsprozessmodelle auf einer Prozess-Engine bereitgestellt werden. Somit ist ein Prozess bereit zur Instanziierung und Ausführung und steht gleichzeitig als Service gekapselt zum Aufruf bereit. Damit ein solcher Prozess korrekt aufgerufen werden kann, sind verschiedene Ausführungsschichten nötig. Die Implementierungsschicht im sBPM wird hauptsächlich durch die semantische Prozess-Engine repräsentiert. Diese instanziiert Prozesse und Services und führt sie aus. Eine SWS-Infrastruktur steht als Registry-Schicht zur Verfügung und stellt Funktionalitäten zur Suche, Selektion und Aufruf von SWS bereit. Die Servicekonsumenten-Schicht beinhaltet Nutzeranfragen mittels definierter Ziele oder direktem Aufruf eines SWS. Eine solche Anfrage kann auch durch die sBP-Engine gestellt werden. (Wetzstein et al. (2007), S. 7)

Wetzstein et al. (2007) beschreiben die benötigten Funktionalitäten und Anwendungsfälle in der Ausführungsphase (Wetzstein et al. (2007), S. 8):

- *Ausführung semantischer Geschäftsprozesse:* Ein semantisches Prozessmodell wird ausgeführt, indem es instanziiert und anhand seines Kontrollflusses abgearbeitet wird. Die Instanziierung wird durch eine entsprechende Anfrage an den als SWS bereitgestellten Prozess ausgelöst. Zur Ausführung der Aktivitäten innerhalb des Prozesses werden entsprechende Anfragen an die SWS Infrastruktur delegiert.
- *Kommunikation mit der SWS Infrastruktur:* Um eine Aktivität zu unterstützen, nimmt die sBP-Engine die Rolle eines Servicekonsumenten ein. Aufgrund der angefragten Beschreibung oder des mit der Aktivität verknüpften Ziels kann ein passender SWS gefunden und aufgerufen werden.
- *Zielerreichung:* Die Funktionalität zum Umgang mit Zielanfragen wird von der SWS Infrastruktur bereitgestellt. Zielanfragen können durch Konsumenten gestellt werden und beschreiben die von einem Service (oder einer Servicekomposition) geforderte Funktionalität. Die Erfüllung eines solchen Ziels besteht aus zwei Phasen:

- (1) *SWS Discovery und Selektion*: In der ersten Phase wird auf Basis der Funktionalen Anforderungen ein Set an Services gesucht. Auf Basis nicht-funktionaler Anforderungen wird aus diesem Set ein Service ausgewählt.
- (2) *SWS Aufruf*: Nach der Selektion wird der entsprechende Service aufgerufen und das Ergebnis an den Servicekonsumenten weitergeleitet. Der Service kann in Form eines gekapselten semantischen Geschäftsprozesses vorliegen. In diesem Fall übernimmt die sBP-Engine die Ausführung. Alternativ existieren „konventionelle“ SWS, welche auf entsprechenden Applikationsservern ausgeführt werden.

Ein bestehender Ansatz zur *Ausführung semantischer Geschäftsprozesse* ist die Nutzung bestehender Prozess-Engines für BPEL und die Erweiterung dieser um eine Unterstützung von BPEL4SWS. Zum besseren Verständnis dieses Ansatzes soll im Folgenden zunächst eine kurze Vorstellung des BPEL4SWS-Konzeptes gegeben werden.

Wie der Name vermuten lässt, basiert BPEL4SWS auf BPEL4WS (Alves et al. (2007)). Ein Nachteil von BPEL4WS ist die starke Abhängigkeit von Web Service Schnittstellen. Innerhalb der Prozesslogik eines BPEL-Modells werden Web Service als Implementierung von Partnerschnittstellen fest definiert (*engl.*: hard-coded). Services, welche die gleiche Funktionalität mit einer anderen Schnittstelle bereitstellen, können somit nicht verwendet werden. Der Vorteil von SWS ist es allerdings die Funktionalität schnittstellenunabhängig definieren zu können. Eine schnittstellenunabhängige Variante von BPEL4WS ist BPEL^{light} (Nitzsche et al. (2007b)). BPEL^{light} verwenden Nitzsche et al. (2007a) als Basis für BPEL4SWS, um die Anbindung von SWS-Beschreibungen zu realisieren. In BPEL^{light} werden Konversationen (<conversation>) verwendet, um schnittstellenunabhängige <partnerLink> Elemente zu erlauben. Zusätzlich ist es notwendig, eine weitere Form von Interaktionsaktivität (<interactionActivity>) zu definieren, welche keine WSDL-Abhängigkeit besitzt. Im Gegensatz zu BPEL4WS 2.0 führt BPEL^{light} das <partner> Element wieder ein, um mehrere Konversationen mit einer Geschäftsentität definieren zu können. Neben den Anpassungen von BPEL^{light} erfordert BPEL4SWS die Möglichkeit semantischer Annotationen, vorzugsweise auf der Konversationsebene des Prozesses. Nitzsche et al. (2007a) verwenden hierfür das WS-PolicyAttachment (Bajaj et al. (2006)). Dabei wird zwischen „anbietenden“ und „konsumierenden“ Konversationen unterschieden. Eine Konversation, in welcher ein Partner einen Service des Prozesses verwendet, wird als „anbietend“ bezeichnet. Eine Konversation, welche den Service eines Partners innerhalb des Prozesses verwendet, wird als „konsumierend“ definiert. Nitzsche et al. (2007a) beschreiben weiterhin die Nutzung bestehender SWS-Beschreibungsansätze. Die Anbindung von OWL-S Services in BPEL4SWS ist möglich, allerdings nur unter Verwendung synchroner Kommunikationsmuster, da kein Partnermodell für eine asynchrone Kommunikation definiert ist. Für synchrone Serviceaufrufe kann ein OWL-S Service an eine Konversation annotiert werden und das Servicemodell beschreibt daraufhin die einzelnen Interaktionsaktivitäten der Konversation. WSMO hingegen ermöglicht eine asynchrone Kommunikation. Außerdem können sowohl Ziele als auch WSMO Web Services an Konversationen an-

notiert werden. Ziele werden für „konsumierende“ und Web Services für „anbietende“ Konversationen verwendet. Weiterhin kann die Choreographie in WSMO eine Sequenz der Interaktionsaktivitäten der jeweiligen Konversation beschreiben. Zusätzlich zu den semantischen Konzepten in WSMO und OWL-S wird SAWSDL in BPEL4SWS verwendet. SAWSDL dient hierbei vornehmlich als Transformationswerkzeug zwischen den Datenstrukturen in Form von Ontologiekonzepten und XML-Daten. Notwendig ist die Verwendung von *Lifting* und *Lowering* (siehe Abschnitt 2.2.3.3), da für die Abwärtskompatibilität von BPEL4SWS der gesamte Prozess als WSDL-Schnittstelle exponiert wird. (Nitzsche et al. (2007a))

Auf Basis von BPEL4SWS präsentieren van Lessen et al. (2009) eine Prozess-Engine für semantische Geschäftsprozesse. Im Gegensatz zu traditionellen Prozess-Engines muss eine semantische Ausführungsumgebung neben dem direkten Aufruf auch eine zielbasierte Kommunikation über eine semantische Middleware erlauben. Weiterhin sollen für beide Szenarien (direkt und zielbasiert) sowohl synchrone als auch asynchrone Aufrufe möglich sein. Für diese vier Aufrufmöglichkeiten präsentieren van Lessen et al. (2009) entsprechende Anwendungsfälle (van Lessen et al. (2009), S. 202 ff). Beispielfähig soll im Folgenden nur auf den semantischen asynchronen Aufruf eines BPEL4SWS-Prozesses eingegangen werden (siehe Abbildung 37).

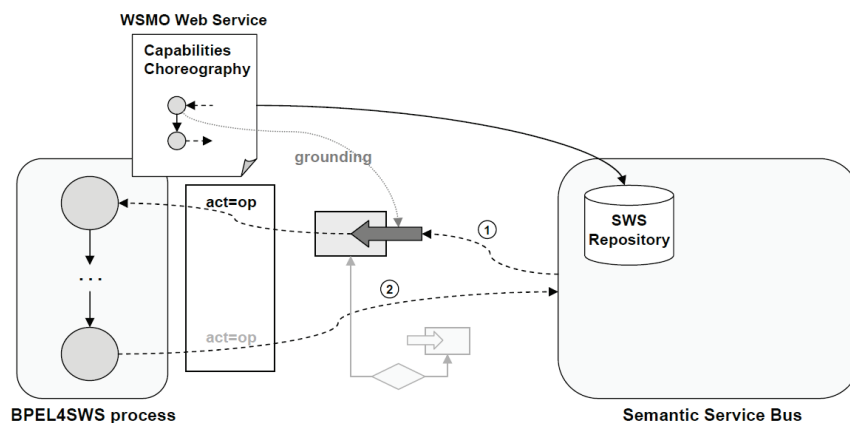


Abbildung 37: Semantischer asynchroner Aufruf eines BPEL4SWS-Prozesses, nach van Lessen et al. (2009), S. 204.

Bei einem asynchronen Aufruf wartet der Client nicht auf die Antwort des Prozesses, sondern liefert beim Aufruf einen WSDL-Endpunkt, an welchen nach Abschluss des Prozesses das Ergebnis geliefert werden kann. Das sogenannte *Grounding* verbindet dafür zum einen die Empfangsaktivität des Prozesses mit der aufrufbaren Operation. Zum anderen wird die darauffolgende Sendeaktivität des Prozesses mit dem vom Client gelieferten WSDL-Endpunkt verknüpft. Für den Aufruf mittels einer semantischen Middleware wird der Prozess zusätzlich in Form eines WSMO Web Service definiert. Dabei werden Schnittstelle und Funktionalität semantisch beschrieben und die Eingangsnachricht mit der entsprechenden Operation in der WSDL des Prozesses verknüpft. Im Falle

des asynchronen Aufrufes kann die ausgehende Nachricht noch nicht mit einer WSDL-Beschreibung verknüpfen, da diese erst zum Zeitpunkt des Aufrufs vom Client bereitgestellt wird. Wie in Abbildung 37 zu sehen, kann der semantische Service-Bus nach der Suche innerhalb des SWS-Repository das *Grounding* abrufen und eine entsprechende Anfrage an den definierten Endpunkt stellen. Die Umwandlung der semantischen in XML Daten (*lowering*) übernimmt hierbei der Service-Bus. Ein solcher semantischer Aufruf des Prozesses wird von der Engine erkannt (durch entsprechende Header-Informationen). Dies führt dazu, dass die Antwort des Prozesses durch die definierte Sendeaktivität keine WSDL-Operation benötigt, sondern eine Nachricht mit entsprechendem Kontext an den semantischen Service-Bus liefern kann (siehe ausgegrauter Teil der Abbildung 37). (van Lessen et al. (2009), S. 203-204)

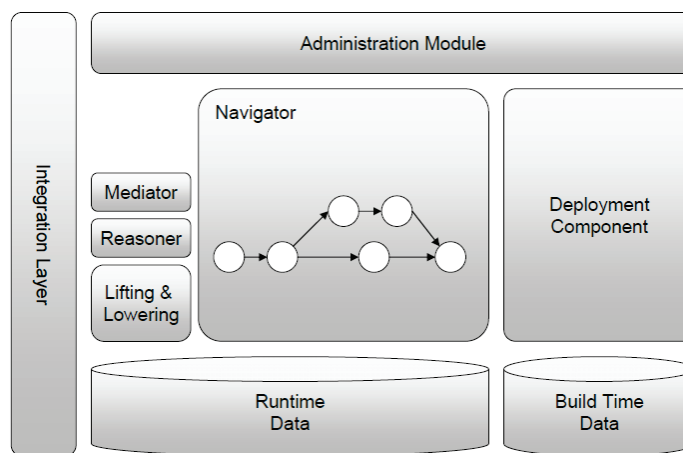


Abbildung 38: Architektur einer BPEL4SWS-Engine, nach van Lessen et al. (2009), S. 207.

Eine mögliche Architektur zur Realisierung einer solchen Ausführungsumgebung für semantische Geschäftsprozesse (semantische Business Process Engine - sBPE) ist in Abbildung 38 skizziert. Das Management und die Konfiguration werden über das Administrationsmodul gesteuert. Die Deployment-Komponente stellt die nötigen Artefakte zur Ausführung eines BPEL4SWS-Prozesses bereit (Beschreibung des Prozesses, WSDL-Daten, etc.). Validierte und kompilierte Prozessmodelle können anschließend in der „Build-Time“-Datenbank gespeichert werden. Daten für laufende Prozessinstanzen werden dahingegen in der „Runtime“-Datenbank gespeichert. Diese Instanzen beinhalten Referenzen auf die korrespondierenden Modelle in der „Build-Time“-Datenbank. Für die Kommunikation mit externen Services und Clients ist die Integrationsschicht zuständig. Anfragen und Antworten werden durch diese weitergeleitet. Ein Prozessnavigator ist notwendig, um auf die Prozessmodelle der „Build-Time“-Datenbank zugreifen und Instanzen derer starten zu können. Weiterhin speichert der Navigator die einzelnen Status der instanziierten Prozesse in der „Runtime“-Datenbank. Eine Reasoner-Komponente übernimmt Entscheidungen im Falle logischer Ausdrücke innerhalb des Prozesses (zum Beispiel bei Kontroll-Konnektoren oder Verbindungser-

eignissen). Falls Prozesse verschiedene Ontologien verwenden, kommt eine Mediator-komponente zum Einsatz, um eventuelle Diskrepanzen zu überbrücken. Die bereits erwähnten semantischen Inkompatibilitäten zwischen XML-Daten und Ontologiekonzepten werden mittels der „Lifting & Lowering“-Komponente gelöst. (van Lessen et al. (2009), S. 206-207)

Eine Implementierung der vorgestellten Architektur wurde durch van Lessen et al. (2009) auf Basis von Apache ODE⁵¹ durchgeführt, welche dafür in mehreren Bereichen erweitert werden musste. Für weitere Details der Implementierung sei hier direkt auf van Lessen et al. (2009) (S. 208ff) verwiesen.

Eine mögliche semantische Infrastruktur beziehungsweise semantische Middleware mit welcher *kommuniziert* (siehe Anforderungen nach Wetzstein et al. (2007)) werden kann ist die bereits erwähnte Referenzimplementierung von WSMX (siehe Abschnitt 3.1.2.3). Die Vorstellung der gesamten Implementierung von WSMX würde allerdings den Rahmen dieser Arbeit überschreiten. Weitere Details finden sich in Bussler et al. (2005) und Haller et al. (2005). Dort ist auch ersichtlich wie eine *Zielerreichung* durch *SWS Discovery*, *Selektion* und *Aufruf* realisierbar ist.

3.2.4 Analyse semantischer Geschäftsprozesse

Die Analyse von semantischen Geschäftsprozessen teilt sich zum einen in die Überwachung von laufenden Prozessen und zum anderen in die Auswertung (Mining) bereits ausgeführter Prozessinstanzen. Beide Ansätze stützen sich auf die Ereignisprotokolle, welche von Prozess-Engines bei der Ausführung der Prozesse angelegt werden. Im Vergleich zum klassischen BPM beinhalten diese Ereignisse im sBPM semantische Annotationen, sowohl zur Beschreibung der Ereignistypen als auch des Inhalts der Ereignisse (zum Beispiel in Form von Variablen). Auf Basis einer passenden Ontologie können die Ereignisse einfach klassifiziert und ausgewertet werden. (Wetzstein et al. (2007), S. 8)

Nach Wetzstein et al. (2007) sollte ein sBP Management System sowohl eine passive als auch eine aktive Überwachung der Prozesse unterstützen (Wetzstein et al. (2007), S. 9):

- *Passive Überwachung:* Hierbei können Analysten bestimmte Ereignisse „abonnieren“. Sobald die sBP-Engine ein solches Ereignis erzeugt, wird der Analyst benachrichtigt.
- *Aktive Überwachung:* In der aktiven Überwachung sucht ein Analyst nach konkreten Informationen in den Ereignisprotokollen. Dabei können auch weitere Anfragen an die sBP-Engine gestellt werden, um die Suche zu verfeinern. Solche Anfragen profitieren von den semantischen Annotationen der Ereignisse und liefern implizites Wissen durch Reasoning-Mechanismen.

⁵¹ Apache Orchestration Director Engine (ODE), ode.apache.org

Nach Celino et al. (2007) können die Daten einer BP-Engine als Grundlage für Prozess-Mining und sogenanntes *Reverse Business Engineering* (RBE) verwendet werden. Da beide Konzepte aktiv von einer semantischen Unterstützung profitieren können, werden sie kurz definiert und mögliche semantische Erweiterungen erklärt.

Im Rahmen des Prozess-Minings werden automatisch relevante Informationen über die bereits ausgeführten Prozesse erhoben. Sie basieren auf den Protokolldaten (siehe oben), stellen dafür aber auch bestimmte Anforderungen an diese. Zum einen müssen Prozesse innerhalb der Protokolle eindeutig identifizierbar sein. Zum anderen muss die Reihenfolge der Aktivitäten innerhalb der protokollierten Prozessinstanzen erkennbar sein. Anwendung findet das Prozess-Mining unter anderem in der Darstellung des Ablaufs einer Gruppe an Prozessen, der sozialen Netzwerke bei der Übergabe von Aufgaben innerhalb der Prozesse oder in der Engpassanalyse. RBE hingegen ist eine Methode, welche spezifisch auf die Analyse von ERP-Systemen abzielt. Die gesammelten Daten über die bestehenden Prozesse innerhalb der Systeme werden verwendet, um eine Ist-Prozess- und Systemlandschaft zu erheben. Im Vergleich mit einer Soll-Landschaft können somit mögliche Lücken und Bereiche für Verbesserungen identifiziert werden. (Celino et al. (2007), S. 45-46)

Zur Erweiterung von Prozess-Mining und RBE mit Semantik und somit höherer Automatisierung durch Maschinenlesbarkeit schlagen Celino et al. (2007) einen dreistufigen Ansatz vor. Zunächst sollen Ontologien erstellt werden, welche die Bedeutung der einzelnen Bestandteile der Prozessmodelle (und somit der späteren Protokolle) abbilden können. Anschließend werden die Prozessmodelle mit den entsprechenden Konzepten der Ontologien angereichert. Aufbauend auf diesen semantischen Prozessmodellen können bestehende Methoden des Prozess-Minings und des RBE erweitert werden, um diese zusätzlichen Informationen auch nutzen zu können. Für ein semantisches RBE können zusätzlich zu den reinen Ontologiekonzepten auch vordefinierte geschäftsrelevante Fragen formuliert und vorgehalten werden. Dies erlaubt eine Analyse basierend auf unterschiedliche Szenarien und entsprechenden Fragen. (Celino et al. (2007), S. 46)

Sowohl Celino et al. (2007) als auch Wetzstein et al. (2007) beschreiben eine Anzahl von Techniken zur Umsetzung der semantischen Geschäftsprozessanalyse (Wetzstein et al. (2007), S. 9, Celino et al. (2007), S. 46-47):

- *Semantische Suche nach Prozessen*: Eine Suche auf den bestehenden Prozessdaten der Protokolle. Aus diesen werden Prozessmodelle abgeleitet, welche mit implementierten Prozessen verglichen werden können. Dies kann zum Beispiel durch die Erstellung von Subsumptionsbäumen der Ontologien in den Ereignisprotokollen ermöglicht werden, wobei bestehende Ansätze nur eine flache Betrachtung der Modelle erlauben.
- *Semantische Überprüfung von Übereinstimmungen*: Die definierten Prozessmodelle können mit den Modellen der Ereignisprotokolle verglichen werden. Etwaige Abweichungen können identifiziert und analysiert werden.
- *Semantische Auswertung von Organisationsdaten (organisatorisches Mining)*: Diese Auswertung funktioniert ähnlich wie die Suche nach Prozessen, zielt jedoch

vornehmlich auf die Auswertung (sozialer) Netzwerkstrukturen ab. Somit können Gruppen und Teams basierend auf bestimmten Aufgaben identifiziert werden. Bestehende Techniken in diesem Bereich verwenden vornehmlich String-basierte Vergleichsalgorithmen. Die Verknüpfung von Aktivitäten mit Konzepten einer Ontologie kann die Analyse der sozialen Strukturen der Prozesse weiter verbessern.

- *Semantische Leistungsanalyse*: Die Auswertung der semantischen Annotationen der Ereignisprotokolle kann verwendet werden, um etwaige Engpässe oder sonstige Möglichkeiten der Leistungsverbesserung zu erkennen. Hierbei können auch Verletzungen von Service Level Agreements identifiziert werden.
- *Semantische Audits*: Verschiedene Eigenschaften der annotierten Ereignisprotokolle können überprüft werden. Die (Wieder-)Verwendung von bestehenden Konzepten aus Ontologien für die Annotation von bestimmten Eigenschaften an die Prozesselemente erleichtert diesen Prozess ungemein.
- *Semantisches Reverse Business Engineering*: Der Fokus des semantischen RBE liegt auf einer Ontologie mit sogenannten Geschäftsfragen (oder Fachfragen, engl.: *business questions*), welche RBE generalisierbar und wiederverwendbar machen. Somit soll eine standardisierte und flexible Variante des RBE für unterschiedliche Systeme, Prozesse und Modellierungstechniken geschaffen werden.

3.2.4.1 Ontologische Grundlage der semantischen Geschäftsprozessanalyse

Wie bereits in den vorangegangenen Phasen des sBPM-Lebenszyklus spielen auch in der Analysephase die zu verwendenden Ontologien eine große Rolle. Pedrinaci und Domingue (2007) versuchen in ihrer Arbeit für unterschiedliche Akteure und auf unterschiedlichen Ebenen Ontologien für die semantische Geschäftsprozessanalyse zu schaffen.

Für die Entwicklung der entsprechenden Ontologien verwenden Pedrinaci und Domingue (2007) eine Dreiteilung der Geschäftsprozessanalyse nach Gordijn und Akkermans (2001): (1) die Fachebene, (2) die Prozessebene und (3) die IT-Ebene. Diese Dreiteilung gewinnt zudem an Komplexität, da innerhalb der Ebenen durchaus Unterebenen sowie vor allem unterschiedliche Sichten auf die Ebene existieren können. So kann die fachliche Analyse sich auf Individuen, Abteilungen oder auch die gesamte Organisation beziehen. In der Prozessebene können verschiedene Unterprozesse mit dem zu analysierenden Prozess verknüpft sein. Weiterhin existieren Verknüpfungen der einzelnen Ebenen untereinander. Zum einen können Entscheidungen auf der Fachebene weitreichende Auswirkungen auf die Prozess- und anschließend auf die IT-Ebene haben. Zum anderen besitzt eine ausgeführte (oder auch nicht ausgeführte) Aktivität auf IT-Ebene durchaus das Potential, den darüber liegenden Prozess und letztendlich die Fachebene zu beeinflussen. Allein diese „einfache“ Betrachtung der verschiedenen Ebenen der Geschäftsprozessanalyse macht deutlich, welche Anforderungen an die ontologischen Grundlagen dieses Bereiches gestellt werden. (Pedrinaci und Domingue (2007), S. 80)

Die Grundlage für die Analyse auf den unterschiedlichen Ebenen bilden Daten und Protokolle der verschiedenen Systeme eines Unternehmens, im Englischen auch als *audit trail* bezeichnet. Die meisten Systeme besitzen wenig bis gar keine direkte Möglichkeit der Auswertung und Überwachung dieser Daten. Hinzu kommt, dass Form und Semantik der Daten von System zu System unterschiedlich sind. Erste Versuche in der standardisierten Darstellung solcher Daten in Form von XML finden sich im ProM-Werkzeug durch die MXML (van Dongen und van der Aalst (2005)). Pedrinaci und Domingue (2007) argumentieren, dass eine Erweiterung dieser Daten um semantische Annotationen zum einen die Interoperabilität und Integration verbessern. Zum anderen legen Ontologien die Grundlage für automatisierte Schlussfolgerungen auf der entstehenden Wissensbasis der Ereignisprotokolle. Schlussendlich können durch Regeln und Axiome Prozesse automatisch überwacht und wenn nötig angepasst werden. (Pedrinaci und Domingue (2007), S. 81-82)

Die von Pedrinaci und Domingue (2007) entwickelte *Process Mining Ontology* basiert zunächst auf einer generellen Ontologie zur Darstellung von Ereignissen, welche durch IT-Systeme generiert werden. Diese *Events Ontology* (EVO) basiert auf MXML, welche auf Basis verschiedener Ereignisprotokolle realer IT-Systeme entwickelt wurde. Dies ermöglicht auch die spätere Anpassung des ProM-Werkzeugs zur Analyse von EVO-Protokollen. Das Kernkonzept in EVO ist das *Monitoring Event*, welches unter anderem durch Attribute wie Akteure (welche ein Ereignis generieren), Zeitpunkte (der Generierung und des Empfangs) sowie Eingangs- und Ausgangsdaten verfeinert wird. Spezialisiert werden *Monitoring Events* durch *Message* und *Process Events*. Diese beschreiben Ereignisse welche während des Lebenszyklus eines Geschäfts- oder IT-Prozesses auftreten können. Sie können weiter verfeinert werden, um zum Beispiel Ereignisse bezüglich bestimmter Aktivitäten, wie deren Abschluss oder Abbruch, zu beschreiben. (Pedrinaci und Domingue (2007), S. 82-84)

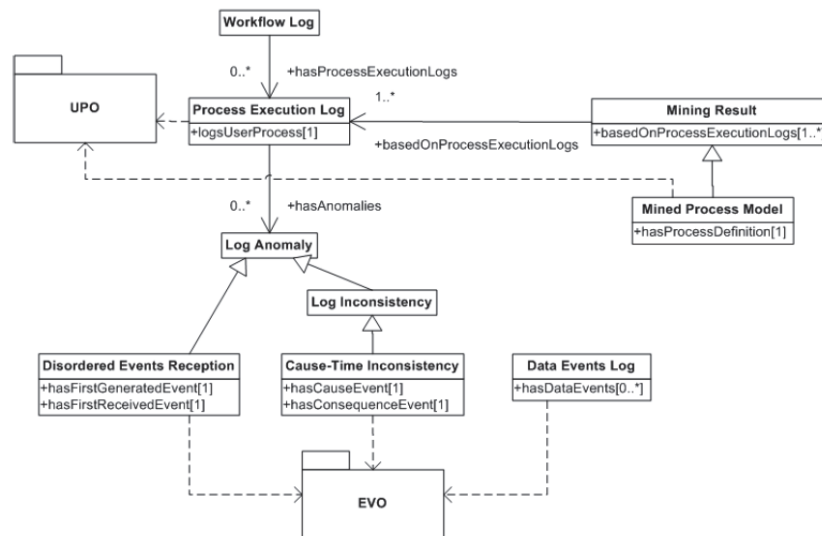


Abbildung 39: Die *Process Mining Ontology* als Aktivitätsdiagramm in UML, nach Pedrinaci und Domingue (2007), S: 85.

Aufbauend auf der EVO formalisiert die *Process Mining Ontology* (PMO) verschiedene Aspekte für die Auswertung von Geschäfts- und IT-Prozessen. Die Ontologie stellt Container für die Daten aus Prozessen, Workflows und Ereignissen, um verschiedene Perspektiven der Analyse zu unterstützen (siehe Abbildung 39). Zusätzlich beinhaltet die Ontologie Regeln und Axiome zum Umgang mit ungeordneten Ereignissen sowie inkonsistenten Zeitabhängigkeiten zwischen Ereignissen. Diese führen zur Erstellung entsprechender Konzepte innerhalb der Ontologie (siehe *Disordered Events Reception* und *Cause-Time Inconsistency* Abbildung 39). Außerdem ist es möglich die, Resultate von Analyse- beziehungsweise Prozess-Mining-Algorithmen in der Ontologie abzulegen. (Pedrinaci und Domingue (2007), S. 84-86)

3.2.4.2 Semantisches Prozess-Mining

Auf Basis der Erweiterung von Protokolldaten um semantische Informationen können Ansätze für ein semantisches Prozess-Mining implementiert werden.

Alves de Medeiros und van der Aalst (2009) präsentieren einen solchen Ansatz auf Basis des ProM-Werkzeuges⁵². Die Autoren betonen, dass die Protokolle über die ausgeführten Prozesse zwar viele Fragen bereits beantworten können (zum Beispiel *Welche Engpässe existieren?* oder *Wie ist die Verteilung der Fälle innerhalb eines Prozesses?*), diese aber an die Beschriftung (*engl.*: labels) der Prozesse gekoppelt sind. Schlussfolgerungen über

⁵² www.processmining.org/prom

die Labels sind nicht möglich, beziehungsweise nur, wenn manuell die einzelnen Prozesse zum Beispiel einem bestimmten Konzept zugeteilt werden. Die Suche im Prozess-Mining soll daher um semantische Annotationen erweitert werden. Bei einer solchen Erweiterung ist es wichtig, dass die Daten auch weiterhin von traditionellen Prozess-Mining Ansätzen ausgewertet werden können. (Alves de Medeiros und van der Aalst (2009), S. 37-38)

Die Evolution des Prozess-Minings durch semantische Annotation wird von Alves de Medeiros und van der Aalst (2009) als Transition von Label-basierter Analyse zu einer Konzept-basierter Analyse bezeichnet. Beispielhaft werden von den Autoren drei Ontologien zur Annotation eines Prozesses zur Reparatur von Telefonen definiert: (1) eine Aufgaben-Ontologie (*TaskOntology*), (2) eine Rollen-Ontologie (*RoleOntology*) und (3) eine Ausführer-Ontologie (*PerformerOntology*). Innerhalb der Ontologien können eine Reihe von Subsumptionen der Konzepte definiert werden, zum Beispiel kann eine Aufgabe vom System oder einer Person der Ausführer-Ontologie erledigt werden. Die Verknüpfung der Prozess-Protokolle mit Konzepten dieser Ontologien ist in verschiedenen Szenarien sinnvoll (Alves de Medeiros und van der Aalst (2009), S. 68-73):

- *Inspektion der Protokolle:* Eine Inspektion gibt einen Überblick über die Elemente der Protokolle. Mit Hilfe von annotierten Konzepten können unterschiedliche semantische Perspektiven eingenommen werden. Gruppierungen und Schlussfolgerungen über die Prozesse sind so auch möglich (zum Beispiel durch Axiome, die identifizieren, welche Prozesse abgeschlossen sind).
- *Säuberung von Protokollen:* Mit Hilfe von Filtern können Protokolle gesäubert beziehungsweise mit relevanten Daten erweitert werden. Filter auf Basis von Konzepten sind generischer einsetzbar als Filter auf Basis von Labels. So können semantische Filter auch über mehrere Prozesse wiederverwendet werden.
- *Verwendung in Suchalgorithmen:* Plugins von ProM erlauben den Einsatz unterschiedlicher Suchalgorithmen. Bei der Suche und Analyse von Kontrollflüssen der Prozesse können semantische Annotationen verwendet werden, um verschiedene Ebenen des Prozessflusses auf Basis der Subsumptionen der Konzepte darzustellen. Dies erlaubt eine Abstraktion und somit eventuell eine weniger komplexe Darstellung des Prozesses zur weiteren Analyse. Zur Analyse bestimmter Fälle (*engl. cases*) erlauben Annotationen die Untersuchung der häufigsten Pfade bezogen auf ein oder mehrere Konzepte.
- *Analyse von Konformität und Performanz der Prozesse:* Neben den Protokollen werden bei dieser Analyse auch explizite Modelle der Prozesse benötigt. Diese müssten im Rahmen der semantischen Analyse auch Verbindungen zu Konzepten von Ontologien besitzen (zum Beispiel durch die semantische Geschäftsprozessmodellierung in Abschnitt 3.2.1). Die Konformität der Protokolle mit den modellierten Prozessen kann anschließend auf Ebene der Konzepte durchgeführt werden. Weiterhin kann die Performanz der Prozesse in Bezug auf bestimmte Konzepte untersucht werden (zum Beispiel einzelnen Rollen oder Aufgabentypen).

Als Basis für die vorgestellten Szenarien des semantischen Prozess-Minings definieren Alves de Medeiros und van der Aalst (2009) abschließend eine semantische Erweiterung der in ProM verwendeten Mining-XML (MXML). Diese „Semantically Enhanced MXML“ (SA-MXML) erlaubt die Annotation von Konzepten innerhalb einer MXML-Datei durch Referenzierung externer Ontologien. So bleibt das Format abwärtskompatibel zu MXML.

3.3 Ontology Matching, Alignment und Integration

Der folgende Abschnitt knüpft an die Einführung zu Ontology Matching und Alignment aus Abschnitt 2.3 an. Zunächst werden bestehende Matching Techniken und Algorithmen klassifiziert und erläutert. Dies bildet die Grundlage für einen Überblick über den Stand der Forschung und Werkzeuge für das Matching und Alignment von Ontologien. Abschließend werden Techniken zur Evaluierung dieser Ansätze vorgestellt, welche im Verlauf der Arbeit auch auf den eigenen Ansatz anwendbar sind.

3.3.1 Ontology-Matching-Techniken und Strategien

Wie bereits in der Begriffsklärung in Abschnitt 2.3 erläutert bildet das Matching von Ontologien die Grundlage für deren Alignment. Daher wird im Folgenden nur von Strategien und Techniken für das Matching gesprochen, welche aber auch für das Alignment verwendet werden können. Grundlage des folgenden Abschnitts ist die Kategorisierung von Matching Techniken nach Euzenat und Shvaiko (2007) (S. 61ff). Ein Überblick über die Kategorisierung gibt die Abbildung 13 (S. 37). Dabei wurden bereits die Dimensionen der Klassifikation auf Basis von *Granularität* und *Art des Inputs* der einzelnen Techniken erläutert. Weiterhin wurde die *Granularität* zusätzlich unterteilt in Techniken zur Analyse von *Struktur* oder *Elementen* sowie welche Form der Interpretation die jeweilige Technik verwendet: *semantisch*, *syntaktisch* oder *extern*. Die *Art des Inputs* kann zusätzlich auf zwei Ebenen unterteilt werden. Die erste Ebene unterteilt die Techniken nach Daten, welche als Input verwendet werden können. Dies sind Strings (*terminologisch*), Strukturen (*strukturell*), Modelle (*semantisch*) oder Instanzen (*extensional*). Die zweite Ebene der *Art des Inputs* stellt zusätzliche Kategorien zur Einteilung der ersten Ebene zur Verfügung. *Terminologische* Techniken können auf Strings basieren oder auf der Interpretation dieser Strings als linguistische Objekte (*linguistisch*). *Strukturelle* Techniken betrachten entweder die interne Struktur der Entitäten (*intern*) oder die Relationen der Entitäten untereinander (*relational*). (Euzenat und Shvaiko (2007), S. 64-66)

In den folgenden Abschnitten wird diese Klassifikation im Detail dargestellt und einige Grundtechniken des Matchings erläutert. Abschließend werden verschiedene Strategien zum Einsatz dieser Techniken erläutert. Die Struktur wird dabei von Euzenat und Shvaiko (2007) übernommen und inhaltlich um weitere Quellen erweitert.

3.3.1.1 Klassifikation von Ontology-Matching-Techniken

Die Mehrdimensionalität der Klassifikation von Euzenat und Shvaiko (2007) erlaubt zwar eine detaillierte Analyse der Techniken, führt allerdings auch zu einer höheren Komplexität bei der Erläuterung der verschiedenen Basistechniken aus Abbildung 13. Daher werden die Basistechniken im Folgenden nur über die Unterscheidung nach *Granularität* in *Element-* und *Strukturebene* vorgestellt. Die folgenden Erläuterungen basieren auf einer Zusammenfassung von Euzenat und Shvaiko (2007) (S. 67-72).

Matching-Techniken auf der *Elementebene* untersuchen die Entitäten einer Ontologie unabhängig von anderen Entitäten oder deren Instanzen. Unterschieden werden können *String-basierte* und *sprachbasierte* Techniken sowie Techniken auf Basis bestimmter *Einschränkungen (constraints)* und *linguistischer Ressourcen*. Weiterhin existieren in diesem Bereich Techniken auf Basis der *Wiederverwendung von Alignments* sowie *Upper-level* und *domänenspezifischer Ontologien*.

String-basierte Techniken setzen auf die Analyse von Namen und Beschreibungen der Entitäten einer Ontologie. Dabei wird davon ausgegangen, dass je ähnlicher die Strings (also die Verkettung von Buchstaben des Alphabets) zweier Konzepte sind, desto ähnlicher sind sich auch die Konzepte selbst. Zum Vergleich von Strings wird oft auf Distanzberechnung zurückgegriffen. Die Distanz zwischen zwei Strings kann zum Beispiel durch eine reale Zahl ausgedrückt werden, wobei eine niedrigere Zahl auf eine höhere Ähnlichkeit der Konzepte hindeutet. Auf die Vor- und Nachteile dieser Technik und beispielhafte Algorithmen wird im Rahmen der Grundtechniken in Abschnitt 3.3.1.2 eingegangen.

Sprach-basierte Techniken behandeln Namen und Beschreibungen von Konzepten als Wörter einer natürlichen Sprache, zum Beispiel Deutsch oder Englisch. Sie verwenden Algorithmen zum Umgang mit natürlichen Sprachen, sogenanntes *Natural Language Processing (NLP)*, um morphologische Besonderheiten der Wörter in ihrer jeweiligen Sprache zu analysieren. Sprach-basierte Techniken werden oft als Vorstufe zu String-basierten Techniken oder einer lexikalischen Suche (im Rahmen *linguistischer Ressourcen*) genutzt. Als grundlegende Techniken können sie aber auch alleinstehend eingesetzt werden.

Constraint-basierte Techniken basieren auf Algorithmen, welche den zu untersuchenden Entitäten bestimmte Einschränkungen auferlegen. Diese können zum Beispiel anhand von Typen, Kardinalitäten, Multiplizitäten, Attributen oder Schlüsseln definiert werden.

Andere Techniken nutzen (externe) *linguistische Ressourcen* wie Lexika oder domänen-spezifische Thesauri, um die Wörter der Namen und Beschreibungen von Konzepten einer Ontologie anhand ihrer sprachlichen Verknüpfungen zu untersuchen. So können zum Beispiel Ähnlichkeiten und Unterscheidungen auf Basis von Synonymen und Homonymen identifiziert werden. Das bereits erwähnte WordNet (Miller (1995)) ist eine solche externe *linguistische Ressource*.

Techniken auf Basis der *Wiederverwendung von Alignments* nutzen die Resultate vorangegangener Matching-Verfahren, um bessere Ergebnisse bei der Ähnlichkeitsberechnung der gleichen oder zumindest ähnlicher Ontologien zu erzielen. Dies ist vor allem bei Alignments innerhalb der gleichen Domäne sinnvoll. Weiterhin ist ein Einsatz dieser Techniken auch bei einem Vergleich sehr großer Ontologien sinnvoll. In diesen Fällen kann das Problem des Matchings in Teilprobleme zur Untersuchung der Ontologiefragmente aufgespalten werden. Eine solche Wiederverwendung der Resultate vorangegangener Matching-Verfahren wurde bereits im Rahmen des Schema-Matchings durch Rahm und Bernstein (2001) (S. 341-342) vorgeschlagen und kann auch bei einem Vergleich von Ontologien eingesetzt werden.

Eine weitere Technik auf der *Elementebene* des Matchings nutzt bestehende Ontologien zur Kalkulation von Ähnlichkeiten der Konzepte. Dies können zum einen „höhere“ Ontologien, sogenannte Upper Level Ontologies (ULO) sein, welche generische Konzepte auf einer höheren Ebene als die zu vergleichenden Ontologien bereitstellen. Beispiele für ULOs sind die Suggested Upper Level Ontology (SUMO) (Niles und Pease (2001)) und die Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) (Gangemi et al. (2003)). Zum anderen können domänenspezifische Ontologien als externe Ressourcen für das Matching-Verfahren genutzt werden. Dies erlaubt das Matching von fachspezifischen Konzepten, zum Beispiel in der Medizin, Biologie oder anderen Naturwissenschaften.

Matching-Techniken auf der *Strukturebene* untersuchen die Beziehungen der Entitäten, Konzepte und Instanzen, einer Ontologie. Unterschieden werden können *Graphen-basierte*, *Taxonomie-basierte* und *Modell-basierte Techniken* sowie die Verwendung von *Struktur-Repositories*. Weiterhin können bestimmte *Techniken der Datenanalyse* und *Statistik* eingesetzt werden.

Graphen-basierte Techniken verwenden Algorithmen zur Untersuchung der Ontologien in Form beschrifteter Graphen. Eine Ähnlichkeit zwischen zwei Knoten der Graphen (Ontologien) kann zum Beispiel durch deren Position im Gesamtgraph ermittelt werden. Dabei werden die benachbarten Knoten in den Vergleich mit einbezogen unter der Annahme, dass zwei Knoten ähnlich sind wenn ihre Nachbarknoten auch eine gewisse Ähnlichkeit aufweisen und umgekehrt.

Taxonomie-basierte Techniken verwenden auch Algorithmen der Graphentheorie, allerdings nur im Hinblick auf die Spezialisierung der Konzepte einer Ontologie. Es wird davon ausgegangen, dass bestimmte „ist-ein“ Verknüpfungen der Begriffe auf eine Ähnlichkeit der verknüpften Entitäten hinweisen.

Modell-basierte Techniken nutzen die sogenannte modelltheoretische Semantik. Es wird davon ausgegangen, dass zwei gleiche Entitäten die gleichen semantischen Interpretationen besitzen. Dies kann zum Beispiel mittels Reasoning-Methoden der Beschreibungslogik überprüft werden. Somit interpretieren Techniken dieser Klasse Eingangsdaten in Bezug auf ein semantisches Modell. Ein entsprechendes Modell muss dafür ex-ante vorhanden sein.

Zusätzlich zu den bereits erwähnten Techniken können *Struktur-Repositories* zur Ähnlichkeitsbestimmung eingesetzt werden. In diesen werden Ontologien und Ontologiefragmente paarweise mit entsprechendem Ähnlichkeitskoeffizienten vorgehalten. Die Idee bei der Nutzung bestehender Ähnlichkeiten ist, dass der Vergleich der Ausgangsontologien mit den gespeicherten Strukturen weniger aufwendig ist als ein direkter Vergleich. Dabei können auch Meta-Daten der Strukturen verwendet werden, wie zum Beispiel die Anzahl der Knoten, der Name der Struktur (Ontologie) oder Wurzel sowie der längste Pfad der Ontologie. Ein Ansatz zur Nutzung von *Struktur-Repositories* präsentieren Rahm et al. (2004).

Techniken der Datenanalyse und der *Statistik* nutzen eine repräsentative Auswahl an Daten, um Regelmäßigkeiten und Abweichungen zu finden, welche für den gesamten Datensatz verwendbar sind. Repräsentative Aussagen über Ontologien können helfen, die Distanz zwischen den Konzepten zu berechnen oder deren Verteilung (wobei Konzepte mit ähnlicher Verteilung innerhalb der Ontologien korrelieren).

3.3.1.2 Grundtechniken des Matchings

Basierend auf der Art des Inputs der Matching-Techniken in Abbildung 13 (S. 37) (*terminologisch, strukturell, extensionell, semantisch*) werden im Folgenden einige Grundtechniken des Matchings vorgestellt. Der Abschnitt basiert vornehmlich auf einer Zusammenfassung des Kapitels 4 aus Euzenat und Shvaiko (2007) (S. 73ff) mit zusätzlichen Quellen.

Terminologische Methoden basieren vornehmlich auf den Namen der Entitäten einer Ontologie und ihrer Beschreibungen. Viele analysieren diese auf Basis einer oder mehrerer Strings, also einer Aneinanderreihung von Buchstaben und weiterer Zeichen (z.B. Trennungszeichen). Strings wiederum bestehen aus sogenannten Substrings, welche zusammengesetzt den Ursprungsstring ergeben. Bei einem Vergleich der Entitäten ist zu entscheiden, ob und wenn ja welche Substrings analysiert werden sollen.

Problematisch bei der Analyse von Namen und Beschreibungen sind Mehrdeutigkeiten der natürlichen Sprache. Dabei ist zwischen *Synonymen* und *Homonymen* zu unterscheiden. *Synonyme* bezeichnen die gleiche Entität mit unterschiedlichen Wörtern (wenn auch nur in bestimmten Kontexten), wie zum Beispiel *Kunde* und *Klient*. *Homonyme* verwenden das gleiche Wort, um unterschiedliche Entitäten zu beschreiben, wie zum Beispiel *Bank* (Parkbank gegenüber einem Finanzinstitut). Synonyme und Homonyme können auch durch die Verwendung unterschiedlicher Sprachen oder syntaktischer Variationen (wie Abkürzungen) entstehen. Diese Mehrdeutigkeiten können sowohl innerhalb einer als auch über mehrere Ontologien hinweg entstehen. Terminologische Vergleiche können also rein stringbasiert sein oder eine sprachliche Interpretation der (Sub-) Strings betrachten.

Vor dem Vergleich von Strings auf einer syntaktischen Ebene können Techniken der *Normalisierung* eingesetzt werden. Diese sollen die anschließenden Ergebnisse der Vergleichsmethoden verbessern, indem zum Beispiel Buchstaben in Kleinbuchstaben umgewandelt, „leere“ Bestandteile (Leerzeichen, Tabulatoren etc.) entfernt oder Interpunk-

tion innerhalb des Strings unterdrückt werden. Die *Normalisierung* ist mit Vorsicht einzusetzen, da sie meistens abhängig von Sprache und Reihenfolge (der Art der Normalisierung) ist und wichtige Informationen durch sie verloren gehen können.

Im Folgenden wird eine Auswahl an stringbasierten Techniken mit möglichen formalen Metriken zur Operationalisierung präsentiert (Euzenat und Shvaiko (2007), S. 77-83):

- *Gleichheit von Strings*: Falls zwei Strings identisch sind, ist diese Metrik wahr, andernfalls falsch. Sinnvoll ist ein solcher Vergleich am ehesten nach der Normalisierung. Zusätzlich kann berechnet werden, wie unterschiedlich die Strings sind. Hierfür kann die *Hamming-Distanz* (Hamming (1950)) eingesetzt werden, welche die Anzahl an Positionen zählt, an welchen sich die Strings unterscheiden.
- *Substring Test*: Es kann überprüft werden, ob ein String ein Substring eines zweiten Strings ist. Diese einfache Überprüfung kann erweitert werden, indem geprüft wird, welchen Anteil der größte gemeinsame Substring am Gesamtstring hat. Zur Berechnung wird oft die *N-Gramm* Ähnlichkeit eingesetzt. Dabei werden die Strings in Sequenzen von *N* Buchstaben unterteilt und überprüft, wie viel Übereinstimmungen dieser N-Gramme existieren. Die Methode ist vor allem bei fehlenden Buchstaben effizient einzusetzen.
- *Edit Distance*: Die sogenannte *Edit Distance* beschreibt die minimalen Kosten oder auch die Anzahl an Operationen, um einen String in den anderen umzuwandeln. Der Einsatz lohnt sich vor allem, um Rechtschreibfehler ausschließen zu können. Eine weit verbreitete Methode zur Berechnung der *Edit Distance* ist die *Levenshtein Distanz* (Levenshtein (1965)). Sie beschreibt die minimale Anzahl an Einfügungen, Löschungen und Ersetzungen, die nötig sind, um einen String in den anderen umzuwandeln. Diese Methode kann zudem angepasst werden, indem höhere Kosten für bestimmte Operationen (zum Beispiel für das Einfügen und Löschen) verwendet werden.
- *Nicht-symmetrische Distanzmessung*: Ähnlich der *Edit Distance* wurde basierend auf der Anzahl und der Nähe gemeinsamer Buchstaben zweier Strings die *Jaro-Metrik* (Jaro (1989)) entwickelt. Diese wurde von Winkler (1999) um eine höhere Gewichtung für Strings mit längeren gemeinsamen Präfixen erweitert.
- *Token-basierte Distanzmessung*: Aus dem Bereich des Information Retrieval können Methoden zur Aufteilung längerer Wörter oder Texte in *Token* auch zum Vergleich von Ontologien verwendet werden. Zusätzlich können Entitäten auch mit den Wörtern aus der Beschreibung (soweit vorhanden) verglichen werden. Das erlaubt den Vergleich der Entitäten mit diesen Multisets von Strings. Stellt man die Multisets als Vektoren dar, welche zu den Wörtern auch die Anzahl ihres Auftretens annotieren, so kann man diese Vektoren vergleichen. Eine Möglichkeit zum Vergleich ist zum Beispiel die Messung des Kosinus zwischen den zwei Vektoren. Eine weitere Möglichkeit zum Vergleich von Multisets ist die Messung der Frequenz der Strings im Multiset im Vergleich zur Gesamtheit an Sets (*Term Frequency – Inverse Document Frequency*). Dies ist keine direkte Ähnlichkeitsmessung, erlaubt aber eine Aussage über die

Relevanz bestimmter Begriffe bei der Beschreibung der Entität. Weniger relevante Begriffe können dann bei einem Vergleich wenig oder gar keine Beachtung finden.

- *Pfadvergleich*: Entitäten können nicht nur anhand ihrer eigenen Namen sondern auch der Namen ihres Pfades innerhalb der Ontologie verglichen werden. So ist es zum Beispiel möglich den Namen eines Konzeptes und alle seine Superkonzepte zu verketteten. Die entstehenden Sequenzen sind durch die bereits erwähnten String-Distanzmessungen vergleichbar. Wichtig dabei ist eine Abschwächung der Superkonzepte einzuführen, welche abnimmt je „näher“ man dem zu vergleichenden Konzept kommt. In einer Verkettung **Buch** → **Sachbuch** → **Biografie** können beispielweise **Buch** mit 0,7 und **Sachbuch** mit 0,3 in ihrer Wertung geschwächt werden.

Die rein *stringbasierten Techniken* sind vor allem dann sinnvoll einsetzbar, wenn ähnliche Wörter verwendet werden, um die gleichen Konzepte zu beschreiben. Dies kann vor allem beim Einsatz von synonymen Strings mit wenigen Übereinstimmungen der Buchstaben zu Problemen führen.

Die Interpretation von Strings als eine Reihenfolge von Buchstaben kann erweitert werden um die Interpretation der Strings als Wörter einer natürlichen Sprache. Auch bei Konzepten aus zusammengesetzten Wörtern oder längeren Texten (zum Beispiel in der Beschreibung des Konzeptes) müssen relevante Begriffe der Sprache identifiziert werden können. *Sprachbasierte Techniken* des Ontologie-Matchings basieren daher meist auf dem bereits erwähnten Natural Language Processing, um sinnvolle Begriffe aus den Konzeptnamen und -beschreibungen zu extrahieren. Es kann weiter unterschieden werden zwischen rein algorithmischen Ansätzen (*intrinsische Methoden*) und Ansätzen basierend auf externen Quellen (*extrinsische Methoden*).

Intrinsische Methoden setzen auf eine Normalisierung der Begriffe unter Verwendung der Regeln der natürlichen Sprache. Verschiedene Varianten eines Begriffes können existieren, welche während einer sprachbasierten Ähnlichkeitsanalyse zu berücksichtigen sind. Zu unterscheiden ist zwischen morphologischen (Variation der Form und Funktion basierend auf dem gleichen Ursprung des Worts), syntaktischen (Variation der grammatikalischen Struktur), semantischen (Variation eines Teils des Begriffes, zum Beispiel mit Hyperonymen (Oberbegriffen) oder Hyponymen (Unterbegriffen)) und multilingualen Varianten (Variation der Sprache).

Für die Normalisierung können verschiedene Methoden eingesetzt werden (Euzenat und Shvaiko (2007), S. 84-85):

- *Zerlegung in Tokens*: Wie bereits erwähnt, können Strings in eine Sequenz von Tokens eingeteilt werden, unter anderem durch Eliminierung von Trenn- und Leerzeichen. Die entstehenden Begriffe können daraufhin weiter normalisiert oder verglichen werden.
- *Lemmatisierung*: Die Strings der Tokens können auf eine Ursprungsform (oder Grundform) des Begriffs reduziert werden. Die morphologische Analyse eines

Begriffs zur Findung des Ursprungs wird als *Lemmatisierung* bezeichnet. Dies vereinfacht den Vergleich von unterschiedlichen Begriffen gleichen Ursprungs.

- *Extraktion von Begriffen*: Die Extraktion von Begriffen wird vor allem im Umgang mit (Fließ-)Texten eingesetzt. Dabei werden morphologisch ähnliche Textabschnitte identifiziert und Mustererkennung verwendet. Dies erlaubt die Identifizierung von Begriffen mit gleicher Bedeutung, benötigt allerdings längere Texte, um genaue Ergebnisse zu erzielen. Zum Einsatz einer solchen Technik siehe auch Cerbah und Euzenat (2001).
- *Eliminierung von Stoppwörtern*: Stoppwörter sind häufig vorkommende, für die Analyse eines Textes meistens irrelevante Wörter. Dies können Artikel, Konjunktionen, Präpositionen und ähnliches sein. Diese werden für das Matching des Textes eliminiert.

Nach der Normalisierung können die Begriffe oder auch Begriffsketten auf Basis der *stringbasierten Techniken* verglichen werden. Neben der Verwendung von *intrinsischen Methoden* werden zusätzliche sprachliche Ressourcen wie Lexika oder Wörterbücher für die Ähnlichkeitsbestimmung eingesetzt. Mögliche Ressourcen für diese *extrinsischen Methoden* sind (Euzenat und Shvaiko (2007), S. 86):

- *Lexika*: Ein Lexikon besteht aus einer Ansammlung von Wörtern einer natürlichen Sprache und deren Definitionen. Für ein Wort können allerdings mehrere Definitionen existieren.
- *Mehrsprachliche Lexika*: Diese Form von Lexika wird oft als Wörterbuch bezeichnet und liefert statt einer Definition die Äquivalenz des Wortes in einer anderen natürlichen Sprache. Dies ist vor allem für das Matching mehrsprachiger Ontologien sinnvoll.
- *Semantisch-syntaktische Lexika*: Semantische und semantisch-syntaktische Lexika beschreiben die Wörter in Form von Kategorien, zum Beispiel **Lebewesen** oder **Säugetiere**. Zudem können auch die Verbindungen von Verben und Adjektiven mit möglichen Substantiven dargestellt werden.
- *Thesauri*: Ein Thesaurus ist eine Form eines Lexikons, welches bestimmte Verbindungen zwischen den Wörtern herstellt. Solche Verbindungen können zum Beispiel Synonyme, Hyperonyme, Hyponyme und Antonyme (das Gegenteil eines Wortes, beispielsweise **Theorie** und **Praxis**) beschreiben. Das bereits mehrmals erwähnte WordNet (Miller (1995)) ist ein solcher Thesaurus.
- *Terminologien*: Eine Terminologie kann auch als Thesaurus für bestimmte Begriffe verstanden werden. Allerdings werden in einer Terminologie oft Formulierungen mehrerer Wörter, oft auch domänenspezifisch, beschrieben.

WordNet ist ein gutes Beispiel für die Nutzung einer externen Ressource im Bereich des Ontology Matchings. Miller beschreibt in seinen Arbeiten (Miller et al. (1990); Miller (1995)) eine lexikalische Datenbank, welche speziell für die Interpretation durch Maschinen konzipiert wurde. Entwickelt für die englische Sprache beinhaltet WordNet Substantive, Adjektive, Verben und Adverbien, organisiert in sogenannten Synonym-Sets (Miller (1995), S. 39). Bestimmte Wörter können natürlich in verschiedenen syn-

taktischen Kategorien auftauchen, wie zum Beispiel **back**. Neben den Synonym-Sets (kurz SynSet) werden semantische Verbindungen der Wörter beschrieben: Antonyme, Hyperonyme, Hyponyme, Meronymie (Teile eines Ganzen), Holonyme (Gegenteil eines Meronyms), Troponyme (Hypnome eines Verbs, eine bestimmte „Art“ etwas zu tun, zum Beispiel **Schlurfen** als Art des **Gehens**). Implikationen (Entailment) zwischen Verben sind auch in WordNet ausgedrückt (zum Beispiel: **schnarchen** impliziert **schlafen**). Eine Übersicht der semantischen Relationen in WordNet und Beispiele gibt Tabelle 2.

Semantic Relation	Syntactic Category	Examples
Synonymy (similar)	N, V, Aj, Av	pipe, tube rise, ascend sad, unhappy rapidly, speedily
Antonymy (opposite)	Aj, Av, (N, V)	wet, dry powerful, powerless friendly, unfriendly rapidly, slowly
Hyponymy (subordinate)	N	sugar maple, maple maple, tree tree, plant
Meronymy (part)	N	brim, hat gin, martini ship, fleet
Troponymy (manner)	V	march, walk whisper, speak
Entailment	V	drive, ride divorce, marry
<i>Note: N = Nouns Aj = Adjectives V = Verbs Av = Adverbs</i>		

Tabelle 2: Semantische Relationen in WordNet, nach Miller (1995), S. 40.

Externe Ressourcen wie WordNet können auf verschiedene Arten für das Matching von Entitäten in Ontologien eingesetzt werden, unter anderem (Euzenat und Shvaiko (2007), S. 88):

- die Ähnlichkeit von zwei Begriffen wird angenommen, wenn sie zu demselben SynSet gehören;
- die Distanz zwischen den Begriffen kann anhand ihrer Hyperonym-Relationen gemessen und als Indiz für die Ähnlichkeit verwendet werden;
- die Definition der Konzepte kann verwendet werden, um die Distanz zwischen verschiedenen SynSets zweier Begriffe zu messen.

Die Relationen innerhalb von WordNet können für das Matching auch in Form logischer Operatoren und Regeln ausgedrückt werden ($\sqsubseteq, \supseteq, =, \perp$). Mögliche Maßeinhei-

ten für die Messung von Ähnlichkeiten bei Synonymen sind (Euzenat und Shvaiko (2007), S. 89-90):

- *Direkte Ähnlichkeit der Synonyme*: Konzepte sind entweder synonym (1) oder nicht (0)
- *Ko-Synonyme*: Berechnung der Entfernung der Synonyme anhand der SynSets
- *Semantische Ähnlichkeit nach Resnik*: Nach Resnik (1995) kann bei der Ähnlichkeitsberechnung der Synonyme auch die wahrscheinliche Häufigkeit einer Instanz eines Konzeptes innerhalb eines zu untersuchenden Korpus (zum Beispiel eines längeren Textdokumentes) mit einbezogen werden. Dabei wird der Informationsgehalt der Konzepte gemessen und berechnet, wie viel Information die zu vergleichenden Konzepte gemein haben.
- *Informationstheoretische Ähnlichkeit*: Wie bei Resnik (1995) wird in anderen Ansätzen der Informationsgehalt gemessen, allerdings auf Basis gemeinsamer Hyperonyme der Konzepte (ohne einen Korpus).
- *Überschneidungen der Definitionen*: Die Definitionen der Konzepte innerhalb der externen Ressourcen können verwendet werden, um Überschneidungen zu berechnen, zum Beispiel mittels des bereits erwähnten Jaccard-Maßes (siehe S. 53ff).

Bei der Nutzung von Synonymen können aber auch Probleme entstehen, da die Interpretation der Konzepte erweitert wird. Dies führt teilweise zur Identifizierung ähnlicher Konzepte, erhöht aber gleichzeitig die Anzahl möglicher Homonyme und somit falsch-positiver Treffer.

Weiterhin soll auf die *strukturbasierten Techniken* des Ontologie-Matchings eingegangen werden. Diese untersuchen zum einen die *interne* und zum anderen die *externe* Struktur der Konzepte einer Ontologie.

Die Untersuchung der *internen* Struktur der Entitäten einer Ontologie basiert unter anderem auf deren Eigenschaften, die Attribute und Relationen der Eigenschaften, der Kardinalität und Multiplizität sowie der Transitivität oder Symmetrie der Eigenschaften. Für die Ähnlichkeitsbestimmung wird davon ausgegangen, dass Konzepte mit einer vergleichbaren internen Struktur potentiell gleich sind. Da aber ein reiner Strukturvergleich nicht immer akkurate Ergebnisse liefert, werden diese Techniken oft mit terminologischen Vergleichsmethoden kombiniert, welche auf den identifizierten Clustern der strukturellen Ähnlichkeitsanalyse aufbauen. (Euzenat und Shvaiko (2007), S. 92-93)

Zum Vergleich der Elemente der internen Struktur von Entitäten bieten sich mehrere Methoden an (Euzenat und Shvaiko (2007), S. 93-98):

- *Vergleich von Eigenschaften und Schlüsseln*: Ähnlich dem Vergleich von Datenbank-Schemata können bestimmte Klassen und Konzepte einer Ontologie anhand eindeutiger Schlüssel identifiziert werden. Falls die zu vergleichenden Konzepte in ihren Eigenschaften gleiche oder ähnlich strukturierte Schlüssel besitzen (wie zum Beispiel die Nummer des Personalausweises oder eine

ISBN⁵³), so kann dadurch auf eine gewisse Ähnlichkeit dieser Konzepte geschlossen werden.

- *Vergleich von Datentypen:* Zu dem Vergleich der Eigenschaften von Ontologiekonzepten gehört auch der Vergleich der Datentypen dieser Eigenschaften. Im Gegensatz zu den subjektiv entwickelten Konzepten der zu vergleichenden Ontologien besitzen Datentypen ein gewisses Maß an Objektivität. Dabei muss zwischen „elementaren“ Datentypen (wie **String**, **Integer**, **Boolean** etc.) und sogenannten Domänen unterschieden werden. Letztere beschreiben ein Subset bestimmter Datentypen, zum Beispiel durch die Verwendung von Wildcards (*) oder Datenbereiche.
Wichtig bei dem Vergleich von Datentypen ist es, dass sie meist nicht völlig disjunkt sind. Das heißt, dass bei der Ähnlichkeitsberechnung die Kompatibilität der Datentypen mit einbezogen werden muss.
- *Domänenvergleich:* Abhängig von der Art einer Entität (Klasse oder Eigenschaft) kann beschrieben werden, welcher Bereich von Datentypen oder Werten abgedeckt wird. Es ist möglich, auf Basis der Generalisierung der Domänen, ihrer Schnittmenge und ihrer Vereinigung ein Distanzmaß zu definieren (siehe hierzu zum Beispiel Valtchev (1999)). Diese Maß kann zudem mit der größtmöglichen Distanz der verglichenen Eigenschaften normalisiert werden.
- *Vergleich von Multiplizitäten und Eigenschaften:* Eigenschaften der Entitäten von Ontologien können Restriktionen bezüglich ihrer Kardinalität unterliegen. So kann neben einem Vergleich der Datentypen auch die damit verbundenen Multiplizitäten verglichen werden. Ein mögliches Maß für die Ähnlichkeit der Multiplizitäten berechnet sich zum Beispiel auf Basis von Intervallen von ganzen Zahlen (beispielsweise [0 4] und [1 20]) wie folgt: Differenz zwischen Minimum der Endpunkte und Maximum der Startpunkte geteilt durch die Differenz zwischen Maximum der Startpunkte und Minimum der Endpunkte.

Der Vergleich der internen Strukturen einer Ontologie erlaubt den Einsatz von einfach implementierbaren und effizienten Algorithmen. Allerdings ist bei diesen Ansätzen der Umfang an verwendbaren Informationen begrenzt und nicht immer eindeutig. Daher sollten Techniken des internen Strukturvergleichs immer mit weiteren Methoden des Matchings kombiniert werden.

Die *externe* Struktur der Entitäten einer Ontologie, auch *relationale* Struktur genannt, bietet zusätzliche Möglichkeiten der Ähnlichkeitsbestimmung. Zur Untersuchung dieser können Ontologien als Graphen dargestellt werden, deren Kanten mit den Namen der Relationen gewichtet sind. Auf diese Weise können Methoden der Graphentheorie und -optimierung eingesetzt werden. Die Techniken der Ähnlichkeitsberechnung hängen hierbei allerdings von Art der untersuchten Relationen in der Ontologie ab. Es

⁵³ International Standard Book Number

kann zwischen folgenden Strukturen und den damit verbundenen Ähnlichkeitsberechnungen unterschieden werden (Euzenat und Shvaiko (2007), S. 99-104):

- *Taxonomische Struktur*: Diese Strukturen basieren auf der Unterklassenrelation (**subClassOf**), welche eine grundlegende Relation fast aller Ontologien ist. Die meisten Ähnlichkeitsmaße in diesem Bereich basieren auf der Anzahl an Kanten zwischen den zu vergleichenden Klassen der Taxonomie (Ontologie). So können zum Beispiel Algorithmen zur Berechnung des kürzesten Pfades innerhalb der Graphen eingesetzt werden. Je nach Entfernung zueinander oder auch zur Wurzel der Taxonomie können die Kanten unterschiedlich gewichtet werden. Ein Ansatz hierzu präsentieren beispielsweise Wu und Palmer (1994), welche davon ausgehen, dass Konzepte nahe der Wurzel zwar nur wenige Kanten voneinander entfernt, konzeptionell aber oft stark unterschiedlich sind.
- *Mereologische Struktur*: Eine mereologische Struktur beschreibt die Verbindungen als Teil-von (**partOf**) Relationen. Diese Relationen können teilweise schwer erkannt werden, da sie zum Beispiel nur implizit über Eigenschaften der Klassen definiert sind. Falls eine mereologische Struktur gefunden werden kann, so können Ähnlichkeiten auf Basis gemeinsamer Teile berechnet werden.
- *Allgemeine Relationen*: Unabhängig von taxonomischen oder mereologischen Strukturen können Entitäten auf Basis aller ihrer möglichen Relationen miteinander verglichen werden. Dies können unter Umständen auch Verbindungen über die Verwendung gemeinsamer Eigenschaften sein. Falls ein Wissen über die Ähnlichkeit von Konzepten und deren Relationen bereits besteht, können daraus auch weitere Ähnlichkeiten abgeleitet werden. Solche Rückschlüsse können auch verwendet werden, wenn ein bestimmtes Set an Relationen aus Ontologie mit einem Set an Relationen in Ontologie Ähnlichkeiten aufweist. Dann ist es unter Umständen möglich, dass die Klassen der Domänen dieser Relationen auch ähnlich sind. Bei der Untersuchung allgemeiner Relationen muss allerdings auf die Vermeidung von zirkulären Beziehungen geachtet werden.

Bis zum jetzigen Punkt wurden nur Matching-Techniken besprochen, welche die abstrakten Entitäten einer Ontologie (zum Beispiel Klassen) untersuchen. Diese Techniken sind wichtig, da man nicht davon ausgehen kann, dass auch immer eine größere Anzahl an Instanzen in den zu vergleichenden Ontologien zur Verfügung steht. Falls jedoch Instanzen existieren, bietet dies eine sehr gute Möglichkeit zur Verbesserung der Matching Ergebnisse. Dies ist vor allem der Fall, wenn die Ontologien die gleichen Instanzen beziehungsweise Individuen besitzen.

Eine Berechnung der Ähnlichkeit anhand der Instanzen der Ontologien wird durch sogenannte *extensionelle Techniken* möglich. Dabei ist zu unterscheiden, ob die Techniken (1) exakt gleiche Instanzen der Ontologien benötigen, (2) gleiche Instanzen durch zusätzliche Techniken identifizieren müssen oder (3) auch mit unterschiedlichen Instanzen umgehen können (Euzenat und Shvaiko (2007), S. 105-107):

- *Vergleich anhand gleicher Instanzen:* Falls die zu vergleichenden Ontologien die gleichen Instanzen besitzen ist, dies der einfachste Fall für den Einsatz einer *extensionellen Technik*. Zum Einsatz kommt die Mengentheorie, um festzustellen, ob ein Set der Instanzen (A) das andere (B) beinhaltet ($A \cap B = A$), Teilmenge dessen ist ($A \cap B = B$), von ihm disjunkt ($A \cap B = \emptyset$) oder gleich ($A \cap B = A = B$) ist. Da ein absoluter Mengenvergleich zu Problemen bei leichten Abweichungen führt, kann mit verschiedenen Maßen eine Ähnlichkeit der Mengen berechnet werden. Eine Möglichkeit ist der Einsatz der Hamming-Distanz, welche auch bei einer geringen Anzahl an falsch klassifizierten Individuen bei an sich gleichen Mengen eine kurze Distanz liefert. Auch das bereits mehrfach erwähnte Jaccard-Maß (Jaccard-Koeffizient) kann eingesetzt werden, wobei die Wahrscheinlichkeit berücksichtigt wird, ob eine zufällige Instanz in der Vereinigung oder der Teilmenge der Sets auftreten kann⁵⁴.
- *Vergleich auf Basis von Instanzidentifizierung:* Für den Fall, dass keine Menge gemeinsamer Instanzen existiert, kann versucht werden Instanzen zu identifizieren, welche miteinander zumindest korrespondieren. Eine einfache Technik für eine solche Identifizierung ist die Nutzung von Schlüsseln, falls diese vorhanden sind. Diese können entweder explizit ausgewiesen sein (zum Beispiel in Datenbanken) oder implizit ausgelesen werden (ISBN, Personalnummern, etc.). Falls solche Schlüssel nicht vorhanden sind, können die bereits vorgestellten Techniken zum Matching von *Strings* und *internen Strukturen* eingesetzt werden, um gleiche oder ähnliche Objekte anhand ihrer Eigenschaften zu identifizieren. Im Bereich der Datenbanktheorie sind solche Techniken auch als Objektidentifizierung ((Lim et al. (1993)) oder Duplikatidentifizierung bekannt (Calado et al. (2010))).
- *Vergleich unterschiedlicher Instanzen:* Es ist nicht immer möglich ein gemeinsames Set von Instanzen der Ontologien abzuleiten. Die Daten der Instanzen können allerdings verwendet werden, um Statistiken der Eigenschaften zu erstellen. Somit können (unter der Annahme einer repräsentativen Stichprobe) Daten über bestimmte Klassen erhoben und verglichen werden. Man kann im Weiteren davon ausgehen, dass diese Statistiken für gleiche Klassen auch gleiche Ausprägungen bilden.
Weitere Methoden der sogenannten *Linkage Aggregation* (dt. Übersetzung unbekannt) aus dem Bereich der Datenanalyse erlauben auch die Ähnlichkeitsberechnung von Instanzsets, welche keine gemeinsamen Individuen besitzen, wobei vor allem die Unähnlichkeit der Sets gemessen wird.
Ein Problem bei dem Einsatz der *Linkage Aggregation* ist, dass immer die Distanz genau eines Paares der Instanzen der Sets berechnet wird. Es wird nicht beachtet, ob das untersuchte Paar das Ähnlichste der Sets ist. Dieses Problem

⁵⁴ $\sigma(A, B) = \frac{P(A \cap B)}{P(A \cup B)}$, wobei $P(X)$ die Wahrscheinlichkeit einer zufälligen Instanz in Set X ist.

behebt Valtchev (1999), wobei zunächst die ähnlichsten Paare der Sets gefunden und basierend darauf deren Distanz bemessen werden.

Bis auf wenige Ausnahmen wurden bis jetzt vornehmlich syntaktische und strukturelle Analysetechniken vorgestellt. Zum Abschluss der Grundtechniken des Matchings soll daher auf die vornehmlich *semantisch-basierten Techniken* eingegangen werden. Diese basieren auf Schlussfolgerungen unter dem Einsatz formaler Modelle (*engl.* model-theoretic semantics). Daher benötigen diese Techniken auch vorangegangene Untersuchungen oder externe Modelle, welche als Basis genutzt werden können, um weitere Schlussfolgerungen zu ziehen. Euzenat und Shvaiko (2007) teilen diese *semantisch-basierten Techniken* daher auch in (1) Techniken basierend auf externen Ontologien und (2) deduktive Techniken ein (Euzenat und Shvaiko (2007), S. 110-115):

- *Einsatz externer Ontologien:* Bei einem Vergleich von zwei Ontologien fehlt oft eine gemeinsame Basis oder ein Kontext, anhand dessen diese analysiert und eventuelle Ähnlichkeiten erkannt werden können. Der Einsatz zusätzlicher Ontologien kann dabei helfen, eine solche Basis zu schaffen. Dabei müssen die zu untersuchenden Ontologien nicht unbedingt eine direkte Bindung zur externen Ontologie haben (wie zum Beispiel die Nutzung der Konzepte einer Upper Ontology wie DOLCE oder SUMO), obwohl dies natürlich hilfreich ist. Es kann stattdessen ein indirektes Matching durchgeführt werden. Dabei werden zunächst die Ursprungsontologien mit der externen Ontologie verglichen. Anschließend werden die gefundenen Ähnlichkeiten genutzt, um Beziehungen zwischen den Ursprungsontologien abzuleiten.
- *Deduktive Techniken:* Diese Techniken basieren stark auf formaler Logik und der Überprüfung von Axiomen und Regeln. Für den Einsatz dieser Techniken werden die Ontologien zusammengeführt (*engl.*: merging). Hier werden bereits erwähnte Matching-Techniken der anderen Bereiche eingesetzt. Nach einer Zusammenführung kann zum Beispiel die Erfüllbarkeit (*engl.* satisfiability, kurz SAT) der Axiome und der (vermeintlichen) Relationen der Klassen ($=, \sqsubseteq, \supseteq, \perp$) überprüft werden. Falls die Ontologien beziehungsweise deren Relationen ($=, \sqsubseteq, \supseteq, \perp$) in Form einer Beschreibungslogik ausgedrückt werden können, ist es möglich, zusätzliche Ähnlichkeiten durch den Einsatz der Subsumption abzuleiten. Hierfür ist aber zunächst ein initiales Alignment der Ontologien notwendig, welches als Basis für weitere Schlussfolgerungen dient.

Die Vielzahl der vorgestellten grundlegenden Techniken könnte leicht um weitere Ansätze ergänzt werden, was jedoch den Rahmen dieser Arbeit übersteigen würde. Es bleibt die Frage, wie diese Techniken sinnvoll kombiniert werden können, um die besten Ergebnisse für verschiedene Kontexte zu erreichen. Mit dieser Frage beschäftigt sich der folgende Abschnitt.

3.3.1.3 Matching Strategien

Wie auch der vorangegangene Abschnitt basieren die folgenden Ausführungen auf der Arbeit von Euzenat und Shvaiko (2007) und orientieren sich dabei an der Struktur von Kapitel 5 der Autoren.

Für die Kombination von Matching-Techniken stehen verschiedene Varianten der Komposition zur Verfügung. Die einfachste Variante einer Komposition ist die sequentielle Anordnung. Dabei wird ein initiales Alignment A (welches auch leer sein kann) durch einen Matching-Prozess (beispielweise eines *string-basierten Matchers*) auf Basis der Ontologien o und o' in ein Zwischenergebnis A' überführt. Basierend auf A' wird ein zweiter Matching-Prozess angestoßen (beispielweise ein *semantischer Matcher*) mit dem Ergebnisalignment A'' . Diese Sequenz kann beliebig oft fortgesetzt werden, was jedoch die Qualität des Ergebnisses nicht unbedingt verbessert. Zudem muss darauf geachtet werden, dass bestimmte Matching-Techniken, wie das *semantische Matching*, auf die Ergebnisse vorangegangener Matcher angewiesen sind und nicht ohne Basisalignment ($A \neq \emptyset$) arbeiten können. Eine solche sequentielle Matching-Strategie kann variiert werden, indem die initialen Ontologien o und o' im zweiten Matching-Prozess durch andere oder umfangreichere Ontologien d und d' ersetzt werden. In einem solchen Szenario dienen o und o' als sogenannte Trainingsontologien bevor ein weiteres Matching auf den „echten“ Ontologien d und d' durchgeführt wird. Dies kann unter Umständen zu einem besseren Ergebnis und zum anderen eine effizientere Ausnutzung der Ressourcen ermöglichen. Sequentielle Matching-Strategien arbeiten zudem oft nicht direkt auf den Alignments sondern, erstellen Ähnlichkeitsmatrizen für die zu vergleichenden Entitätspaare der Ontologien. Dies erlaubt eine schrittweise Verbesserung der Ähnlichkeitsmaße der Matrize. Hierfür wird eine initiale Matrize M durch die Ähnlichkeitsberechnung des ersten Matchers zu M' welche durch einen zweiten Matcher zu M'' verbessert wird. Diese Ähnlichkeitsmatrize kann daraufhin zur Erstellung des benötigten Alignments A' verwendet werden. Eine Darstellung dieses Prozesses findet sich in Abbildung 40. (Euzenat und Shvaiko (2007), S. 117-119)

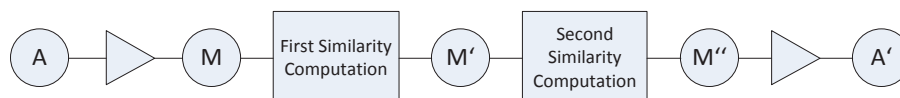


Abbildung 40: Sequentielle Komposition von Matchern auf Basis von Ähnlichkeitsmatrizen, nach Euzenat und Shvaiko (2007), S. 119.

Falls die Ergebnisse der Matching-Prozesse nicht aufeinander aufbauen müssen oder sollen, können diese auch zum Ende der Strategie aggregiert werden. Dies ermöglicht die parallele Ausführung der Matching-Prozesse. Die Aggregation der Ergebnisse kann durch unterschiedliche Methoden geschehen, zum Beispiel durch die Verwendung von Operatoren, die Nutzung von Grenzwerten oder die Auswahl der höchsten Konfidenzwerte. Entsprechende Aggregationsfunktionen sind einfacher auf Ähnlichkeitsmatrizen als auf Alignments definierbar, da entsprechende mathematische Methoden zur Matrizenmanipulation verwendet werden können. Die parallele Komposition

von Matching-Prozessen ist in Abbildung 41 dargestellt. Bei der parallelen Komposition kann unterschieden werden zwischen (Euzenat und Shvaiko (2007), S. 199-120):

- (1) *Heterogener paralleler Komposition*: Bei dieser Form der Komposition wird der Input unterteilt in verschiedene Arten von Daten (Strings, Graphen, etc.), wobei die Aggregation später entweder alle oder nur ausgewählte Ergebnisse nutzen kann.
- (2) *Homogener paralleler Komposition*: Hierbei werden alle Daten von verschiedenen Matchern gemeinsam betrachtet. Die Aggregation verwendet anschließend die besten oder eine Kombination der Ergebnisse.

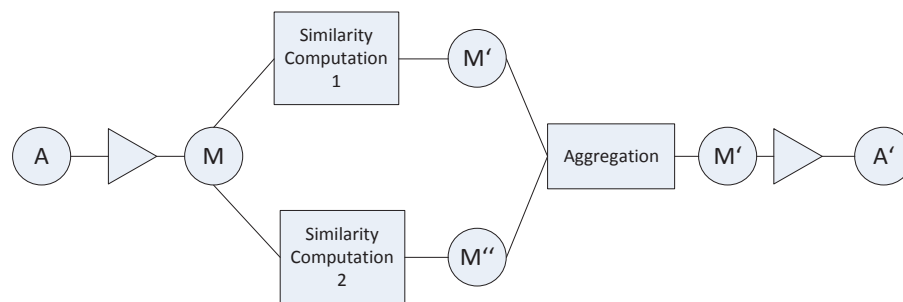


Abbildung 41: Parallele Komposition von Matchern auf Basis von Ähnlichkeitsmatrizen, nach Euzenat und Shvaiko (2007), S. 120.

Bei beiden Formen der Kombination von Matching-Prozessen ergibt sich das Problem, wie die Ähnlichkeitsmaße zu einem Endresultat aggregiert werden sollen. Dieses Problem besteht sowohl auf „lokaler“ als auch auf „globaler“ Ebene. Auf der „lokalen“ Ebene müssen verschiedene Ähnlichkeitsmaße eines untersuchten Objektpaars (zum Beispiel zweier Klassen) miteinander kombiniert werden. So können beispielsweise Ähnlichkeitswerte auf Basis der Namen oder Instanzen der Klasse existieren, welche auf eine sinnvolle Art und Weise kombiniert werden müssen, um einen gemeinsamen Ähnlichkeitswert für die verglichenen Klassen zu liefern. Euzenat und Shvaiko (2007) präsentieren vier mögliche Techniken der Ähnlichkeitsaggregation auf „lokaler“ Ebene (Euzenat und Shvaiko (2007), S. 121-126):

- (1) *Dreiecksnorm (engl. triangular norm, t-norm)*: Dreiecksnormen werden als Operationen in „ungewissen“ Berechnungen (zum Beispiel der Fuzzy-Logic) eingesetzt. Sie können zur Extraktion der größten oder kleinsten Werte der Aggregation verwendet werden. Alternativ kann auch das gewichtete Produkt (engl. weighted product) als Form einer Dreiecksnorm eingesetzt werden.
- (2) *Multidimensionale Distanzen und gewichtete Summen*: Falls die Differenz von Objekten aggregiert werden muss, können multidimensionale Distanzen (zum Beispiel die Minkowski-Distanz oder deren Unterformen der Euklidischen- und Manhattan-Distanz) eingesetzt werden. Eine Normalisierung dieser Distanzen ist möglich, wenn die Resultate durch die längste mögliche Distanz ge-

teilt werden.

Um verschiedenen Ähnlichkeitsmaßen eine unterschiedliche Geltung zu geben, kann die lineare Aggregation der Dimensionen gewichtet werden. Diese gewichteten Summen erlauben zudem die Regulierung der Gewichte zur Generierung besserer Ergebnisse.

- (3) *Fuzzy Aggregation / Fuzzy Weighted Average*: Die Funktionen der sogenannten *fuzzy aggregation* erlauben die Kombination konkurrierender Algorithmen. Ein gutes Beispiel für eine solche Funktion ist der gewichtete Durchschnitt (Fuzzy Weighted Average).
- (4) *Geordneter gewichteter Durchschnitt* (engl. *ordered weighted average, OWA*): Der geordnete gewichtete Durchschnitt verbindet die Gewichtungen gemäß der Werte der unterschiedlichen Dimensionen, die untersucht werden. Damit kann zum Beispiel den höchsten (oder auch den niedrigsten) Werten eine höhere Wichtigkeit zugeteilt werden. So können zum Beispiel immer die höchsten Ähnlichkeitswerte berücksichtigt werden, unabhängig von welchem Matcher diese generiert wurden.

Neben der Ähnlichkeitsberechnung anhand der Entitäten können auf „globaler Ebene“ Ähnlichkeitsbeziehungen zwischen den Entitäten und innerhalb einer ganzen Ontologie berechnet werden. Hierbei ist vor allem der Umgang mit zyklischen Abhängigkeiten wichtig, in welchen Ähnlichkeiten indirekt von sich selbst abhängen. Eine lokale Berechnung solcher Ähnlichkeiten ist nicht möglich. Stattdessen müssen die Ähnlichkeitswerte auf Basis der bereits berechneten Werte iterativ verfeinert werden. Euzenat und Shvaiko (2007) präsentieren in diesem Kontext eine graphbasierte Methode (*Similarity Flooding*) und die Verwendung eines Gleichungssystems (*Similarity Equation Fixed Point*).

Die „Ähnlichkeitsflutung“ (*Similarity Flooding*) untersucht die Ontologien in Form von Graphen und geht davon aus, dass die Ähnlichkeit zweier Knoten von der Ähnlichkeit der angrenzenden Knoten abhängig ist. Der eingesetzte Algorithmus erstellt hierfür einen gerichteten und gewichteten Graph, welcher aus Paaren von Knoten der initialen Graphen der Ontologien besteht. Dieses initiale Matching kann zum Beispiel durch einen einfachen String-Matcher generiert werden. Auf Basis dieser Ähnlichkeiten werden alle Ähnlichkeiten des Graphen neu berechnet. Falls zwei Paare eine bestimmte Ähnlichkeit aufweisen, wird die Ähnlichkeit ihrer benachbarten Knoten erhöht. Dies wird so lange wiederholt, bis keine signifikante Änderung der Ähnlichkeitswerte erzeugt wird. Eine detaillierte Beschreibung dieses Algorithmus und seiner Anwendung im Bereich des Schema-Matchings findet sich in Melnik et al. (2002).

Der Umgang mit Abhängigkeiten zwischen Ähnlichkeitsmaßen ist auch unter Verwendung von Gleichungssystemen möglich. Zunächst werden die Ontologien als Graphen mit verschiedenen Kategorien von Knoten (Klassen, Objekte, Relationen, Eigenschaften, etc.) dargestellt. Die Struktur einer Gleichung basiert dann auf den Ähnlichkeitsfunktionen der jeweiligen Knotenkategorien. Eine Ähnlichkeitsfunktion zweier Klassen besteht beispielsweise aus den Ähnlichkeiten der Label, Instanzen, Super- und Subklassen sowie der Eigenschaften. Durch die Berücksichtigung verschiedener Kategorien und

ihrer Ähnlichkeitsfunktionen entsteht somit ein Gleichungssystem. Der Ausgangspunkt für die Lösung des Gleichungssystems ist die initiale Berechnung der Ähnlichkeiten der zu untersuchenden Paare. Eine detaillierte Beschreibung dieses Algorithmus und dessen Anwendung für OWL-Lite Ontologien findet sich in Euzenat und Valtchev (2004).

Weitere Matching-Strategien beschäftigen sich mit der Verbesserung der Ergebnisse der Ähnlichkeitsberechnung durch Lernmethoden (z.B. neuronale Netze, Bayes'sches Lernen oder Entscheidungsbäume), Wahrscheinlichkeitstheorie (z.B. Bayes'sche Netze) oder die Einbeziehung von Anwendern (z.B. User Feedback). Die einzelne Besprechung dieser Strategien würde allerdings den Umfang dieses Abschnittes überschreiten und es sei daher verwiesen auf Euzenat und Shvaiko (2007) (S. 133-144).

Nach dem Einsatz der Matching-Strategien bleibt zum Abschluss die Extraktion der endgültigen Alignments. Das heißt, dass aus den ermittelten Ähnlichkeitswerten der Entitätspaare Gleichheitsbeziehungen gebildet werden. Zum einen kann dies durch den Einsatz von Grenzwerten erreicht werden. Wird ein bestimmter Grenzwert überschritten, so kann von einer Gleichheitsbeziehung ausgegangen werden. Hierfür kann ein fester Grenzwert (Grenzwert n), ein Delta-Grenzwert (höchster Ähnlichkeitswert abzüglich Delta-Wert d), proportionaler Grenzwert (Prozentwert des höchsten Ähnlichkeitswertes) oder prozentualer Grenzwert ($p\%$ prozentual über den anderen Werten) eingesetzt werden. Zum anderen können die Ähnlichkeitswerte mittels bestimmter Funktionen (zum Beispiel Sigmoidfunktionen⁵⁵) verstärkt oder abgeschwächt werden. Somit entsteht eine Trennung in negative und positive Korrespondenzen. (Euzenat und Shvaiko (2007), S. 144-147)

3.3.2 Evaluierung von Ontology Matching und Alignment

Die Vielzahl an möglichen Techniken und Strategien für das Matching von Ontologien hat zu einer Entwicklung unterschiedlicher Matching- und Alignment-Werkzeuge in den unterschiedlichsten Domänen geführt. Durch unterschiedliche Anforderungen und Einsatzbereiche ist der Vergleich dieser Implementierungen allerdings schwierig. Zur Lösung dieses Problems wurde im Jahre 2005⁵⁶ die Ontology Alignment Evaluation Initiative (OAEI) ins Leben gerufen. Die Vision der OAEI ist die Entwicklung eines Konsens zur Auswertung von Ontologie (und Schema) Matching Algorithmen. Dabei hat die Initiative mehrere Ziele: (1) Bewertung der Stärken und Schwächen von Matching- und Alignment-Systemen, (2) Vergleich der Leistungsfähigkeit der Matching-Techniken, (3) Erhöhung der Kommunikation unter den Entwicklern, (4) Verbesse-

⁵⁵ Beschreibt einen S-förmigen Verlauf.

⁵⁶ Vorläufer der OAEI gab es in Form der Information Interpretation and Integration Conference (I3Con 2004) und dem EON Ontology Alignment Contest 2004, siehe oei.ontologymatching.org.

rung der Techniken zur Evaluierung und (5) die generelle Verbesserung der Arbeiten im Bereich Ontologie-Matching und –Alignment. (Euzenat (2011))

Die OAEI veranstaltet jährlich Workshops zur Auswertung verschiedener Implementierungen im Bereich Ontologie-Matching und –Alignment. Seit 2006 finden die Workshops im Rahmen der International Semantic Web Conference (ISWC) statt. Der Vergleich verschiedener Systeme wird durch den Einsatz standardisierter Tests in unterschiedlichen Themenbereichen (Tracks) möglich. Die Entwickler können entscheiden, welche Tests ihre Systeme durchlaufen und reichen danach ihre Ergebnisse für den Gesamtvergleich ein. In der aktuellen Testsammlung (2010) befinden sich sechs Datensets in vier verschiedenen Tracks (Euzenat et al. (2010), S. 86-87):

- *Benchmark Track*: In diesem Track werden systematische Tests mit dem Ziel durchgeführt, die Bereiche zu identifizieren, in welchen der jeweilige Algorithmus Stärken oder Schwächen aufweist. Verwendet wird eine Ursprungsontologie aus der Domäne „Bibliographie“ und eine Anzahl alternativer Ontologien der gleichen Domäne. Referenzalignments mit den zu erzielenden Korrespondenzen stehen als Vergleich zur Verfügung. Die Tests des Benchmark Tracks decken eine Vielzahl an möglichen Matchingherausforderungen ab und bleiben über die Jahre (soweit wie möglich) stabil, um die Entwicklung der Systeme über einen längeren Zeitraum auswerten zu können. Die Tests beinhalten beispielsweise den Vergleich der Ursprungsontologie mit sich selbst (ID-Test), mit einer irrelevanten Ontologie (Kategorie „einfache Tests“), mit einer Ontologie an Synonymen und einer Ontologie ohne Instanzen (Kategorie „systematische Test“) sowie den Vergleich mit vier realen Ontologien. Insgesamt existieren 73 Testontologien⁵⁷. Zum Vergleich der Systeme werden Precision, Recall und F-Measure gemessen. Das Alignment sieht allerdings nur eine Äquivalenzrelation mit einer Konfidenz von 1 vor.
- *Expressive Ontologies Track*: Die Ontologien dieses Tracks verwenden die Möglichkeiten der Ontologiemodellierung durch OWL (zum Beispiel die *partOf*-Relation) und verschiedene Expressivitäten der Beschreibungslogik. Der Track besteht aus zwei Datensets. Zum einen wird in der Anatomie-Domäne ein Vergleich zwischen der Anatomie einer adulten Maus (Ontologie mit 2744 Klassen) und der menschlichen Anatomie (NCI Thesaurus⁵⁸, Ontologie mit 3304 Klassen) durchgeführt. Zum anderen wird ein Vergleich von Ontologien zur Beschreibung der Organisation von Konferenzen durchgeführt. Dabei wird neben einer automatischen Auswertung anhand eines Referenzalignments auch eine manuelle Überprüfung möglicher „interessanter“ Treffer angeboten.

⁵⁷ Für eine Übersicht und Beschreibung der Tests siehe oei.ontologymatching.org/2010/benchmarks/index.html.

⁵⁸ National Cancer Institute, ncit.nci.nih.gov.

- *Directories and Thesauri Track*: Dieser Track wertet das Matching sehr großer Verzeichnisse von Webseiten aus (Yahoo, Open Directory). Die Untersuchungen des Tracks sind aufgeteilt in 4000 elementare Tests sowie einen Gesamttest, in welchem zwei Ontologien mit 2854 beziehungsweise 6555 Knoten verglichen werden.
- *Instance Matching*: Das Ziel dieses Tracks ist die Evaluierung von Methoden zur Identifizierung von Instanzen beziehungsweise Individuen, welche aus verschiedenen Quellen (Ontologien) stammen, aber dasselbe Objekt in der realen Welt beschreiben.

Zur Unterstützung der Teilnehmer bei der automatischen Auswertung ihrer Systeme kommt seit 2010 die SEALS Plattform⁵⁹ (Semantic Evaluation At Large Scale) zum Einsatz. Die Plattform bietet einen Web Service zur Auswertung eines Matchers an, vorausgesetzt dieser ist auch als Web Service erreichbar und folgt bestimmten Richtlinien der Implementierung (unter anderem die Nutzung des Alignment-Formats der Alignment API, siehe auch Abschnitt 2.3.2). Der SEALS Service liefert eine detaillierte Auswertung für die Ergebnisse des überprüften Tracks inklusive Precision und Recall sowie richtige, falsche und fehlende Korrespondenzen.

Die Auswertung der Gesamtergebnisse wird durch die Mitglieder der OAEI durchgeführt und während des jährlichen Workshops veröffentlicht. Je nach Track werden verschiedene Kriterien zur Auswertung hinzugezogen und entsprechend dargestellt. So ist zum Beispiel im Benchmark-Track der Vergleich der Systeme zur Erreichung eines bestimmten $n\%$ Recalls mit korrespondierender Precision möglich (siehe Abbildung 42 für die Ergebnisse aus 2010).

⁵⁹ www.seals-project.eu

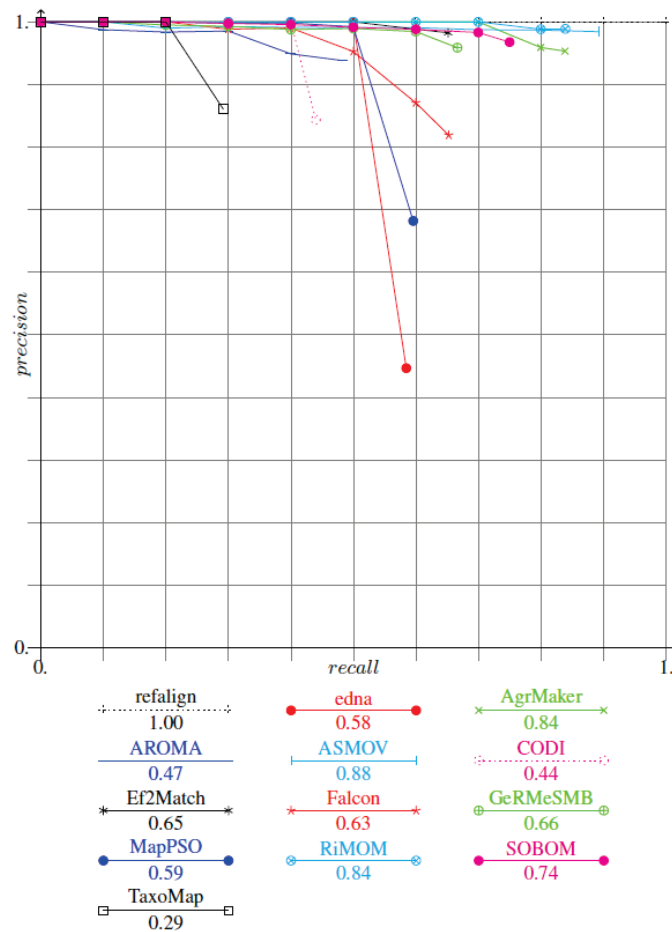


Abbildung 42: Precision/Recall Graph für den Benchmark Track der OAEI 2010, nach Euzenat et al. (2010), S. 96.

Die im nachfolgenden Abschnitt vorgestellten Werkzeuge werden auch im Hinblick auf ihre Ergebnisse in einem oder mehreren OAEI Workshops diskutiert.

3.3.3 Werkzeuge

Im folgenden Abschnitt wird eine Auswahl an Matching- und Alignment-Werkzeugen vorgestellt. Hierbei wird der Focus auf die Implementierungen Falcon-AO (Hu und Qu (2008)) und Lily (Wang und Xu (2009)) gelegt, welche in der prototypischen Implementierung verwendet werden. Lily und Falcon-AO lieferten in den Jahren 2007 und 2008 sehr gute Ergebnisse in der Ontology Alignment Evaluation Initiative (OAEI) 2010 (siehe auch Abschnitt 2.3.2) und wurden daher als Grundlage der Implementie-

rung im Jahre 2009 ausgewählt. Weiterhin soll ASMOV (Jean-Mary et al. (2009)) vorgestellt werden, eine aktuelle Implementierung mit sehr guten Ergebnissen in den Untersuchungen der OAEI 2010. Eine umfangreiche Beschreibung und Klassifizierung von weiteren Werkzeugen findet sich zudem in Euzenat und Shvaiko (2007) (S. 153ff) sowie in Ehrig (2007) (S. 47ff)

3.3.3.1 Falcon-AO

Falcon-AO ist ein Ontology-Alignment Werkzeug und eine Komponente der semantischen Web Infrastruktur Falcon. Das Werkzeug ist Java-basiert und bietet die Möglichkeit RDF(S) und OWL Ontologien miteinander zu vergleichen. (Hu und Qu (2008), S. 237)

Dabei setzt der zugrundeliegende Algorithmus auf *linguistisches* und *strukturelles Matching*. Das linguistische Matching basiert auf der Konstruktion von virtuellen Dokumenten, sogenannten V-Docs (Yuzhong et al. (2006)), welche miteinander verglichen werden können. Die V-Docs werden sowohl aus den Namen und Beschreibungen der zu vergleichenden Entitäten als auch aus den benachbarten Informationen der Entitäten konstruiert. Die Begriffe der V-Docs können nach bestimmten Kriterien gewichtet werden (zum Beispiel die Entfernung zur Ursprungsentität). Die Struktur des RDF Graphen kann genutzt werden, um die benachbarten Informationen zu extrahieren und in die V-Doc mit einzubeziehen. Die virtuellen Dokumente können anschließend mittels traditionellen Vektorraumberechnungen verglichen werden. Als weiterer linguistischer Matcher wird I-Sub (Stoilos et al. (2005)) eingesetzt. Dieser nutzt eine Metrik des stringbasierten Matchings basierend auf der Arbeit von Winkler (1999) und zugeschnitten auf die Domäne des Ontologievergleichs. Das strukturelle Matching verwendet GMO (Graph Matching for Ontologies, Hu et al. (2005)), welches die Entitäten und deren Beziehungen (in Form von Tripeln) auf Basis bipartiter Graphen vergleicht. Auf Basis eines bestehenden Alignments werden die Ähnlichkeiten innerhalb des Graphen rekursiv propagiert. Als initiales Alignment können zum Beispiel die Ergebnisse des linguistischen Matchings verwendet werden. (Hu und Qu (2008), S. 237-238)

Falcon bietet auch die Möglichkeit zum Umgang mit sehr großen Ontologien. Zum Einsatz kommt in diesem Zusammenhang das sogenannte partitionsbasierte Block-Matching (*engl.*: partition-based block matching, PBM). Das PBM teilt die Ontologien in unterschiedliche Blöcke auf und berechnet die Ähnlichkeiten zwischen den Blöcken (sogenannte *Divide-and-Conquer* Methode). Die Entitäten der Blöcke mit dem höchsten Ähnlichkeitsmaß können anschließend durch die linguistischen und strukturellen Matchings verglichen werden. Die Ergebnisse der linguistischen (V-Doc und I-Sub) und strukturellen (GMO) Matchings werden durch die Verwendung von Grenzwerten kombiniert, welche kontinuierlich erhöht werden. Falls die Anzahl an Übereinstimmungen sich an die Anzahl der Entitäten der kleinsten Ontologie annähert, wird von einer hohen strukturellen Treffsicherheit ausgegangen und das GMO muss nicht mehr eingesetzt werden. Somit kann zum Beispiel dynamisch der Grenzwert für die linguistischen Matcher verringert werden, damit diese einen höheren Einfluss auf das finale Alignment haben. (Hu und Qu (2008), S. 238-239)

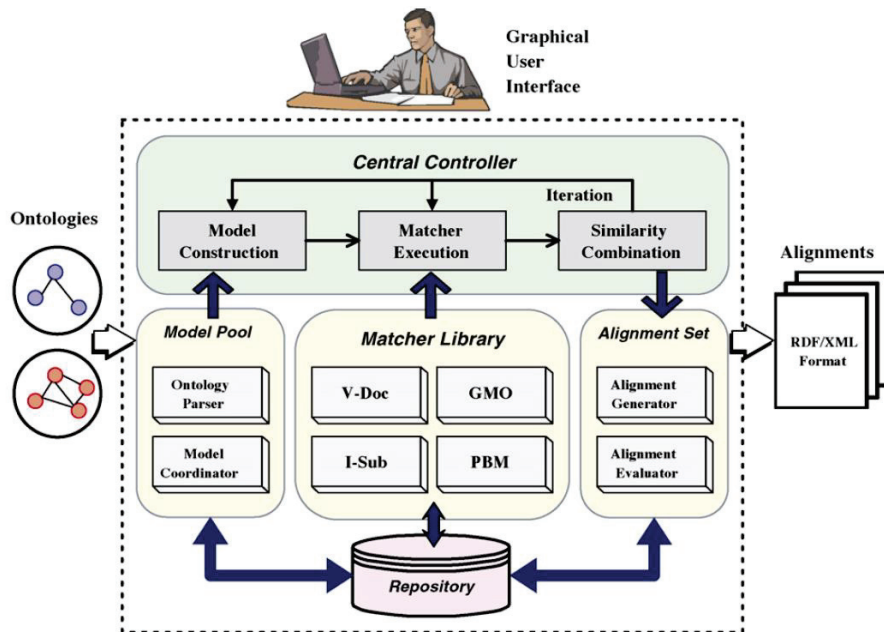


Abbildung 43: Systemarchitektur von Falcon-AO, nach Hu und Qu (2008), S. 238.

Die Architektur von Falcon-AO ist in Abbildung 43 dargestellt. Das Werkzeug besteht aus fünf Hauptkomponenten: (1) Model Pool, (2) Matcher Library, (3) Alignment Set, (4) Central Controller und (5) Repository. Der Model Pool verwendet das Jena Framework⁶⁰, um die Eingangsontologien zu parsen, und eliminiert eventuelle Heterogenität der Ontologien (zum Beispiel überflüssige Axiome) so weit wie möglich. Die Matcher Library stellt die linguistischen Matcher V-Doc und I-Sub, den strukturellen Matcher GMO und die Partitionierungsmethode PBM bereit. Das Alignment-Set ist für die Erstellung und Serialisierung der Ontology Alignments in einem RDF/XML Format (Euzenat (2004), siehe auch Abschnitt 2.3.2) zuständig. Weiterhin können die erstellten Alignments mit Referenz-Alignments verglichen und somit auch die Qualität der gewählten Matching-Strategie evaluiert werden (Überprüfung von *Precision* und *Recall*). Der Central Controller erlaubt die manuelle Konfiguration von Matching-Strategien. Er ist weiterhin für die Ausführung der Matcher und die Kombination ihrer Ergebnisse verantwortlich. Das Repository speichert die benötigten Daten für den Matching-Prozess. Zusätzlich existiert eine grafische Schnittstelle (GUI) zur Konfiguration der Komponenten und Darstellung der Ergebnisse. (Hu und Qu (2008), S. 237)

⁶⁰ jena.sourceforge.net

	Benchmark	Anatomy	Directory	Food	Environment	Library	Conference
Precision	0.92	0.96	0.55	0.84	0.87	0.97	0.73
Recall	0.86	0.59	0.61	0.45	0.35	0.87	0.57
Total time	300 s	12 min	110 s	5.75 h	1.75 h	40 min	160 s

Tabelle 3: Ergebnisse der OAEI 2007 für Falcon-AO, nach Hu und Qu (2008), S. 239.

Wie bereits erwähnt, erzielte Falcon-AO sehr gute Ergebnisse in den Vergleichen der OAEI 2007 (mit einer Teilnahme in den Jahren 2005, 2006, 2007 und 2010). Dabei erzielte Falcon-AO gute bis sehr gute Werte in nahezu allen Testkategorien (mit Ausnahme von *Directory*) und konnte alle Ontologien in einer zumutbaren Zeit vergleichen. Ein Überblick über die Ergebnisse der OAEI 2007 findet sich in Tabelle 3. (Hu et al. (2007); Hu et al. (2010))

3.3.3.2 Lily

Lily ist nach eigenen Angaben ein Ontology-Mapping System und besteht aus vier grundlegenden Funktionen: (1) *Generic Ontology Matching* (GOM), (2) *Large Scale Ontology Matching* (LOM), (3) *Semantic Ontology Matching* (SOM) und (4) *Ontology Mapping Debugging*. (Wang und Xu (2009), S. 186)

Die Grundlage für die Matching-Algorithmen von Lily bildet die Verwendung sogenannter semantischer Subgraphen (*engl.*: semantic subgraphs). Damit wird versucht für jede Entität einen Subgraph zu extrahieren, welcher die Verbindung zu einer anderen Entität am besten beschreibt. Diese Idee basiert auf der Arbeit von Christos et al. (2004) und wurde um die Beschreibung der Bedeutung der Subgraphen erweitert und daher semantischer Subgraph getauft (Wang et al. (2010)). Die Extraktion der Subgraphen geschieht vor der eigentlichen Ähnlichkeitsberechnung zwischen den Entitäten. Im Rahmen des GOM werden die strukturellen und textuellen Eigenschaften der semantischen Subgraphen für die Ähnlichkeitsberechnung ausgewertet. Zum Einsatz kommt in diesem Prozess ein Semantic Description Document (SDD) Matcher, welcher die textuelle Ähnlichkeit (*engl.*: literal similarity) für Konzepte und Eigenschaften der Ontologien berechnen kann. Ein SDD eines Konzeptes beispielsweise besteht aus den Informationen über Klassenhierarchien, verbundene Eigenschaften und Instanzen. Ein SDD einer Eigenschaft hingegen verwendet zusätzlich noch Domänen (domain) und Reichweite (range). Die berechneten Ähnlichkeiten der Entitäten werden anschließend unter Verwendung empirisch ermittelter Gewichte kombiniert. Der SDD Matcher benötigt allerdings eine reichhaltige textuelle Beschreibung der Entitäten. Ist eine solche Beschreibung nicht vorhanden, kann ein sogenannter Similarity Propagation Matcher (SSP) eingesetzt werden. Für die Verbreitung der Ähnlichkeiten (*engl.*: similarity propagation) werden im Vergleich zu anderen Algorithmen die bereits erwähnten semantischen Subgraphen eingesetzt. Somit sollen Ähnlichkeiten ermittelt werden, welche in der reinen textuellen Ähnlichkeitsberechnung nicht auffindbar sind. Durch die Verbreitung von Ähnlichkeiten können allerdings auch zusätzliche falsche Alignments (falsch positive Treffer) entstehen. Daher muss überprüft werden, wann genau die Ähnlichkeitsverbreitung eingesetzt werden soll. (Wang und Xu (2009), S. 188)

Im Gegensatz zu anderen Verfahren verwendet Lily keine Einteilung in Blöcke, um sehr große Ontologien zu vergleichen. Stattdessen werden im Rahmen des LOM positive und negative Anker genutzt, um zu ermitteln welche Paare im weiteren Matchingprozess untersucht werden können. Die Verwendung von Ankerentitäten basiert auf der Annahme, dass bestimmte Sub- oder Superklassen und Nachbarn anhand der Ähnlichkeit der Ankerklassen unter bestimmten Umständen nicht im weiteren Matching betrachtet werden müssen. Somit kann der Aufwand der Ähnlichkeitsberechnung stark reduziert werden. (Wang et al. (2011))

Das semantische Ontology Matching (SOM) basiert auf der Idee, das gesamte Web als Wissensbasis zu verwenden und semantische Zusammenhänge mit Hilfe einer Suchmaschine identifizieren und extrahieren zu können. Anhand lexikalisch-syntaktischer Muster werden zunächst Kandidaten für das Mapping der Entitäten identifiziert. Anhand bestimmter Regeln wird die Kandidatenliste verfeinert und letztlich ein Mapping automatisch ausgewählt. Eine detaillierte Beschreibung des Prozesses findet sich in Li et al. (2007).

Eine Besonderheit von Lily ist die Möglichkeit zur Identifizierung von Fehlern in Ontologie-Mappings (Ontology Debugging). Hierbei wird die in Wang und Xu (2008a) vorgestellte Methode eingesetzt, um Fehler zu erkennen und automatisch oder mit Hilfe des Nutzers zu korrigieren. Mögliche Fehler sind redundante oder inkonsistente Mappings, ungenaue und anormale Beziehungen können als Warnungen gemeldet werden. (Wang und Xu (2009), S. 189)

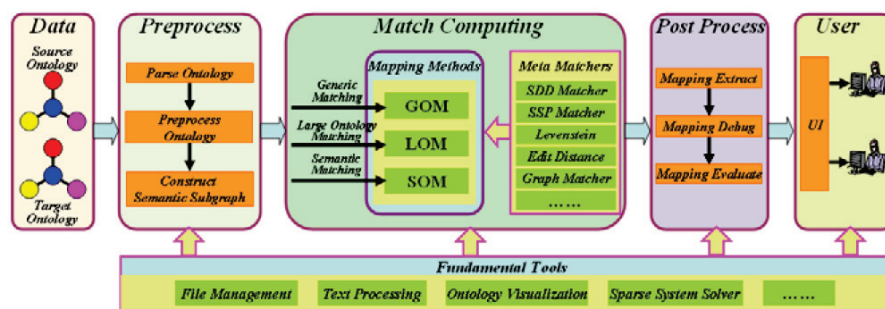


Abbildung 44: Architektur von Lily, nach Wang und Xu (2009), S. 187.

Der generelle Matching-Prozess von Lily besteht aus drei Phasen: (1) Preprocess, (2) Match Computing und (3) Postprocess. In Abbildung 44 ist der Prozess und die Systemarchitektur von Lily veranschaulicht. In der ersten Phase werden die zu vergleichenden Ontologien geparkt und die benötigten semantischen Subgraphen konstruiert. Die zweite Phase bildet den Kern des Systems. Hier werden passende Methoden zur Ähnlichkeitsanalyse ausgewählt und ausgeführt. Die jeweiligen Mapping-Methoden (GOM, LOM und SOM) haben hierbei Zugriff auf eine Vielzahl sogenannter Meta-Matcher. Neben den vorgestellten SDD und SSP Matchingmechanismen stehen auch „klassische“ Techniken der Ähnlichkeitsberechnung wie Levenshtein- und Edit-Distanz

sowie verschiedene Graphmatcher zur Verfügung (siehe Abschnitt 3.3.1.2). In der dritten und letzten Phase werden die Alignments extrahiert und durch das Ontology Debugging bereinigt. Die Architektur stellt noch für den Gesamtprozess nötige Werkzeuge wie ein File Management und eine Ontologievisualisierungskomponente zur Verfügung. (Wang und Xu (2009), S. 186-187)

Year	2004				2005	2006	2007				2008			
System	Fujitsu		PromptDiff		Falcon	RiMOM	ASMOV		Lily		ASMOV		Lily	
test	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.
1xx	0.99	1.00	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
2xx	0.93	0.84	0.98	0.72	0.98	0.97	1.00	0.98	0.99	0.99	1.00	0.98	0.99	0.98
3xx	0.60	0.72	0.93	0.74	0.93	0.83	0.83	0.82	0.85	0.82	0.81	0.80	0.81	0.77
H-means	0.88	0.85	0.98	0.77	0.97	0.96	0.97	0.96	0.97	0.97	0.97	0.96	0.98	0.96

Tabelle 4: Entwicklung der Ansätze mit den besten Ergebnissen im Benchmark-Test der OAEI, nach Caracciolo et al. (2008), S. 82.

Lily konnte im Rahmen der OAEI 2007 und 2008 vergleichsweise sehr gute Ergebnisse produzieren (siehe Tabelle 4, bezogen auf den „Benchmark“-Test der OAEI), hat diese Erfolge in den darauffolgenden Jahren jedoch nicht wiederholen können. (Wang und Xu (2008b))

3.3.3.3 Automated Semantic Matching of Ontologies with Verification (ASMOV)

Im Vergleich zu den zwei vorherigen Ansätzen des Ontology Matchings setzt ASMOV neben einer lexikalischen und strukturellen Analyse auch ein extensionelles Matching (siehe Abschnitt 3.3.1.2) ein. Weiterhin wird eine semantische Verifikation zur Überprüfung des Alignments durchgeführt, in welcher die Axiome der Ontologien unter Berücksichtigung der ermittelten Treffer nachgeprüft werden. (Jean-Mary et al. (2009), S. 235)

Der Matching-Prozess von ASMOV ist in Abbildung 45 dargestellt und besteht aus vier Phasen: (1) Pre-Processing, (2) Ähnlichkeitsberechnung, (3) Alignment Extraktion und (4) semantische Verifikation. In der ersten Phase verarbeitet ein lexikalischer Matcher zunächst die Eingangsontologien unter Berücksichtigung (und soweit vorhanden) eines bestehenden partiellen Alignments. Die zweite Phase berechnet die Ähnlichkeiten aller möglichen Entitätspaare unter Einbeziehung externer, interner und individueller Matcher. Das Ergebnis der Ähnlichkeitsberechnung sind Ähnlichkeitsmatrizen, aus denen ein vorläufiges Alignment extrahiert werden kann (Phase 3). Der letzte Schritt besteht aus einer semantischen Verifikation des vorläufigen Alignments. Dabei können nicht verifizierte Übereinstimmungen entstehen, wodurch eine Anpassung des Alignments nötig wird (Rekursion). Der Prozess ist iterativ angelegt und ermöglicht somit mehrere Wiederholungen der Ähnlichkeitsberechnung, um ein finales Alignment zu generieren. (Jean-Mary et al. (2009), S. 237-238)

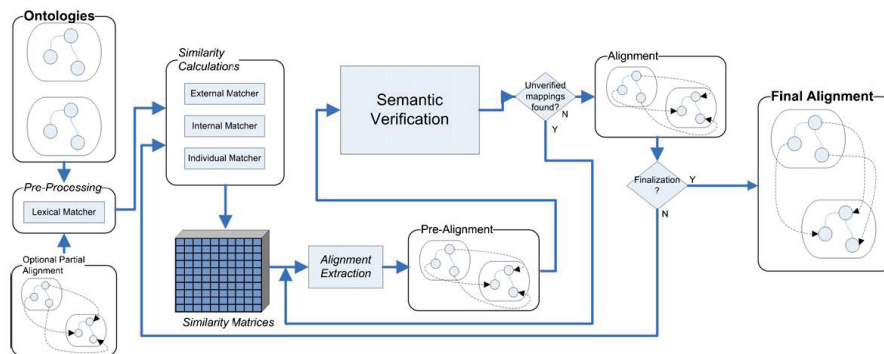


Abbildung 45: Block-Diagramm des ASMOV-Algorithmus, nach Jean-Mary et al. (2009), S. 238.

Für die Ähnlichkeitsberechnung in der ersten und zweiten Phase kommen ein *lexikalisches* (beziehungsweise terminologisches) Maß, zwei *strukturelle* Maße (relational und intern) sowie ein *extensionelles* Maß zum Einsatz. Die *lexikalische* Ähnlichkeitsberechnung verwendet alle menschlich-lesbaren Informationen der Ontologien. Dies beinhaltet die Labels, IDs und Kommentare der Entitäten. Für die Berechnung der Ähnlichkeit von Labels und IDs können auch externe Thesauri eingesetzt werden, welche aber nicht zwingend notwendig sind. Falls externe Ressourcen vorhanden sind, können Synonyme und Antonyme sowie die Bedeutungsnahe der Konzepte nach Lin (1998) überprüft werden. Wenn keine externen Ressourcen zur Verfügung stehen, wird nur die Gleichheit der Strings überprüft. Die Berechnung der lexikalischen Ähnlichkeit der Kommentare hingegen basiert auf einer Levenshtein-Distanz der Token der Kommentare. Die Ähnlichkeiten der Labels, IDs und Kommentare werden anschließend gewichtet und bilden somit das *lexikalische* Ähnlichkeitsmaß. Dieses verändert sich nicht bei mehreren Iterationen des Prozesses und wird daher einmalig in der Pre-Processing Phase berechnet. (Jean-Mary et al. (2009), S. 239-241)

Die *strukturelle* Ähnlichkeit wird anhand der *relationalen* und der *internen* Ähnlichkeit ermittelt. Die *relationale* Ähnlichkeit basiert auf einer Kombination der Ähnlichkeiten von Eltern- und Kindknoten durch eine gewichtete Summe. Diese wird für Klassen, Instanzen und Eigenschaften der Ontologien berechnet. Für die Berechnung der *internen* Ähnlichkeit muss allerdings zwischen *Klassen*, *Instanzen* und *Eigenschaften* unterschieden werden. *Eigenschaften* können verglichen werden, indem der gewichtete Durchschnitt der Ähnlichkeiten von Reichweite (*engl.*: range) und Domäne (*engl.*: domain) gebildet wird. Für die *Klassen* wird die Ähnlichkeit der Eigenschaftsrestriktionen (*engl.*: property restrictions, siehe auch Abschnitt 3.1.2 in Bechhofer et al. (2004)) untersucht, das heißt sowohl die Einschränkungen der Werte als auch der Kardinalität. Zusätzlich wird auch die Ähnlichkeit der Eigenschaften selbst in die Berechnung

miteinbezogen⁶¹. Die *interne* Ähnlichkeit der *Instanzen* basiert auf dem Vergleich der Werte ihrer Eigenschaften (Objekt- und Datenwerte). Die entstehende Matrize wird anschließend anhand eines Algorithmus für die Berechnung der Ähnlichkeit von Entitäts-Sets ausgewertet. (Jean-Mary et al. (2009), S. 242-243)

Die *extensionelle* Analyse nutzt die Instanzen (soweit vorhanden) der Ontologien, um eine Ähnlichkeit der *Klassen* und *Eigenschaften* zu ermitteln. Die Ähnlichkeit der *Klassen* basiert auf den bereits ermittelten Ähnlichkeitsmatrizen der Klasseninstanzen, geordnet nach ihren übergeordneten Klassen. Für die Berechnung der Ähnlichkeiten von *Eigenschaften* werden zunächst alle Instanzen der ersten Ontologie O identifiziert, welche für eine bestimmte Eigenschaft einen Wert (entweder Objekt- oder Datenwert) aufweisen. Daraufhin werden diejenigen Instanzen der zweiten Ontologie O' gesucht, welche aus dem vorangegangenen Instanzenvergleich eine Ähnlichkeit ungleich 0 mit den ermittelten Instanzen von O haben. Die Eigenschaften der auf diese Weise ermittelten Instanzen können abschließend anhand ihrer Werte verglichen werden (beispielsweise anhand ihrer lexikalischen Ähnlichkeit). (Jean-Mary et al. (2009), S. 243-244)

Das Resultat der zweiten Phase ist eine Ähnlichkeitsmatrize der einzelnen Entitäten der Ontologien, wobei die Ähnlichkeit eines Entitätspaares durch den gewichteten Durchschnitt seiner Ähnlichkeitsmaße gebildet wird. Möglicherweise fehlende Entitäten werden mit 0 gewichtet. Optional kann ein Referenzalignment als Input verwendet werden, welches bezüglich fehlender Ähnlichkeiten aktualisiert wird. (Jean-Mary et al. (2009), S. 239)

Zur Überprüfung der semantischen Korrektheit (*semantic verification*) wird anschließend ein vorläufiges Alignment extrahiert. In dieser Phase wird überprüft, inwieweit bestimmte Axiome, welche durch das Alignment abgeleitet wurden, innerhalb der Ontologien verifiziert werden können (Assertion des Axioms). Fünf unterschiedliche Rückschlüsse (*engl.*: inferences) werden überprüft. Beispielsweise können mehrere Entitäten der Ontologie O mit derselben Entität der Ontologie O' korrespondieren. Das entstehende Alignment $\langle e_1, e'_1 \rangle$ und $\langle e_2, e'_1 \rangle$ kann durch die Behauptung $e_1 = e_2$ in der Ontologie O überprüft werden. Der Algorithmus entfernt die nicht verifizierbaren Korrespondenzen und berechnet gegebenenfalls das vorläufige Alignment neu. Dieses wird solange verifiziert bis keine Entfernungen mehr nötig sind. Abschließend kann das vorläufige Alignment als Input für eine erneute Ähnlichkeitsberechnung in Phase zwei dienen oder finalisiert werden. (Jean-Mary et al. (2009), S. 244-245)

Das ASMOV System nimmt seit einigen Jahren regelmäßig mit guten bis sehr guten Ergebnissen an der OAEI teil und konnte 2010 die besten Ergebnisse im Benchmark Test der Evaluierung erzielen. Eine Übersicht über die Ergebnisse der OAEI 2009 und 2010 für das ASMOV System findet sich in Tabelle 5.

⁶¹ Ein anschauliches Beispiel dieser komplexen Berechnung findet sich in Tabelle 5 von Jean-Mary et al. (2009).

Level	ASMOV 2010			ASMOV 2009		
	Precision	Recall	F1	Precision	Recall	F1
0	1.00	1.00	1.00	1.00	1.00	1.00
1	0.99	1.00	0.99	1.00	1.00	1.00
2	1.00	0.99	0.99	1.00	0.99	0.99
3	0.99	0.98	0.98	0.99	0.98	0.98
4	1.00	0.98	0.99	0.99	0.98	0.98
5	0.99	0.94	0.96	0.97	0.93	0.95
6	0.98	0.90	0.94	0.95	0.89	0.92
7	0.98	0.87	0.92	0.93	0.83	0.88
8	0.98	0.77	0.86	0.90	0.71	0.79
9	0.97	0.64	0.77	0.83	0.48	0.61
10	0.90	0.29	0.44	0.40	0.04	0.07
3xx	0.88	0.84	0.86	0.81	0.82	0.81
All	0.98	0.89	0.93	0.95	0.87	0.91

Tabelle 5: Resultate des Benchmark Tests der OAEI für ASMVO 2010 und 2009, nach Jean-Mary et al. (2010), S. 129.

3.4 Heterogene Service Discovery

Die überwiegende Anzahl der Ansätze zur Suche nach Services (siehe Abschnitt 3.1) geht davon aus, dass entweder nur eine Ontologie zur Beschreibung der Services verwendet wird oder dass, falls mehrere Ontologien zum Einsatz kommen, diese durch Mediatoren und vorhandene Alignments miteinander verbunden sind. Die Zielstellung der vorliegenden Arbeit ist allerdings eine Unterstützung der Suche nach Services von unterschiedlichen Anbietern und somit (meistens) heterogenen Beschreibungen (Verwendung von mindestens zwei Ontologien ohne vorherige Definition von Mediatoren oder Alignments). Der aktuelle Abschnitt soll sich mit dem Stand der Forschung von Ansätzen beschäftigen, welche ein solches Ziel der Suche nach heterogen beschriebenen Services unterstützen. In diesem Teilbereich existiert nach Wissen des Autors nur eine geringe Anzahl an Lösungsansätzen, welche sich zum Teil auch mit den bereits vorgestellten „homogenen“ Service Discovery Ansätzen überschneiden beziehungsweise diese ergänzen. Eine Übersicht über bestehende Ansätze zur heterogenen Service Discovery wurde in Rake et al. (2009) vorgestellt. Diese soll im Folgenden aufgegriffen, vertiefend vorgestellt und um aktuelle Entwicklungen erweitert werden. Hierbei werden sowohl von bestehenden Ontologiesprachen und Service Discovery Ansätzen abhängige als auch unabhängige Ansätze besprochen.

3.4.1 Semantische Service Discovery in einer Multi-Ontologie-Umgebung

Usanavasin et al. (2005) beschreiben das Problem der semantischen Suche nach Services anhand eines Vergleichs der In- und Outputs, welche mittels unterschiedlicher Ontologien beschrieben sind. Die Autoren argumentieren dabei, dass die Annahme bestehender Ansätze der Nutzung einer einzelnen Ontologie nicht pragmatisch ist. Zur Lösung

wird ein Ansatz zur Berechnung der semantischen Ähnlichkeit der In- und Outputs von Web Services vorgestellt. Obwohl die Autoren bestehende Ansätze für das Matching und Alignment von Ontologien diskutieren (unter anderem Noy und Musen (2001) sowie Ehrig und Sure (2004)), entscheiden sie sich gegen die Wiederverwendung und für die Neuentwicklung eines eigenen Algorithmus. Hierbei wird argumentiert, dass die bestehenden Ansätze das Ziel der Ähnlichkeitsberechnung während der Service Discovery Phase nicht ausreichend unterstützen. (Usanavasin et al. (2005), S. 59-60)

Der Ansatz von Usanavasin et al. (2005) basiert auf der Ähnlichkeitsberechnung der In- und Outputs der Services anhand folgender Informationen der Eigenschaften der beschreibenden Klassen: (1) URIs, (2) Namen oder Label, (3) Synonymsets, (4) Range-Klassen, (5) Domain-Klassen, (6) Ober- und (7) Unterklassen. Der Vergleich der URIs wird unter der Annahme durchgeführt, dass Konzepte und Eigenschaften mit gleicher URI identisch sind. Ist dies nicht der Fall, so werden weitere Informationen zur Ähnlichkeitsberechnung hinzugezogen. Die Ähnlichkeit der Namen basiert auf einem String-Matching anhand der Levenshtein-Distanz. Zusätzlich werden die Namen verwendet, um eine Ähnlichkeitsberechnung mittels WordNet durchzuführen. Hierbei werden die SynSets der Eigenschaften und deren eventuelle Überschneidungen überprüft. Range-Klassen werden zur Überprüfung der Ähnlichkeit der Datentypen hinzugezogen. Diese basiert auf der Typenhierarchie von XML und beurteilt die Kompatibilität der Datentypen anhand der Möglichkeiten zur Umwandlung in andere Typen. Datentypen, die nur teilweise oder nicht in den jeweils anderen Typ umgewandelt werden können, werden mit einer geringeren beziehungsweise keiner Übereinstimmung klassifiziert. Als weiteres Maß wird die Ähnlichkeit der Domänen-Klassen untersucht. Diese setzt sich aus der Ähnlichkeit der Namen der Domänenklassen (String- und Synonymähnlichkeit, siehe oben) und der Ähnlichkeit der Eigenschaften der Domänenklassen zusammen. Die Ähnlichkeit der Eigenschaften basiert auf der set-theoretischen Berechnung nach Tversky (1977) (*feature similarity*), wobei Übereinstimmungen von Eigenschaften mittels eines minimalen Grenzwertes k anhand ihrer syntaktischen Ähnlichkeit bestimmt werden (das heißt, überschreitet die Ähnlichkeit zweier Eigenschaften den Grenzwert k wird dies als Übereinstimmung gewertet). Letztlich werden die Ober- und Unterklassen der Eigenschaften auf Ähnlichkeiten überprüft. Die Ähnlichkeiten der Ober- und Unterklassen werden, wie bereits die Domänenklassen, anhand der Ähnlichkeit der Namen und ihrer Merkmale (*features*) berechnet. Die Gesamtähnlichkeit der Eigenschaften basiert abschließend auf einer Gewichtung der Ähnlichkeit der Namen, Synonyme, Typen, Domänen und Nachbarn (Ober- und Unterklassen). (Usanavasin et al. (2005), S. 62-64)

Es ist anzumerken, dass Usanavasin et al. (2005) nur die Eigenschaften einer Klasse zur Beschreibung von In- und Outputs nutzen. Eine komplexere Beschreibung der Services unter Nutzung einer Mehrzahl an Konzepten sowie dem Einsatz von Vor- und Nachbedingungen wird nicht diskutiert. Der Ansatz ähnelt in seiner Beschreibung und im vorgestellten Vergleich mit Ehrig und Sure (2004) mehr einer generischen Möglichkeit zum Vergleich von Ontologieklassen unabhängig des Kontextes der semantischen Service Discovery.

3.4.2 Nutzung von Konzept-Matching zum effizienten Umgang mit heterogenen Ontologien

Kim und Lee (2010) motivieren die Problematik heterogener Ontologien und somit auch Servicebeschreibungen mit der These, dass Services auf Basis von bestehenden Ontologien annotiert werden, welche von den Nutzern durch die Suche im Internet gefunden werden. Dieser Umstand würde zu der Notwendigkeit führen, nicht nur mit zwei oder drei, sondern mit einer ungeahnten Vielzahl an heterogenen Ontologien zur Servicebeschreibung umgehen zu müssen. Allerdings argumentieren die Autoren auch, dass durch die Nutzung und eventuelle Erweiterung bestehender Ontologien ein stabiles Fundament an Kernontologien entstehen kann. Somit würden lose-gekoppelte globale Ontologien entstehen, welche eine Vielzahl (teilweise) verbundener und überlappenden Ontologien beinhalten. Dies führt wiederum zu der Beobachtung, dass bestehende Ansätze der semantischen Service Discovery mit Mappings zwischen einer großen Anzahl von Ontologien nicht umgehen können. Unter der Prämisse, dass Nutzer tendenziell jede vorhandene Ontologie für eine Servicebeschreibung nutzen können, wäre außerdem das Vorhalten aller dieser Ontologien für ein Reasoning weder praktikabel noch skalierbar. Diese Überlegungen führen die Autoren zu der Anforderung eines effizienten Konzept-Matchings zwischen vielen unterschiedlichen Ontologien. (Kim und Lee (2010), S. 1536)

Das von den Autoren verwendete Konzept-Match untersucht die Attribute der Servicebeschreibung (in diesem Fall Kategorie, Typ, Input und Output) und misst das Maß der Übereinstimmung mit der Anfrage in Form der Treffer *exact*, *plugin*, *subsume*, *intersection* und *disjoint*. Somit kann ein Treffer für einen Service eventuell durch eine Mehrzahl an Treffern für das Konzept-Match entstehen, sollten mehrere Attribute zur Servicebeschreibung verwendet werden. (Kim und Lee (2010), S. 1537)

Der Service Discovery Prozess von Kim und Lee (2010) basiert vornehmlich auf einem Mechanismus zum Laden und Entladen von Ontologien, um eine Vielzahl verschiedener Ontologien auf Konzept-Matches für eine Serviceanfrage untersuchen zu können. Die Autoren zeigen zunächst, dass sowohl die Ladezeit als auch das Reasoning auf der Ontologie proportional zu der Größe der Ontologie steigt. Zu Beginn des Discovery Prozesses wird überprüft, ob die zu vergleichenden Konzepte aus derselben Ontologie stammen. Ist dies nicht der Fall wird überprüft, ob ein Alignment für die verwendeten Ontologien existiert. Hierzu wird auf ein entsprechendes Alignment-Repository zurückgegriffen (ein solches Repository kann von Nutzern verwendet werden, um bestehende Mappings zwischen Ontologien zu speichern und Anderen zur Verfügung zu stellen). Nachdem das Alignment in ein entsprechendes Format gerendert wurde (zum Beispiel OWL), wird es zusammen mit den Ontologien in den Reasoner geladen. Nach den Autoren liegt der Vorteil darin, dass nur die Ontologien (und eventuell ein Alignment) in den Reasoner geladen werden, welche für das aktuelle Konzept-Match (Vergleich von Konzepten C_A und C_B) benötigt werden. Dies steht im Vergleich zu anderen Verfahren, welche gleichzeitig die verwendeten Ontologien aller verfügbaren Servicebeschreibungen in den Reasoner laden und dann ein entsprechendes Konzept- bezie-

hungsweise Service-Match berechnen. Es wird argumentiert, dass aktuelle Reasoner-Implementierungen nicht mit einer so hohen Anzahl an Ontologien (und Alignments) umgehen können. Stattdessen werden nicht benötigte Ontologien wieder entladen, was zwar ein schnelleres Reasoning, aber auch gleichzeitig eine höhere Antwortzeit auf die Anfragen generiert. Um die Ladezeit zu reduzieren, schlagen Kim und Lee (2010) ein Caching eines Subsets der verwendeten Ontologien vor. Zur Bestimmung der zu cachenden Ontologien werden sowohl Aktualität (*least recently used*) sowie Frequenz (*least frequently used*) berücksichtigt. (Kim und Lee (2010), S. 1538-1539)

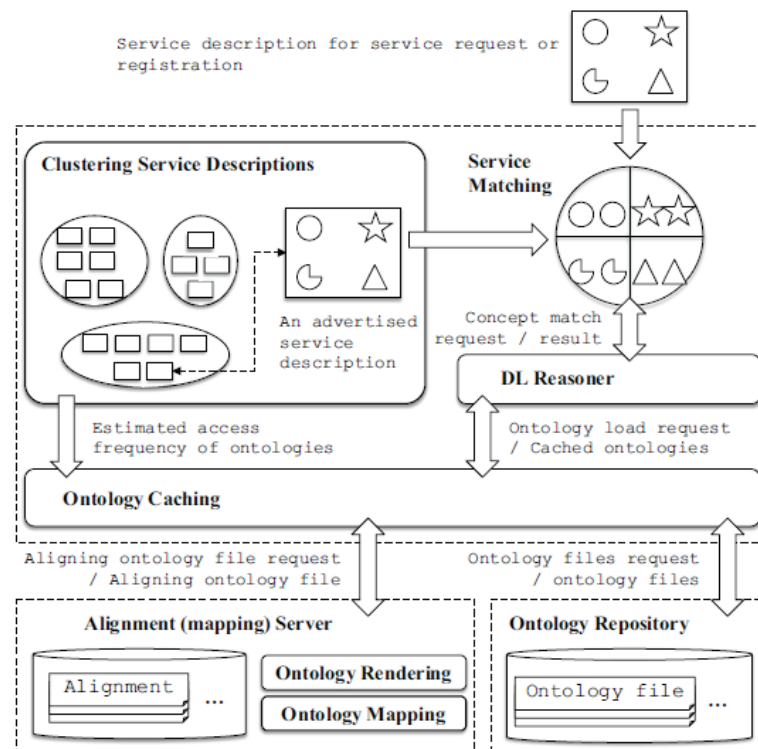


Abbildung 46: Prototypische Implementierung zum Umgang einer Vielzahl heterogener Ontologien, nach Kim und Lee (2010), S. 1540.

Der vorgestellte Algorithmus wurde von den Autoren mittels Java, der OWL-API 2.2 und dem Beschreibungslogik-Reasoner Fact++⁶² (Tsarkov und Horrocks (2006)) prototypisch implementiert. Eine Übersicht der implementierten Architektur findet sich in Abbildung 46. Als Alignment-Server wird die Arbeit von Euzenat (2004) eingesetzt, welcher sowohl das Rendering als auch das Mapping der Ontologien übernimmt. Ein

⁶² owl.man.ac.uk/factplusplus/

wichtiger Bestandteil ist die Caching-Komponente, es dem Reasoner erlaubt eine Vielzahl an Ontologien nach einem Entladen erneut zu laden. Das implementierte System wurde von den Autoren auf seine Leistungsfähigkeit und Antwortzeiten überprüft, allerdings nicht mit anderen Ansätzen verglichen. Eine standardisierte Testsammlung wurde nicht verwendet. (Kim und Lee (2010), S. 1539-1540)

3.4.3 Implementierung von WSMO-Mediatoren in GLUE

Konzeptionell wird im WSMO Ansatz das Problem der Heterogenität von Ontologien durch den Einsatz von Mediatoren gelöst. Eine kurze Erläuterung zu Mediatoren findet sich bereits in Abschnitt 2.2.3.2. Die verschiedenen Arten von Mediatoren sollen zunächst weiter detailliert werden, bevor ein Ansatz vorgestellt wird, der Mediatoren im Rahmen einer automatischen semantischen Service Discovery einsetzt.

Nach Roman et al. (2005) existieren vier Arten von Mediatoren. Die Bezeichnung der Mediatoren richtet sich nach den zu verknüpfenden Komponenten (Roman et al. (2005), S. 95ff):

- *OO-Mediatoren (OOM)*: OO-Mediatoren lösen Diskrepanzen zwischen Ontologien und erlauben so die Integration von verschiedenen Ontologien bei der Nutzung in anderen Komponenten. Quellen von OO-Mediatoren können Ontologien, aber auch andere OO-Mediatoren sein, welche integriert werden sollen. Formen der Integration sind Alignment, Merging und Mapping von Ontologien zur Herstellung einer homogenen Terminologie.
- *GG-Mediatoren (GGM)*: Ein GG-Mediator verbindet WSMO-Ziele (goals) untereinander. Aus bestehenden Zielen können somit neue Ziele generiert und eine Zielhierarchie erstellt werden. Die Erstellung neuer Ziele basiert auf der Reduktion oder Erweiterung bestehender Ziele, wobei auch OO-Mediatoren zur Lösung terminologischer Diskrepanzen zum Einsatz kommen können. Analog zu OO-Mediatoren können Quellen der GG-Mediatoren entweder Ziele oder andere GG-Mediatoren sein.
- *WG-Mediatoren (WGM)*: WG-Mediatoren werden verwendet, um Web Services mit vordefinierten Zielen (oder GG-Mediatoren) zu verbinden. Zusätzlich können funktionale Differenzen zwischen Web Service und Ziel beschrieben sowie terminologische Diskrepanzen mittels OO-Mediatoren eliminiert werden. Quellen können Web Services oder andere WG-Mediatoren sein.
- *WW-Mediatoren (WWM)*: WW-Mediatoren stellen die Interoperabilität von Web Services untereinander sicher. Dabei werden die Choreographien der Web Services aufeinander abgestimmt, so dass diese miteinander interagieren können. Mediation ist dabei unter Umständen auf Prozess-, Daten- und Protokollebene nötig. Wie auch bei den anderen Mediatoren können terminologische Unterschiede durch OO-Mediatoren eliminiert werden.

Ein Beispiel für den Einsatz verschiedener Mediatoren findet sich in Abbildung 47 und beschreibt die Buchung einer Zugfahrt. Ohne zu sehr ins Detail zu gehen ist sichtbar,

dass ein konkretes Ziel G_2 (Buchung einer Zugfahrt zwischen zwei Städten) durch ein generisches Ziel G_1 mittels eines GGM erstellt wird. Ein bestehender Web Service WS_{VTA} (virtuelle Reiseagentur) wird durch einen WGM an das Ziel gekoppelt und grenzt dabei die Funktionalität der Reisebuchung des WS_{VTA} auf eine Zugbuchung ein. Der WS_{VTA} verwendet zudem den Zahlungsservice $WS_{payment}$. Neben einem WWM für eine saubere Interaktion zwischen den zwei Services wird zudem ein OOM eingesetzt, da der $WS_{payment}$ eine andere Ontologie als der WS_{VTA} verwendet. (Roman et al. (2005), S. 96-97)

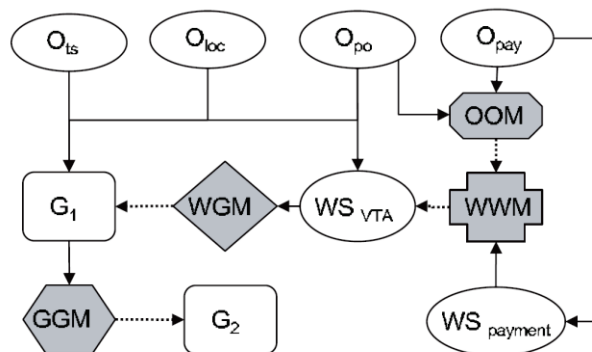


Abbildung 47: Beispiel der Nutzung von WSMO Mediatoren, nach Roman et al. (2005), S. 97.

Die Verwendung von Mediatoren ist in verschiedenen Szenarien einsetzbar. Interessant im Rahmen der vorliegenden Arbeit ist natürlich vornehmlich die Ermöglichung einer Service Discovery auf Basis unterschiedlicher semantischer Beschreibungen.

Ein mögliches Konzept und eine entsprechende prototypische Implementierung präsentieren Della Valle et al. (2005). Die Autoren unterscheiden unterschiedliche Zeitpunkte für die Erstellung und Nutzung von Mediatoren: *Set-up*, *Publishing* und *Discovery*. Für eine spätere automatische Discovery muss während des *Set-up* das System mit allen nötigen Informationen initialisiert werden. Dies beinhaltet vor allem das Wissen einer bestimmten Domäne D , welches in verschiedenen Formen und unterschiedlichen Sichtweisen repräsentiert werden kann. Eine solche Polarisierung kann zu multiplen Domänenverständnissen, D^+ und D^- , führen. Eine Möglichkeit zum Umgang mit diesen unterschiedlichen Sichtweisen ist der Versuch einen Konsens in Form einer gemeinsamen Ontologie zu finden. Da dies aber aus Erfahrung einen sehr hohen Aufwand erzeugen kann, schlagen Della Valle et al. (2005) einen Prozess vor, welcher unterschiedliche Ontologien definiert, Web Services und Ziele auf Basis dieser aufstellt und diese anschließend mit Hilfe von Mediatoren interoperabel macht. Der Prozess ist in Abbildung 48 dargestellt und soll als Grundlage für die nachfolgende prototypische Implementierung kurz besprochen werden. Das Wissen nach D^+ und D^- wird von entsprechenden Domänenexperten in den Ontologien Ont_{D^+} und Ont_{D^-} modelliert. Anhand der Konzepte dieser Ontologien werden zum einen Klassen von Web Services und zum anderen Klassen von Zielen definiert. Wichtig zu erwähnen ist, dass diese Klassen nur als Vorlagen

für spätere konkrete Ausprägungen der Ziele und Web Services dienen. Durch die Definition dieser Klassen ist es nicht nötig alle Konzepte zwischen den Ontologien mittels Mediatoren zu verbinden. Allerdings ein SWS Experte notwendig, welcher sowohl die Erstellung der Ontologien, Web Service-Klassen und Ziel-Klassen begleitet als auch für die Entwicklung der Mediatoren verantwortlich ist. Zur Entwicklung der Mediatoren (zum Einsatz kommen OOMs, GGMs und WGMs) unterstützt ihn ein unabhängiger Domänenexperte. (Della Valle et al. (2005), S. 38-39)

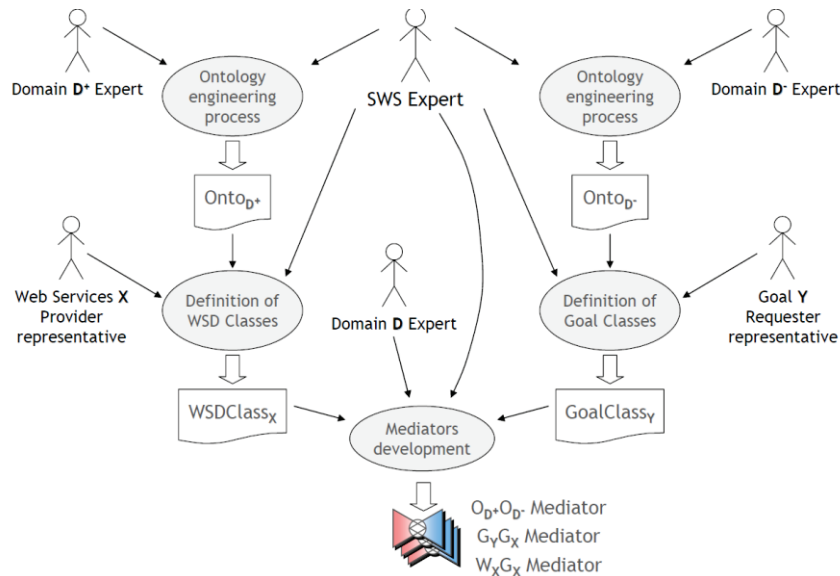


Abbildung 48: Set-up zur Erstellung von Mediatoren für den Umgang mit heterogenen Ontologien in der semantischen Service Discovery, nach Della Valle et al. (2005), S. 39.

In der anschließenden Publishing Phase veröffentlichen Anbieter Instanzen der Web Services Beschreibungen basierend auf den anfangs definierten Klassen. Danach stehen die Beschreibungen im Repository der Implementierung zur Verfügung und können von WGMs verwendet werden. Letztlich werden in der Discovery Phase die Zielklassen verwendet, um Zielinstanzen zu erstellen. Hierbei kommen unter Umständen auch GGMs zum Einsatz, um die vom Nutzer erstellten Ziele mit den bestehenden semantischen Zielen zu verbinden. Passende WGMs werden gesucht und die Web Service Beschreibungen können grob nach eventuellen Übereinstimmungen gefiltert werden. Nachdem eventuelle terminologische Diskrepanzen zwischen dem Ziel und Web Services mittels OOMs entfernt wurden, können präzise passende Web Services gefunden und priorisiert werden. (Della Valle et al. (2005), S. 40)

Der beschriebene Mechanismus wurde von den Autoren auf Basis der WSMX Architektur (siehe Abschnitt 3.2.2.3) prototypisch implementiert. Als Beschreibungssprache kommt F-Logic zum Einsatz, welche vom eingesetzten Reasoner FLORA-2⁶³ interpretiert werden kann. Im Fokus der Implementierung steht der *Proof Generator*, welcher in Glue für die Suche nach passenden Services unter Verwendung entsprechender Mediatoren zuständig ist. (Della Valle et al. (2005), S. 40-43)

3.4.4 Umgang mit heterogenen Ontologien in METEOR-S

In der METEOR-S Web Service Discovery Infrastructure wird die Servicesuche durch eine Erweiterung der UDDI Registry Spezifikation realisiert. Neben der traditionellen Stichwort- und Kategoriensuche von UDDI wird dadurch eine semantische Suche auf Basis von OWL-Konzepten unterstützt. Dies geschieht durch die Definition von Suchvorlagen (*engl.* search templates) (Sivashanmugam et al. (2003)) bestehend aus den Namen des Service und seiner Operationen sowie den In- und Outputs der Operationen. Die In- und Outputs können durch Referenzen auf ontologische Konzepte, realisiert zum Beispiel durch den Einsatz von SA-WSDL, erweitert werden und somit eine semantische Suche ermöglichen.

Oundhakar et al. (2005) erweitern den Ansatz der semantischen Suche auf Basis von Suchvorlagen um die Möglichkeit der Nutzung unterschiedlicher Ontologien für die Beschreibung von Suchanfrage sowie Service. Die Autoren nutzen dabei die Vorarbeit von Cardoso und Sheth (2003), welche bereits eine Ähnlichkeitsbestimmung anhand von syntaktischen und operationellen (basierend auf QoS) Eigenschaften. Diese Maße sollen erweitert werden um Kontext- und Abdeckungsähnlichkeit. (Oundhakar et al. (2005), S. 1-2)

Basierend auf der Anfrage mittels einer Suchvorlage werden syntaktische und funktionale Ähnlichkeit sowie die Konzeptähnlichkeit (annotierte Konzepte, zum Beispiel in OWL) zu einem Servicekandidaten berechnet. Oundhakar et al. (2005) verwenden hierbei keine bestehenden Implementierungen zur Ähnlichkeitsberechnung. Die syntaktische Ähnlichkeit untersucht keine semantischen Informationen und basiert allein auf den Namen und Beschreibungen von Suchvorlage und Servicekandidat. Hierfür wird der gewichtete Durchschnitt der Namens- und Beschreibungsähnlichkeit berechnet. Steht für den Service oder die Anfrage keine Beschreibung zur Verfügung, wird ausschließlich die Namensähnlichkeit verwendet. Zur Berechnung dieser wird eine Kombination aus verschiedenen Algorithmen wie N-Gram, Synonymbestimmung, Abkürzungserweiterung, Stemming und Tokenisierung (*engl.* tokenization) eingesetzt. Die Autoren beschreiben nicht detailliert, wie die Berechnung der Ähnlichkeit realisiert wird. Allerdings wird im Rahmen der Synonymbestimmung auf den Einsatz von WordNet (Miller (1995)) und für die Tokenisierung auf die Verwendung des Porter

⁶³ flora.sourceforge.net

Stemmers (Porter (1993)) hingewiesen. Die Namensähnlichkeit wird mit 1 (auf einer normalisierten Skala 0 bis 1) bestimmt, falls auch nur einer der Algorithmen eine Ähnlichkeit von 1 bestimmt. Ansonsten wird der Durchschnitt aller Ergebnisse verwendet, welche keine 0 als Ergebnis liefern. Nur für den Fall, dass alle Algorithmen eine Ähnlichkeit von 0 bestimmen, wird auch die gesamte Namensähnlichkeit auf 0 gesetzt. Anders als die Namensähnlichkeit wird die Beschreibungsähnlichkeit nur anhand eines N-Gram-Algorithmus bestimmt. (Oundhakar et al. (2005), S. 11-12)

Die funktionale Ähnlichkeit zwischen Suchvorlage und Servicekandidat ist die durchschnittliche beste funktionale Ähnlichkeit der Operationen. Für die beste funktionale Ähnlichkeit einer Operation werden die funktionalen Ähnlichkeiten aller möglichen Operationspaare zwischen Service und Anfrage berechnet. Die funktionale Ähnlichkeit eines Operationspaares wird auf Basis der syntaktischen Ähnlichkeit sowie der Input/Output- und Konzeptähnlichkeit bestimmt. Die syntaktische Ähnlichkeit der Operationen wird auf die gleiche Weise berechnet wie die der Services insgesamt (siehe vorheriger Abschnitt). Zur Bestimmung der Ähnlichkeit von Input und Output werden Eingangs- und Ausgangsparameter zunächst unabhängig voneinander untersucht. Soweit beide in der Anfrage vorhanden sind, wird ihre Ähnlichkeit am Ende über ihr arithmetisches Mittel berechnet. Die weitere Berechnung der Ähnlichkeit verwendet eine rekursive Berechnung der Konzeptähnlichkeit (siehe nächster Abschnitt) der einzelnen In- beziehungsweise Outputs unter Nutzung der dynamischen Programmierung. (Oundhakar et al. (2005), S. 12-13)

Die Konzeptähnlichkeit basiert auf den annotierten ontologischen Konzepten der Operationen. In diesem Kontext wird von Oundhakar et al. (2005) auf bestehende Ansätze im Bereich des Ontology Matchings hingewiesen. Nach Meinung der Autoren (zum Zeitpunkt der Publikation) bestehen Probleme bei der Berechnung der Ähnlichkeit anhand von Namen und Range der Konzepte. Eine Nutzung von Subsumptionsbeziehungen zur Ähnlichkeitsberechnung schließen die Autoren aus, da dies ihrer Meinung nach nur unter Verwendung einer einzelnen Ontologie möglich sei. Um die Ähnlichkeitsberechnung von Konzepten zu verbessern untersuchen die Autoren neben der Ähnlichkeit von Syntax und Eigenschaften auch Abdeckung und Kontext. Die syntaktische Ähnlichkeit wird wie anhand der vorgestellten Techniken zur Bestimmung der Namensähnlichkeit berechnet. Die Ähnlichkeit der Eigenschaften der verwendeten Konzepte untersucht Syntax, Range (Vergleich von Datentypen oder Konzepten) und Kardinalität. In der sogenannten Abdeckungsähnlichkeit (*engl. coverage similarity*) wird gemessen, wie viel Wissen ein Konzept „abdeckt“, indem die Super-Konzepte der zu vergleichenden Konzepte betrachtet werden. Kontextähnlichkeit wird gemessen, indem Sets an Konzepten unterschieden werden, in welchen gleichartige oder divergierende Konzepte zusammengefasst sind. (Oundhakar et al. (2005), S. 13-19)

4 Geschäftsprozessorientierte Service Discovery

Nachdem der Stand der Forschung in den für das Konzept relevanten Bereichen dargelegt wurde, wird im Folgenden ein neuer Ansatz zur Beschreibung und Suche von Services vorgestellt. Der Ansatz nutzt eine fachliche und darin im speziellen geschäftsprozessorientierte Beschreibung von Services, unabhängig von den unterliegenden Implementierungen.

Fachliche Service-Beschreibungen helfen während Entwicklung und Implementierung serviceorientierter Architekturen im Rahmen von Unternehmensarchitekturen. So identifizierten Forscher der Universität St. Gallen die fachliche Servicespezifikation als eine der Herausforderungen für SOAs in Partnernetzen, insbesondere der Automobilindustrie. (Legner (2006), S. 10)



Abbildung 49: SOA in der Automobilindustrie.

Der vorgestellte Ansatz zielt zwar vornehmlich auf den Einsatz in SOAs ab, die fachliche Abstraktion der Beschreibungen erlaubt aber prinzipiell den Einsatz in anderen Schnittstellenarchitekturen wie zum Beispiel der traditionellen Komponentenorientierung. Da die Evaluation des Ansatzes allerdings ausschließlich in Projekten zu serviceorientierten Architekturen durchgeführt wurde, empfiehlt der Autor vor einem Einsatz

des Konzeptes in anderen Schnittstellenarchitekturen eine zusätzliche Evaluierung anhand der entsprechenden Anforderungen. Eine Evaluierung des Ansatzes in Architekturen außerhalb der SOA würde den Umfang dieser Arbeit übersteigen.

Die Beschreibung von Services durch die Fachseite stellt besondere Ansprüche an die Semantik, welche in den nachfolgenden Abschnitten identifiziert werden sollen.

4.1 Geschäftsprozessorientierte Service Beschreibungen

Die Beschreibung von Services auf Basis der Geschäftsprozesse eines Unternehmens erlaubt es Fachservices zu definieren, welche den Nutzen einer Wiederverwendung steigern und welche die Anzahl an aufzurufenden Services verringern (Heutschi (2007), S. 43 ff).

Basierend auf dem aktuellen Stand der Forschung (siehe Kapitel 3) und der Nutzung von Expertenwissen aus Forschungsprojekten mit großen internationalen Industriepartnern soll im Folgenden ein erweiterter Ansatz der Servicebeschreibung vorgestellt werden. Die funktionale Beschreibung der Services steht hierbei im Vordergrund, da die Entwicklung von nichtfunktionalen Beschreibungen bereits sehr weit fortgeschritten ist (siehe Schröpfer (2009)) und zudem den Rahmen der vorliegenden Arbeit übersteigen würde.

Für die Entwicklung eines Rahmenwerks zur Servicebeschreibung muss zunächst identifiziert werden, welche Klassen von Services sich zur Beschreibung mittels Geschäftsprozessinformationen eignen. Daher wird zunächst eine Klassifizierung von Services diskutiert, auf deren Basis weitere relevante Informationen für die Beschreibung abgeleitet werden können. Anschließend werden Beschreibungsformen für die Daten und die Funktionalität der Services abgeleitet. Abschließend sollen exemplarisch Services mit den Beschreibungssprachen OWL-S, WSMO und SAWSDL unter Verwendung der entwickelten Konzepte vorgestellt werden.

4.1.1 Serviceklassifizierung

Eine Klassifizierung von Services ist notwendig, um die Anforderungen für die spätere Beschreibung reduzieren und Informationen im Sinne einer Geschäftsprozessorientierung ableiten zu können. Die vorliegende Arbeit verwendet hierzu die Klassifikation von Offermann (2009), welche in Abbildung 50 zu sehen ist. Diese basiert unter anderem auf den Arbeiten von Krafzig et al. (2004) und Erl (2005). Dabei werden neben dem Einsatzszenario (B2B-Integration, Enterprise Application Integration, unternehmensweite Softwarearchitektur) und dem Ursprung (Externes Hosting, Software von Fremdanbietern, Altsystem, neue Implementierung) vor allem verschiedene Formen von Servicetypen definiert (Offermann (2009), S. 31-32):

- *Geschäftsprozessservice*: Diese Art von Service besteht meist auf einer Orchestrierung bestehender Services zur Unterstützung eines ganzen oder größerer

Teile eines Geschäftsprozesses. Offermann (2009) bezeichnet dies auch als Workflow.

- *Benutzerschnittstelle (Präsentation)*: Diese Schnittstelle ist die Möglichkeit zur Interaktion mit dem Service und wird in Form einer GUI implementiert. Sie ist daher nicht als klassischer (Web) Service zu sehen.
- *Geschäftslogikservice*: Hierbei handelt es sich, anders als bei der Orchestrierung von Services zur Unterstützung eines Geschäftsprozesses, um einen Service zur Umsetzung einer einzelnen Geschäftsaktivität. Davon auszunehmen sind Formen der Datenverwaltung (siehe unten), da es sich hierbei um reine Datenverarbeitung handelt.
- *Datenverwaltungsservice*: Diese Form von Services verwaltet Geschäftsdaten. Hierbei liegt der Fokus auf Geschäftsobjekten, deren Eigenschaften in den anschließenden Abschnitten noch weiter diskutiert werden.
- *Infrastrukturservice*: Ein Infrastrukturservice ist zum Betrieb der anderen Servicearten notwendig. Es existiert dabei keine direkte fachliche Beziehung. Beispiele sind die Transaktionsverwaltung oder die Überwachung von Sicherheitsrichtlinien.

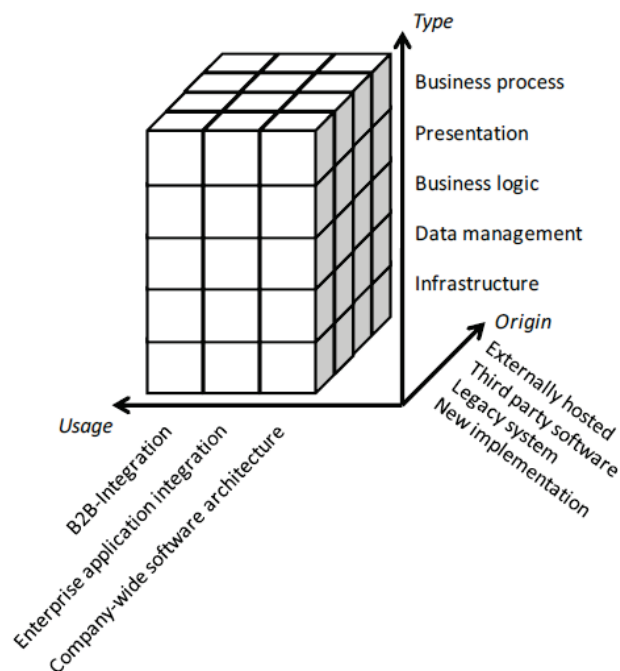


Abbildung 50: Klassifizierung von Services nach Offermann (2009), S. 31.

Für den vorliegenden Ansatz werden im folgenden Abschnitt die verschiedenen Servicearten auf die Notwendigkeit einer Beschreibung für die spätere geschäftsprozessorientierte Service Discovery (GSD) untersucht sowie entsprechende Anforderung an eine

solche Beschreibung abgeleitet. In diesem Zusammenhang ist es nach Meinung des Autors nicht relevant, welches Einsatzszenario (Serviceverwendung) für den Service angedacht ist oder welchen Ursprung der Service hat. Allerdings kann es hilfreich sein für diese weiteren Dimensionen in der Zukunft zusätzliche Beschreibungen zu definieren, um weitere Optionen für die Suche zu erschließen (zum Beispiel in Form von Filtern).

4.1.2 Ableitung relevanter Informationen

Das Ziel der angestrebten geschäftsprozessorientierten Service Discovery ist das Auffinden von Services zur Unterstützung von Geschäftsprozessen. Um gute Ergebnisse zu erreichen, müssen diese für die Nutzer der Geschäftsprozesssicht relevant sein. Obwohl die Services technische Implementierungen der fachlichen Logik sind, sollten die Resultate vor allem aus fachlicher Sicht relevant sein. Mögliche Zielgruppen, welche für die Nutzung der GSD in Frage kommen, sind unter anderem Geschäftsprozessmodellierer (Business Process Modeler), Geschäftsprozessexperten (Business Process Experts) und Unternehmensarchitekten (Enterprise Architects). Das Auffinden relevanter Services hängt somit stark von der verständlichen Beschreibung der Services an sich und der eigentlichen Suchanfrage ab. Daher soll im Folgenden abgeleitet werden, welche Informationen im Kontext der Geschäftsprozessmodellierung vorhanden und vor allem relevant für eine effiziente und effektive Beschreibung und somit nachfolgend auch Suche der Services nötig sind.

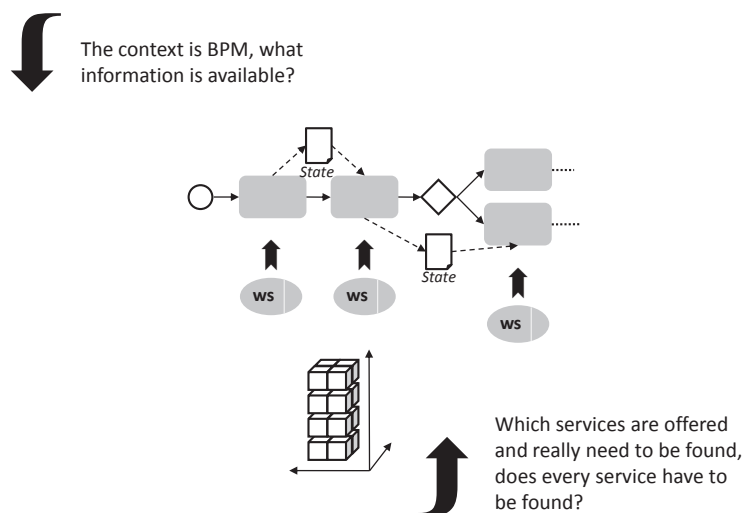


Abbildung 51: Ableitung relevanter Informationen aus dem Geschäftsprozessmodellierungskontext.

Eine Analyse der bestehenden Informationen im GPM Kontext wurde sowohl „Top-Down“, aus Sicht der Geschäftsprozessmodelle, als auch „Bottom-Up“, aus Sicht der für die Geschäftsprozesse relevanten Servicekategorien, durchgeführt (siehe Abbildung 51).

Hilfreich bei der Analyse, beginnend mit der „Top-Down“ Sicht, ist die Tatsache, dass Geschäftsprozessmodelle von sich aus bereits eine implizite Serviceorientierung beinhalten. Die Transformation oder Manipulation einer gegebenen Eingabe (Input) im Rahmen von einer Reihe an Aktivitäten (Tasks) führt zu einer gewünschten Ausgabe (Output). Ausgehend von verschiedenen Formen und Perspektiven der Modellierung (siehe Beispiel in Abbildung 52 in der BPMN) können folgende relevante Informationen für die mögliche Beschreibung des Service (beziehungsweise der Serviceanforderungen) identifiziert werden:

- Geschäftsprozessaktivitäten oder –schritte
- Daten- und/oder Geschäftsobjekte (beziehungsweise deren IDs⁶⁴)
- Statusinformation der Daten- und Geschäftsobjekte
- Ereignisse

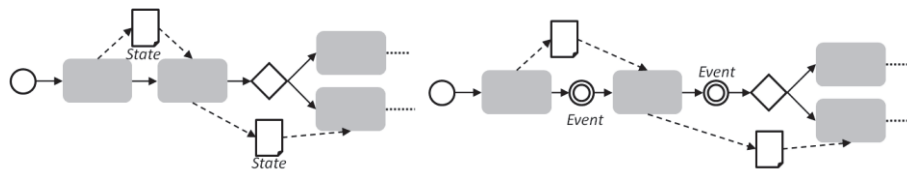


Abbildung 52: Informationen in Varianten der Prozessmodellierung am Beispiel BPMN.

Der zweite Schritt der Analyse, „Top-Down“, basiert auf der Identifizierung relevanter Serviceklassen für den Kontext der Geschäftsprozessmodellierung. Hierbei kann die bereits vorgestellte Klassifizierung von Services aus Abschnitt 4.1.1 (siehe Abbildung 50) genutzt werden.

⁶⁴ IDs ersetzen beziehungsweise repräsentieren oft reale Daten- und Geschäftsobjekte in der Modellierung und späteren Implementierung. Dies läuft zwar dem Prinzip der losen Kopplung entgegen, ist aber in der Praxis häufig notwendig.

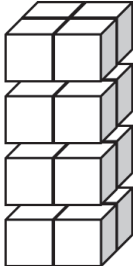
		Relevance for Business Process Experts	Reason
	Business Service	High	Business functionality directly relates to process steps/activities
	UI Service	Medium*	Generally paired with business or database service
	BO Access Service	High	Allows access to business objects
	Technical Service	Low	Does not represent business functionality, too fine grained

Abbildung 53: Relevanz verschiedener Serviceklassen für die GPM.

Die Auswertung der Relevanz der Serviceklassen für die Geschäftsprozessmodellierung erlaubt eine Abschätzung über die nötigen Informationen, welche in der späteren Servicebeschreibung vorhanden sein müssen. Somit lassen sich vor allem Fachservices (Business Services) als auch Geschäftsobjektzugriffservices (BO Access Services) als solche Services identifizieren, welche auf jeden Fall mit der zu entwickelnden Servicebeschreibung abgedeckt werden müssen. Services zur Bereitstellung der Benutzerinteraktion (UI Services) sind zwar relevant, können aber entweder bereits mit einem bestehenden Service gepaart oder dynamisch generiert. Technische und Infrastruktur Services (Technical Services) sind nicht relevant für unsere Betrachtung, da sie zu feingranular für die Orchestrierung auf der Geschäftsprozessebene sind.

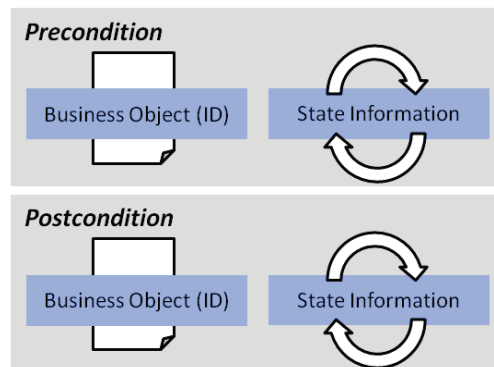


Abbildung 54: Nutzung von Vorbedingungen und Effekten zur Servicebeschreibung

Durch die vorangegangene Analyse lässt sich feststellen, dass vor allem Geschäftsobjekte (oder eine Referenz mittels der ID) und deren Statusinformationen nützliche Beschreibungen im GPM-Kontext zulassen. In Anlehnung an bestehende Ansätze der semantischen Servicebeschreibung (siehe Abschnitt 2.2.3) sollen diese Informationen zur Be-

schreibung der Vor- und Nachbedingung eines Service genutzt werden, um die Discovery im vorliegenden Kontext nutzen zu können. Es soll noch erwähnt werden, dass eine Mehrzahl der Tupel aus Geschäftsobjekt und Statusinformationen für die Beschreibung der jeweiligen Vorbedingung und des jeweiligen Effekts genutzt werden können, wie auch in späteren Beispielen illustriert wird.

4.1.3 Beschreibung der Geschäftsobjekte

Die kontrollierte Beschreibung der Daten von Software-Services, -Schnittstellen und -Komponenten auf der Unternehmensebene lässt sich zurückverfolgen auf das Ende der achtziger und den Beginn der neunziger Jahre (siehe u.a. Scheer (1990)). Zu Beginn wurden diese Beschreibungen vor allem in Form von Stichwortverzeichnissen oder Glossaren geführt. Diese konnten genutzt werden, um einheitliche Benennung und eindeutige Identifikation von Objekten zu ermöglichen. Mit der fortschreitenden Automatisierung vor allem im Bereich der Geschäftsprozessausführung begannen Sammlungen (Repositories) von Datenobjekten und Prozessen zu entstehen (Thomas (2006)). Zur Verwendung und Pflege solcher Repositories ist es sinnvoll, die verwendeten Begriffe unternehmensweit oder zumindest domänenweit zu vereinheitlichen und zentral zugänglich zu machen.

Der vorliegende Ansatz zielt darauf ab, bestehende Begriffe einer Domäne oder eines Unternehmens so weit wie möglich für die Beschreibung der Services wiederzuverwenden. Eventuell entstehende Heterogenität zwischen Unternehmen und Domänen können anschließend durch ein automatisiertes Matching überbrückt werden. Wie bereits in Abschnitt 4.1.2 dargestellt basiert ein Teil der Beschreibung auf dem Konzept der Geschäftsobjekte, welches im Folgenden weiter vertieft werden soll. Zudem wird dieses Konzept von Ein- und Ausgabeparametern abgegrenzt und erklärt, wie diese im Rahmen der Servicesuche behandelt werden können.

4.1.3.1 Geschäftsobjekte und Geschäftsartefakte

„Geschäftsobjekte (Business Objects) sind die für die Leistungserstellung einer Unternehmung relevanten diskreten, unterscheidbaren Entitäten.“
(Nüttgens und Zimmermann (1998), S. 28)

“A business object is a representation of a thing active in the business domain, including at least its business name and definition, attributes, behavior, relationships, rules, policies and constraints. A business object may represent, for example, a person, place, event, business process, or concept. Typical examples of business objects are: employee, product, invoice and payment.” (Ohne Autor (1996), S. 19)

In den 1990er Jahren wurde durch die Umstellung auf objektorientierte Modellierung und Programmierung ein stärkeres Augenmerk auf die Nutzung fachlicher Komponenten in Form von Anwendungs- oder Geschäftsobjekten (engl. Business Objects) gelegt

(Weske (1999)). Grundlage dieser Entwicklung ist unter anderem die Konzeption von Geschäftsobjekten für die Informationsmodellierung in betrieblichen Anwendungen (Ferstl und Sinz (1990)). Die Möglichkeit der Nutzung des Konzepts der Geschäftsobjekte als verteilte Komponenten mit inhärenter Logik, Regeln und Methoden soll allerdings nicht im Fokus stehen. Vielmehr ist es für die semantische Beschreibung der Objekte in einem Geschäftsprozess und der weiteren Nutzung durch Services wichtig, eine beherrschbare Granularität zu identifizieren. Da diese Granularität je nach Unternehmung und Anwender variieren kann ist es hilfreich, Geschäftsobjekte als Artefakte mit eigenem Lebenszyklus und Informationsmodell zu verstehen (Nigam und Caswell (2003); Cohn und Hull (2009)).

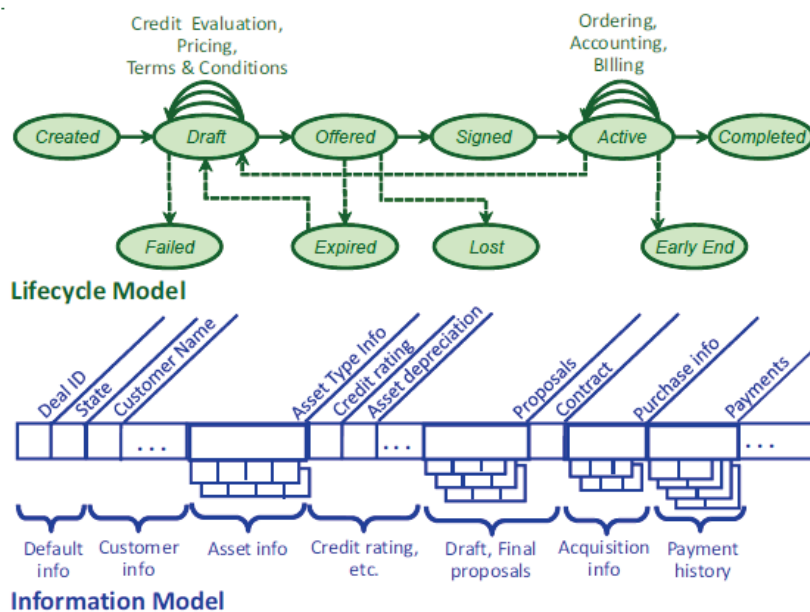


Abbildung 55: Beispielhaftes "Deal" Geschäftsartefakt nach Cohn und Hull (2009), S. 5.

Die Kombination von Informations- und Lebenszyklusmodell erlaubt auch eine zusätzliche Beschreibung der Funktionalität von Services in Form von Statusinformationen, welche in Abschnitt 4.1.4.1 weiter detailliert wird. Dem Aspekt der Dualität von Prozessmodellen (informationszentrisch und aktivitätszentrisch, siehe Bellahsene et al. (2008)) wird auf diese Weise auch Rechnung getragen.

Eine weiterer Grund zur Nutzung von Geschäftsobjekten für die Beschreibung von Services im Kontext der Prozessorientierung sind die Vorgaben verbreiteter Modellierungssprachen und die Empfehlungen zur Modellierung von Prozessen. So empfehlen Mendling et al. (2010) unter anderem die Nutzung einer Kombination aus Verb und Objekt zur Beschreibung von Aktivitäten. So kann zwischen der Servicebeschreibung aus Geschäftsobjekten (und Statusinformationen) eine Verbindung zu den Objekten

der Geschäftsprozessmodellierung (und oft deren Status anhand des Verbs) hergestellt werden.

Durch die Verwendung von Ontologien und Repräsentation dieser durch formale Beschreibungssprachen wie OWL oder WSMO ist es möglich, Hierarchien der Geschäftsobjekte zu bilden sowie Eigenschaften und Regeln bezüglich der Objekte zu definieren. Ein Beispiel (siehe Abbildung 56) hierzu ist das Geschäftsobjekt *Geschäftspartner* (BusinessPartner) welches sowohl in Form von *Kunden* als auch *Lieferanten* existieren kann. Zudem können *Geschäftskunden* und *Privatkunden* existieren. Diese Formen der Hierarchie erlaubt eine sinnvolle Definition und Beschreibung der Services, indem spezialisierte Services auf den unteren Ebenen und generische Services auf den oberen Ebenen entstehen. Weiterhin wird auch die Suche nach den Services vereinfacht. Eine Suche nach spezialisierten Konzepten kann unter Umständen durch generische Services erfüllt werden. Falls nach generischen Konzepten gesucht wurde können in bestimmten Fällen durch den Nutzer spezialisierte Services auch verwendet werden. Aus bestehenden Geschäftsobjekten können sich weitere Objekte ableiten beziehungsweise abhängig sein. Die Unterscheidung in *Lieferant* und *Kunde* macht eventuell die Einführung einer weiteren Unterteilung des Konzeptes *Rechnung* in *Kundenrechnung* und *Lieferantenrechnung* nötig. Formale Beschreibungssprachen ermöglichen dabei die Verknüpfung der Konzepte unter Beachtung von bestimmten Regeln. So kann kontrolliert werden, dass das Konzept *Kundenrechnung* nicht mit einem *Lieferant* als Schuldner verknüpft wird.

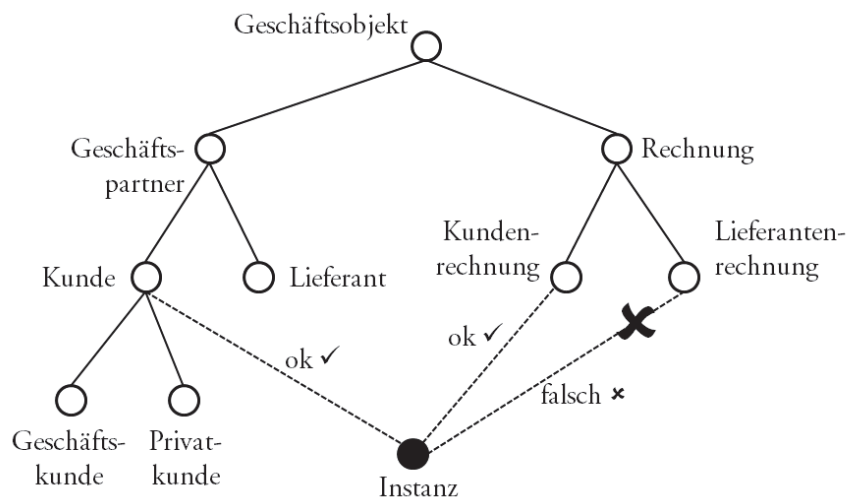


Abbildung 56: Beispiel zur Verwendung von Geschäftsobjekten und deren Semantik

Die oben genannten Beispiele werden im Verlauf der Beschreibung des Ansatzes weiterverwendet, um ein besseres Verständnis anhand eines praktischen Beispiels zu entwi-

ckeln. Angelehnt aber nicht unbedingt identisch sind die verwendeten Konzepte an die öffentlich verfügbaren Geschäftsobjekte der SAP AG⁶⁵. Die Sammlung von Geschäftsobjekten ist dabei ein gutes Beispiel einer umfangreichen Aufstellung von zur Beschreibung verwendbaren Konzepten.

Andere Unternehmen setzen zudem auf eigens entwickelte Objektmodelle, zum Beispiel verwendet die Deutsche Telekom AG ein eigenes „Business Object Model“ um ihre zentralen Geschäftsobjekte zu beschreiben und zu managen (Schmidt et al. (2010), S. 19ff). Dieses wird als semantisches Modell bezeichnet und beschreibt die Eigenschaften und Beziehungen der Geschäftsobjekte. Die definierten Begriffe sollen dabei möglichst allgemeingültig und redundanzfrei sein. Weiterhin sind die Begriffe in ein sogenanntes Domänenmodell eingeordnet, welches verschiedene Bereiche einer Unternehmung beschreibt und das Management von Geschäftsobjekten (und auch Services) vereinfachen soll. Das BOM steht als UML-Klassenmodell zur Verfügung. Eine direkte Weiterverwendung für die Beschreibung semantischer Services wäre allerdings somit nicht möglich. Es müsste zunächst einen Transfer der Konzepte in eine passende Ontologiesprache unter Einbeziehung von Ontologieingenieuren und Domänenexperten geleistet werden. Dabei müssten vor allem die Hierarchien, Abhängigkeiten und Regeln innerhalb des semantischen Modells überprüft werden. Nichtsdestotrotz würde ein solches bestehendes Modell die Beschreibung von Services im Sinne des vorliegenden Ansatzes unterstützen und den initialen Aufwand stark reduzieren.

Neben bestehenden unternehmensinternen Geschäftsobjektmodellen existieren auch Standardisierungsinitiativen einzelner Industrien zur Vereinheitlichung des semantischen Verständnisses der verwendeten Konzepte. Ein Beispiel hierfür ist die enhanced Telecommunication Operations Map (eTOM) der Telekommunikationsindustrie (Kelly (2007)). In dieser werden Referenzprozesse für alle Domänen eines Telekommunikationsunternehmens definiert. Dies beinhaltet gleichzeitig die Definition semantisch (meist) eindeutiger Geschäftsobjekte, welche neben der Beschreibung der Prozesse auch in Teilen für die Beschreibung von Services einsetzbar sind. Diese Eignung wird durch die Bemühungen der Wissenschaft unterstützt eTOM in Form von Ontologiesprachen bereitzustellen und für die Modellierung von Prozessen sowie die Beschreibung von Services (zum Beispiel in der Klassifizierung) zu verwenden (siehe unter anderem Shangquan et al. (2008) und Duke et al. (2005)).

Die Geschäftsobjekte zur Beschreibung von Services in der GSD werden, wie in Abschnitt 4.1.2 beschrieben, sowohl in Vor- als auch in Nachbedingungen eingesetzt. Ansätze zur Beschreibung von Services geben generell keine Vorgaben, wie eine Vor- oder Nachbedingung definiert werden soll. Um eine möglichst umfassende Beschreibung der Funktionalität der Services zu erreichen und somit eine geschäftsprozessorientierte Suche zu ermöglichen, können im vorliegenden Ansatz mehrere Geschäftsobjekte und dazugehörige Statusinformationen verwendet werden. Details zur Beschreibung der Funk-

⁶⁵ Abrufbar im Enterprise Service Workplace unter <http://esworkplace.sap.com>.

tionalität sowie Beispiele anhand bestehender Servicebeschreibungssprachen folgen im Abschnitt 4.1.5. Zunächst soll allerdings eine Abgrenzung zu den Ein- und Ausgabeparametern der Services beziehungsweise der Servicebeschreibungen folgen.

4.1.3.2 Abgrenzung zu Eingabe- und Ausgabeparametern

Die Beschreibung von Services mittels Geschäftsobjekten (und Statusinformationen) erlaubt die effektive und effiziente Suche während der Geschäftsprozessmodellierung. Hier muss jedoch differenziert werden zu der Beschreibung mittels Eingabe- und Ausgabeparametern der Services. Viele der bestehenden Ansätze zur Service-Suche nutzen diese Parameter bereits auf unterschiedlichen Ebenen der Granularität. Hierbei reichen die Ansätze von einfachen semantischen Ähnlichkeiten der Namen der Objekte (zum Beispiel *Vertrag* oder *Rechnung*) bis hin zur Nutzung von Tree-Matching-Algorithmen auf den XSDs der Daten (vgl. Elhadad et al. (2008)).

Im vorliegenden Ansatz ist die Beschreibung der Eingabe- und Ausgabeparameter auf Geschäftsobjektebene vorgesehen. Daher ist es wichtig, dass die genutzten Geschäftsobjekte auch eine Basis im Objektdatenmodell des Unternehmens besitzen. Allerdings ist eine Unterscheidung zwischen den Ein- und Ausgabeparametern und den Geschäftsobjekten für die Beschreibung der Vor- und Nachbedingungen des Service notwendig. Dies liegt vor allem daran, dass es nach Ansicht des Autors nicht möglich ist, die Funktionalität eindeutig durch die reine Beschreibung von Ein- und Ausgabeparametern zu definieren. Ein einfaches Beispiel soll dieses Argument unterstützen:

Es soll ein Service beschrieben werden, welcher für die elektronische Versendung einer Rechnung zuständig ist. Eine einfache Variante dieses Service erhält ein Datenobjekt *Rechnung* und gibt ein Datenobjekt *Statusmeldung* zurück, welches den Erfolg oder Misserfolg der Sendung meldet. Würde man den Service nur anhand seiner Ein- und Ausgabeparameter beschreiben oder suchen, würden auch Services mit völlig anderer Funktionalität gefunden werden. Ein Beispiel wäre ein zweiter Service, welcher überprüft, ob eine Rechnung fehlerfrei ist, um sie an einen Kunden zu senden. Beschreibt man den ersten Service allerdings mittels Vor- und Nachbedingung anhand von Geschäftsobjekten, so lässt sich zum einen überprüfen, ob eine Rechnung überhaupt gesendet werden darf (zum Beispiel wenn der *Auftrag* abgeschlossen ist). Zum anderen unterscheiden sich die Nachbedingungen zwischen beiden Services eindeutig, da im ersten die *Rechnung gesendet* und im zweiten die *Rechnung validiert* ist.

Weiterhin ist die Beschreibung der Ein- und Ausgabeparameter mittels derselben Ontologiekonzepte unter Umständen problematisch. Dies liegt in der Auslegung der Subsumption innerhalb einer Ontologie. Wird zum Beispiel ein Geschäftsobjekt auf einer hohen Granularitätsebene (wie *Geschäftspartner*) im vorliegenden Ansatz zur Beschreibung eines Services verwendet, so drückt dies aus, dass dieser Service auch mit Subkonzepten dieses Geschäftsobjektes umgehen kann. Im Gegensatz dazu ist es in Datenhierarchien eventuell möglich, dass auf der unteren Ebene mehr Informationen in einem Datenobjekt vorhanden sind, wodurch eine Kompatibilität bei Services mit Ein-

gangsparametern höherer Granularität eventuell nicht gegeben ist. Daher entkoppelt der vorliegende Ansatz diese Beschreibungen. Im Weiteren soll aufgrund der Vielzahl der bestehenden Ansätze, welche sich auf die Suche nach Service anhand Ein- und Ausgangsparametern beschränken (siehe Abschnitt 3.1.2), dieser Bereich nicht weiter vertieft werden. Allerdings ist es möglich, die Konzepte zum Umgang mit der Heterogenität von Ontologien (siehe Abschnitt 3.3) in der Zukunft auf eine Service Discovery anhand von Ein- und Ausgabeparametern zu transferieren. Dies übersteigt im Moment allerdings den Umfang der vorliegenden Arbeit.

4.1.4 Beschreibung der Funktionalität

Die alleinige Verwendung von Geschäftsobjekten für die geschäftsprozessorientierte Beschreibung von Services wäre nicht ausreichend für eine eindeutige Identifikation und effiziente Suche. Zum einen würden ähnliche Probleme wie bei einer alleinigen Beschreibung mittels Eingangs- und Ausgangsparametern (siehe oben) entstehen, indem viele Services unterschiedlicher Funktionalität aber mit gleichen Geschäftsobjekten existieren würden. Zum anderen wäre die Beschreibung von Vorbedingungen und Effekten ohne zusätzliche Information über die Geschäftsobjekte nicht möglich. Daher verwendet der vorliegende Ansatz, wie bereits erwähnt, Statusinformationen in Verbindung mit den Geschäftsobjekten. Im den folgenden Abschnitten wird zunächst das Konzept der Statusinformation erklärt und durch bestehende Publikationen verfeinert. Anschließend wird im Detail erklärt, wie die Statusinformationen zu Beschreibung von Vorbedingungen und Effekten der Services verwendet werden können. Dies wird in den direkten Kontext der Geschäftsprozessmodellierung gesetzt. Letztlich sollen zudem eventuelle Einschränkungen dieser Art der Beschreibung diskutiert werden.

4.1.4.1 Statusinformationen

Zur Beschreibung der Funktionalität sollen zusätzliche Statusinformationen mit den Geschäftsobjekten verbunden werden. Dieser Ansatz basiert auf der Vorarbeit von Born et al. (2007) zur benutzerfreundlichen semantischen Annotation von Geschäftsprozessen (siehe auch Abschnitt 3.2.1.2) sowie dem Verständnis von Geschäftsobjekten und deren Lebenszyklen nach Cohn und Hull (2009). Allerdings sollen weiterhin die Prozesse und nicht die Daten, wie von Cohn und Hull (2009) gefordert, im Vordergrund stehen. Beide Ansätze sind ursprünglich für die Beschreibung und Vervollständigung von Geschäftsprozessen und den damit verbundenen Geschäftsobjekten entstanden. Der vorliegende Ansatz überträgt diese Konzepte anhand der Überlegung, dass Aktivitäten eines Geschäftsprozesses, wie auch Services, eine Form einer „Black Box“ sind. Dies erlaubt es, den Effekt (beziehungsweise das Ergebnis der Aktivität) mit den Effekten der Services zu vergleichen. Gleiches gilt für eventuell vorhandene wenn auch nicht immer notwendige Vorbedingungen für die Aktivität.

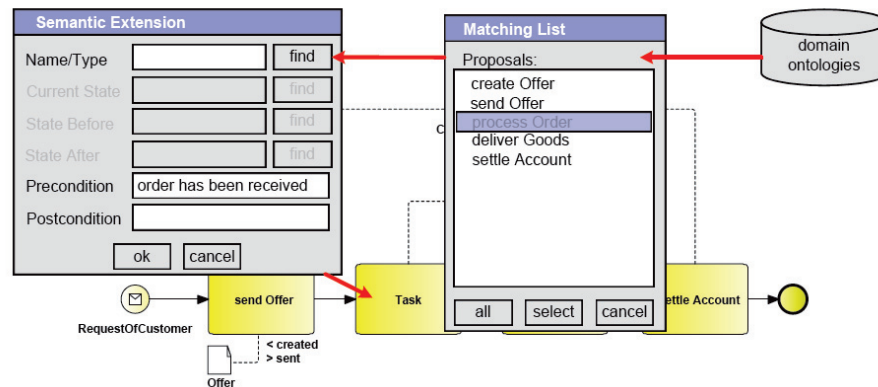


Abbildung 57: Verwendung von Datenobjekten und deren Status für die Beschreibung von semantischen Geschäftsprozessmodellen, nach Born et al. (2007), S. 265.

Born et al. (2007) verwenden neben der Beschreibung von Vor- und Nachbedingungen (welches im vorliegenden Kontext als Effekt verstanden werden kann) in Form von Datenobjekt und Status die Idee einer Transition zwischen den Status der Datenobjekte. Diese Transitionen beschreiben den Lebenszyklus des Objektes, wie ihn auch Cohn und Hull (2009) (siehe auch Abbildung 55) gezeigt haben. Allerdings verwenden Born et al. (2007) in ihrem Beispiel nur einen linearen Lebenszyklus, nach Cohn und Hull (2009) können innerhalb des Lebenszyklus auch Rekursionen, weitere Zyklen sowie Iterationen entstehen.

Im Rahmen zweier Forschungsprojekt mit unterschiedlichen Industriepartnern aus der Software- und Telekommunikationsindustrie wurde die Idee der Kombination von Objekten und deren Status zur geschäftsprozessorientierten Beschreibung von Services diskutiert und evaluiert. Das Ergebnis dieser Diskussion mit dem Ziel einer Grundlage für eine prototypische Implementierung kombiniert die oben erwähnten Ansätze und variiert sie geringfügig. Die beschriebenen Objekte sollen nicht die Datenobjekte (nach Born et al. (2007)), sondern die Geschäftsobjekte des Unternehmens oder auch des Serviceanbieters referenzieren. Zudem existiert nach Born et al. (2007) eine strikte Kopplung der Objekte an die jeweiligen Status. Diese Kopplung soll im vorliegenden Ansatz nicht verwendet werden, um eine Wiederverwendung von Statuskonzepten (zum Beispiel *gesendet*, *abgeschlossen*, *verifiziert*) zu ermöglichen. Eine spätere Kopplung im Rahmen einer Implementierung zur Unterstützung des Benutzers während der Suche und der Modellierung ist allerdings weiterhin durch den Einsatz von Regeln innerhalb der verwendeten Ontologie möglich. Regeln beziehungsweise Axiome innerhalb der Ontologien können auch die Transitionen zwischen den Status der Geschäftsobjekte beschreiben. Für die eigentliche Beschreibung der Services ist die Verbindung zwischen den Status und deren Transition allerdings nicht relevant. Diese Informationen unterstützen vielmehr die geschäftsprozessorientierte Service Discovery aus einer Modellierungsumgebung heraus (siehe Abschnitt 4.2).

Unterstützt wird die Idee der Verwendung von Geschäftsobjekten und deren Status zudem durch die Arbeit von Mendling et al. (2009). Die Autoren haben dabei die Beschriftung von Aktivitäten im Bereich der Geschäftsprozessmodellierung untersucht und stellen fest, dass die Verwendung einer Kombination von Objekten und Verben den Probanden am nützlichsten und eindeutigsten erscheint. Falls man wie im vorliegenden Ansatz die bestehenden Informationen innerhalb eines Geschäftsprozessmodells für die Suche nach passenden Services weiterverwenden will, ist diese Erkenntnis sehr hilfreich. Objekte innerhalb der Aktivitätsbeschreibung können somit unter Umständen mit den Geschäftsobjekten und Verben mit den Status der Servicebeschreibungen verglichen werden. Aus der Arbeit von Mendling et al. (2009) ergibt sich allerdings auch, dass die Beschriftung der Aktivität vornehmlich das Resultat dieser beschreibt. So wird voraussichtlich ein Abgleich mit dem Effekt einer Servicebeschreibung sinnvoller sein, als ein Vergleich mit der Vorbedingung. Mögliche Vorbedingungen können sich allerdings aus vorangegangenen Aktivitäten ergeben. Weitere Überlegungen zu diesem Gebiet finden sich in Abschnitt 4.2.

Der konkrete Einsatz von Geschäftsobjekten und Status zur Beschreibung von Services im Speziellen für deren Vorbedingungen und Effekte, wird im nachfolgenden Abschnitt beschrieben.

4.1.4.2 Vorbedingungen und Effekte

Obwohl vor allem die funktionale Beschreibung des Service im Vordergrund des vorliegenden Ansatzes steht, soll für ein besseres Verständnis die gesamte Struktur einer möglichen Servicebeschreibung betrachtet werden. Eine Übersicht über die wichtigsten Bestandteile der im vorliegenden Ansatz verwendeten Beschreibung ist in Tabelle 6 zu sehen.

Es ist wichtig zu betonen, dass die vorgeschlagene semantische Beschreibung eine zusätzliche Informationsschicht über der syntaktischen Beschreibung (zum Beispiel mittels WSDL) einführt. Die Suche nach den Services kann allein auf der vorgestellten Beschreibung durchgeführt werden. Die Ausführung benötigt allerdings die syntaktische Beschreibung des Service, um die gewünschten Operationen aufrufen und die Ergebnisse maschinell weiterverarbeiten zu können.

Geschäftsprozessorientierte Servicebeschreibung

Servicename/-ID	Eindeutiger Name und ID des Service, Verwendung bestehender Namenskonventionen des Anbieters sinnvoll (Einbeziehung von Domänen und Geschäftsobjekten sinnvoll)	
Nichtfunktionale Eigenschaften	Nichtfunktionale Eigenschaften und Daten für die Verwaltung des Service im Rahmen des Servicemanagements, enthält auch Informationen zum Serviceanbieter	
Funktionale Domäne	Ordnet den Service einer Domäne der Organisation zu (optional aber empfehlenswert)	
Serviceendpunkt	Verweis auf die syntaktische Beschreibung des Service (zum Beispiel in Form von WSDL)	
Serviceoperationsnamen/-IDs	Eindeutige Namen und IDs der Operationen des Service, Namenskonventionen beinhalten unter Umständen bereits Geschäftsobjekte und Zielstatus in aktiver Form (zum Beispiel <i>Auftrag_abschliessen</i> , <i>Rechnung_versenden</i>)	
	Eingabeparameter	Verweis auf Datenobjekt(e), welche zur Ausführung der Operation notwendig sind
	Ausgabeparameter	Verweis auf Datenobjekt(e), welche nach der Ausführung der Operation ausgegeben werden
	Vorbedingung	Bedingungen, die vor Aufruf des Service erfüllt sein müssen. Geordnete Paare (2-Tupel) aus Geschäftsobjekt und Status, Verknüpfung mittels formaler Sprache und eindeutiger Konzepte (URI einer Ontologie)
	Effekt	Effekte, die nach Aufruf des Service eintreten. Geordnete Paare (2-Tupel) aus Geschäftsobjekt und Status, Verknüpfung mittels formaler Sprache und eindeutiger Konzepte (URI einer Ontologie)

Tabelle 6: Konzeptionelle Bestandteile der geschäftsprozessorientierten Servicebeschreibung.

Die in Tabelle 6 verwendete Struktur baut vornehmlich auf der Arbeit von Schröpfer (2009) (S. 275) sowie einer Untersuchung bestehender formaler Beschreibungsansätze für semantische Services auf (siehe dazu Polleres et al. (2006); Kopecký et al. (2007); Martin et al. (2007)). Für eine Detaillierung der nichtfunktionalen Eigenschaften sei abermals auf Schröpfer (2009) verwiesen.

Der Kern der funktionalen Beschreibung eines Service liegt wie bereits erwähnt in der Formulierung von Vorbedingung und Effekt. Für ein besseres Verständnis der Formalisierung von Vorbedingung und Effekt ist es hilfreich, einen Beispielservice und dessen Beschreibung zu verwenden (siehe Abbildung 58). Betrachtet werden soll nur eine der drei Operationen eines fiktiven Abrechnungsservice. Die Funktionalität der Operation 1 liegt in der Versendung einer Rechnung an einen Kunden (die eigentliche Umsetzung und somit physische oder elektronische Versendung ist hier nicht relevant). Eine einfache Beschreibung der Vorbedingungen ist unter Verwendung der Geschäftsobjekte *Auftrag* und *Rechnung* sowie der Statusinformationen *abgeschlossen* und *überprüft* möglich. Bevor eine Rechnung versendet werden kann, muss (zumindest in diesem Beispielunternehmen) zunächst der damit verbundene *Auftrag abgeschlossen* worden sein. Weiterhin muss eine *Überprüfung der Rechnung* (zum Beispiel mittels OP3 desselben Service) durchgeführt worden sein. Nach der Ausführung wurde wie gewünscht die *Kundenrechnung versandt* und eine *Zahlungsprüfung* zur Kontrolle der Rechnungsbegleichung wird *angesetzt*. Wie auch die Vorbedingung können die Effekte durch eine Kombination der entsprechenden Geschäftsobjekte und Statusinformationen dargestellt werden.

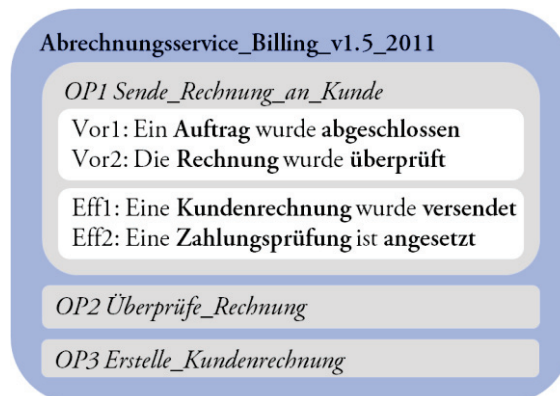


Abbildung 58: Beispiel für den Einsatz von Vorbedingung und Effekt in einem Abrechnungsservice.

Je nach Komplexität der Operation und Einsatzszenario ist es notwendig, die Beschreibung für Vorbedingungen und Effekte zu erweitern und deren Komplexität zu erhöhen. Dabei muss allerdings bedacht werden, dass eine Erhöhung der Komplexität der Beschreibung auch zu einer Erhöhung der Komplexität für eventuelle Suchanfragen führen kann. Hier gilt die Devise die Beschreibung so einfach wie möglich zu halten, aber auch nicht einfacher. Das Beispiel in Abbildung 58 verwendet für Vor- und Nachbedingung noch eine Formulierung in natürlicher Sprache. Die eigentliche Formalisie-

nung hängt von der gewählten Ontologiesprache zur Umsetzung der Servicebeschreibung ab. Zwei Grundvoraussetzungen für die Umsetzung sind die Verwendung eines *Geschäftsobjekt*- und eines *Status*-Konzeptes sowie die Verknüpfung dieser, so dass einem Geschäftsobjekt immer auch ein Status zugewiesen werden kann. Für die eindeutige Beschreibung der Funktionalität ist dies unabdingbar, für die spätere Suche sind Anfragen zu Geschäftsobjekt *oder* Status allerdings möglich. Eine Demonstration zur Umsetzung des Beispielservice in Form verschiedener Ontologiesprachen folgt in Abschnitt 4.1.5.

Auf eine (logische) Verbindung zwischen Vorbedingung und Effekt wird im vorliegenden Ansatz verzichtet. Dies liegt zum einen an der Annahme einer „Black Box“ der Aktivitäten und damit verbundenen Services. Zum anderen ist eine solche Beschreibung nur sinnvoll, wenn sie für die spätere Suche durch die Nutzer verwendet werden kann. Dies würde allerdings zu einer, nach Ansicht des Autors nicht notwendigen, Erhöhung der Komplexität der Suchanfragen führen. Falls sich in späteren Untersuchungen herausstellen sollte, dass eine solche Erweiterung notwendig ist, so erlaubt der vorliegende Ansatz durch seine Flexibilität auch die Beschreibung dieser Abhängigkeiten.

4.1.4.3 Einschränkungen

Die Ausdrucksstärke zur Beschreibung der Funktionalitäten mittels Geschäftsobjekten und Status unterliegt bestimmten Einschränkungen. Zum einen wird davon ausgegangen, dass ein Service beziehungsweise eine Operation durch ihren Aufruf ein oder mehrere Geschäftsobjekte transformiert, das heißt auch ihren Status (im Lebenszyklus) ändert. Ist dies nicht der Fall, müssen im vorliegenden Ansatz Hilfsstatus-Konzepte eingeführt werden.

Bei der Manipulation von Daten mittels sogenannter „CRUD“-Operationen (*Create*, *Read*, *Update*, *Delete*) auf Geschäftsobjekten wird im Falle von *Read* und *Update* zum Beispiel keine Statusveränderung erreicht. Obwohl keine Veränderung im Lebenszyklus des Geschäftsobjektes entsteht, ist es im vorliegenden Ansatz erlaubt explizit CRUD-Status zu verwenden. Dafür wird auch eine vordefinierte Klasse innerhalb der Ontologien definiert, welche diese Datenstatus gruppiert und gleichzeitig von anderen Status abgrenzt.

Weiterhin kann es Probleme geben, Operationen zu beschreiben welche, ein oder mehrere Geschäftsobjekte überprüfen. Dies ist eine Erweiterung einer *Read* Operation, wobei die gelesenen Daten anhand bestimmter Regeln kontrolliert werden können. Die Lösung dieses Problems ist entweder die Einführung eines *Checked*-Konzeptes, welches den Effekt unabhängig von dem Ergebnis der Überprüfung beschreibt. Dies führt allerdings dazu, dass der Effekt nicht unbedingt als Vorbedingung für weitere Aktivitäten oder Services verwendet werden kann, weil keine Aussage über den Erfolg der Operation getroffen wird. Oder es können andere Überprüfungsstatus, wie zum Beispiel *erfolgreich_überprüft*, als Hilfsstatus verwendet werden, welche das Ergebnis der Überprüfung bereits semantisch beinhalten. Der Einsatz letzterer bedingt jedoch Services, welche bei nicht erfolgreicher Überprüfung einen Fehler (*exception*) produzieren, welcher abgefan-

gen werden kann (*exception handling*). Der Einsatz der jeweiligen Hilfsstatus ist je nach Einsatzszenario zu entscheiden.

In den durchgeführten Forschungsprojekten sind neben den aufgeführten Einschränkungen keine weiteren Ausnahmen aufgetreten. Die Konzepte wurden in den Projekten mit Fach- und Domänenexperten diskutiert. Es ist nicht ausgeschlossen, dass in weiteren Einsätzen des Ansatzes zusätzliche Hilfskonzepte zur Beschreibung der Operationen der Services eingeführt werden müssen. Dies ist allerdings unkritisch, solange die semantische Eindeutigkeit (innerhalb der Domäne beziehungsweise Ontologie) und der Konsens der Modellierer gewahrt bleiben.

4.1.5 Nutzung bestehender formaler Beschreibung

In diesem Abschnitt sollen die Beispiele zur konzeptionellen geschäftsprozessorientierten Beschreibung von Services des vorangegangenen Abschnitts aufgegriffen und durch den Einsatz bestehender Beschreibungssprachen für semantische Web Services umgesetzt werden. Die Verwendung unterschiedlicher Sprachen soll die vielseitige Einsetzbarkeit des Ansatzes zeigen, die eigentliche prototypische Implementierung (siehe Abschnitt 5) basiert allerdings nur auf WSMO. Zusätzlich wird im folgenden Abschnitt die Möglichkeit der Repräsentation der Konzepte in OWL-S und SAWSDL untersucht.

4.1.5.1 WSMO

Das Web Service Modeling Ontology Framework bietet für den vorliegenden Ansatz drei wichtige Konzepte zur Beschreibung der Services und der Suchanfragen. Die eigentliche Servicebeschreibung verwendet das *Web Service* Konzept und Suchanfragen werden mittels des *Goal* Konzepts definiert. Beide Konzepte verwenden das *Ontology* Konzept, um verschiedene Klassen (Geschäftsobjekte, Status) referenzieren zu können. Zu jedem dieser Konzepte soll ein kurzes Beispiel auf Basis des Service in Abbildung 58 vorgestellt werden.

```
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-d1"
namespace {
    _"http://example.org/services/",
    example _"http://example.org/ontologien/example"}

webService Abrechnungsservice_Billing_v1.5_2011
importsOntology { _"http://example.org/ontologien/example"}

capability Abrechnungsservice_BillingCap
precondition Sende_Rechnung_an_KundeVor1
definedBy
    ?x memberOf example#Auftrag and forall ?y (?x[example#status hasValue
    ?y] implies ?y memberOf example#abgeschlossen).

precondition Sende_Rechnung_an_KundeVor2
definedBy
    ?x memberOf example#Auftrag and forall ?y (?x[example# status hasValue
    ?y] implies ?y memberOf example#ueberprueft).
```

```

effect Sende_Rechnung_an_KundeEff1
definedBy
    ?x memberOf example#Kundenrechnung and forall ?y (?x[example#status
    hasValue ?y] implies ?y memberOf example#versendet).

effect Sende_Rechnung_an_KundeEff2
definedBy
    ?x memberOf example#Zahlungspruefung and forall ?y (?x[example#status
    hasValue ?y] implies ?y memberOf example#angesetzt).

```

Abbildung 59: Beispielcode für einen Abrechnungsservice mit Vorbedingungen und Effekt als WSMO Web Service.

Die Abbildung 59 beschreibt eine mögliche Umsetzung der geschäftsprozessorientierten Servicebeschreibung des Abrechnungsservice. Zunächst wird definiert, welcher WSML-Dialekt verwendet werden soll. In diesem Fall wird WSML-DL als eine Form der Beschreibungslogik zur Formulierung der Aussagen innerhalb der Vorbedingungen und Effekte verwendet. Weiterhin werden der Namespace des Service und der zu importierenden Ontologie deklariert und diese Beispielontologie importiert. Nichtfunktionale Eigenschaften sowie eine Domänenzuordnung werden für das Beispiel ausgelassen. WSMO Web Services ermöglichen leider innerhalb der *Capability* (der Bereich für die Definition von Vor- und Nachbedingungen sowie Annahmen und Effekten) keine explizite Angabe von Operationen. Dieser Nachteil wird durch die Verwendung von eindeutigen Namenskonventionen (Operationsname + {Vor, Eff} + n) umgangen. Für die jeweiligen Vorbedingungen und Effekte werden im Anschluss Axiome definiert, welche für die spätere Suche verwendbare Konzepte auf Basis einer Verknüpfung von Geschäftsobjektkonzepten und Statuskonzepten beschreiben.

```

wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-dl"
namespace { _http://example.org/ontologien/}

ontology Beispielontologie

concept Geschaeftsobjekt
    status impliesType Status
concept Status

concept Auftrag subConceptOf Geschaeftsobjekt
concept Fertigungsauftrag subConceptOf Auftrag
concept Serviceauftrag subConceptOf Auftrag

concept Rechnung subConceptOf Geschaeftsobjekt
concept Kundenrechnung subConceptOf Rechnung

concept Pruefung subConceptOf Geschaeftsobjekt
concept Zahlungspruefung subConceptOf Geschaeftsobjekt

concept abgeschlossen subConceptOf Status
concept ueberprueft subConceptOf Status
concept versendet subConceptOf Status
concept angesetzt subConceptOf Status

```

Abbildung 60: Beispielcode für eine Ontologie zur Darstellung des Abrechnungsservice als WSMO Ontology.

Für ein besseres Verständnis der Servicebeschreibung ist in Abbildung 61 ein Beispielcode für die verwendete Ontologie beschrieben. In dieser werden die Konzepte zur Weiterverwendung in anderen Beschreibungen wie Service und Suchanfrage definiert. Wie auch in der Servicebeschreibung müssen Namespace und WSMML-Dialekt definiert werden. Ontologien können auch andere Ontologien durch die Importfunktion wiederverwenden, was in diesem Beispiel allerdings nicht nötig ist. Die Kernkonzepte der Ontologie sind das *Geschäftsobjekt* und der *Status*, wobei *Status* als Attribut des *Geschäftsobjekt*-konzeptes verwendet wird. Anschließend werden die notwendigen Subkonzepte definiert, wobei die Subsumption von Konzepten eine wichtige Rolle spielt. So ist die *Kundenrechnung* ein Unterkonzept einer *Rechnung*, was für die spätere Servicesuche ein wichtiger Faktor sein kann.

```
wsmlVariant _ "http://www.wsmo.org/wsmo/wsmo-syntax/wsmo-dl"
namespace {
  _http://example.org/ziele/
  example _http://example.org/ontologien/example}

goal Beispielanfrage

capability AnfrageCap
importsOntology { _ "http://example.org/ontologien/example" }

precondition AnfrageVor1
definedBy
  ?x memberOf example#Fertigungsauftrag and forall ?y (?x[example#status
  hasValue ?y] implies ?y memberOf example#abgeschlossen).

effect AnfrageEff1
definedBy
  ?x memberOf example#Rechnung and forall ?y (?x[example#status hasValue
  ?y] implies ?y memberOf example#gesendet).
```

Abbildung 61: Beispielcode für die Anfrage eines Service zur Rechnungsstellung nach einem Fertigungsauftrag als WSMO Goal.

Aufbauend auf der Ontologie aus Abbildung 61 können Suchanfragen nach Services mit einer bestimmten Funktionalität formuliert werden, wie in Abbildung 58 zu sehen. *Goals* in WSMO sind nahezu identisch mit dem *Web Service* Konzept. Sie müssen aber keine vollständige Beschreibung der Funktionalität des gewünschten Service beinhalten. So sind auch Anfragen nur auf Basis von Vorbedingungen oder Effekten möglich. Das vorliegende Beispiel sucht nach einem Service, welcher nach Abschluss eines *Fertigungsauftrags* eine *Rechnung* versenden kann. Der Abrechnungsservice aus Abbildung 59 würde in diesem Fall als Teiltreffer gewertet werden können. Zwar wird eine Vorbedingung mehr als erfüllt (da *Fertigungsauftrag* ein Unterkonzept von *Auftrag* ist). Es besteht aber zusätzlich eine weitere Vorbedingung. Zudem werden nur *Kundenrechnungen* durch den Service versendet. Dies kann, muss aber nicht ausreichend für die Anfrage nach *gesendeten Rechnungen* der Suche sein. Die unterschiedlichen Trefferkategorien werden in Abschnitt 4.2 und 4.3 besprochen.


```

    <profile:hasResult rdf:resource="_Sende_Rechnung_an_KundeEff2"/>
    <profile:hasInput rdf:resource="..." />
    <profile:hasOutput rdf:resource="..." />
</profile:Profile>

```

Abbildung 63: Beispielcode für einen Abrechnungsservice mit Verweis auf Vorbedingungen und Effekt (Result) als OWL-S Service Profile.

In Abbildung 64 ist eine Definition der Vorbedingungen und Effekte im Rahmen der *Process*-Definition von OWL-S zu sehen. OWL erlaubt den Einsatz unterschiedlicher Regelsprachen zur Formulierung von Bedingungen, vornehmlich SWRL und KIF. Im vorliegenden Beispiel kommt KIF zum Einsatz. Um den Umfang des Beispielcodes zu begrenzen, werden jeweils nur eine Vorbedingung und ein Effekt dargestellt. Zudem werden Teile des Codes verkürzt, um die Lesbarkeit zu erhöhen.

```

<process:hasResult>
  <process:Result>
    ...
    <process:inCondition rdf:ID="_Sende_Rechnung_an_KundePrel">
      <expr:KIF-Condition>
        <expr:expressionBody>
          (=> (instance ?x example#Auftrag)
              (hasStatus ?y
                (forall (?y)
                  (instance ?y example#abgeschlossen))))
        </expr:KIF-Condition>
      </process:inCondition>
      ...
    <process:hasEffect rdf:ID="_Sende_Rechnung_an_KundeEff1">
      <expr:KIF-Condition>
        <expr:expressionBody>
          (=> (instance ?x example#Kundenrechnung)
              (hasStatus ?y
                (forall (?y)
                  (instance ?y example#versendet))))
        </expr:KIF-Condition>
      </process:hasEffect>
    </process:hasResult>

```

Abbildung 64: Beispielcode für die Definition von Vorbedingung und Effekt als OWL-S Service Process unter Verwendung des Knowledge Interchange Format (KIF).

Die Formulierung von Vorbedingung und Effekt unterscheidet sich logisch nicht von den Beschreibungen in WSMO. Als Vorbedingung wird wieder angenommen, dass ein *Auftrag* zunächst *abgeschlossen* sein muss, bevor die Operation des Service durchgeführt werden kann. Der dargestellte Effekt ist die *Versendung* der *Rechnung*. Die ontologischen Konzepte für die Beschreibung des Service stammen aus einer OWL Beispielsontologie. Aufgrund ihres Umfangs ist diese zum besseren Verständnis in Abbildung 90 im Anhang A dargestellt. Die Formulierung einer Anfrage nach Services im Rahmen der Suche unterscheidet sich in OWL-S nicht von der Beschreibung des Service. Wie auch bei WSMO müssen diese Beschreibungen für die Suche nach einer bestimmten Funktionalität nicht vollständig sein.

4.1.5.3 SAWSDL

Die geschäftsprozessorientierte Servicebeschreibung mittels SAWSDL unterscheidet sich von den WSMO und OWL-S Varianten. Da SAWSDL weder eine eigene Ontologiesprache noch eine Möglichkeit zur Formulierung von logischen Ausdrücken mit sich bringt, müssen bestehende Ansätze in Form von Referenzen verwendet werden. Allerdings gibt es keine Vorgabe welche Konzepte durch SAWSDL referenziert werden müssen, der Ansatz ist ontologieagnostisch. Für das nachfolgende Beispiel werden daher die logischen Beschreibungen innerhalb des WSMO Web Service aus Abbildung 59 referenziert. Denkbar ist jedoch auch, dass die entsprechenden Ausdrücke mit OWL-S oder einer anderen Servicebeschreibungssprache formuliert werden. Wie bereits in Abschnitt 2.2.3.3 beschrieben und dort in Abbildung 11 zu sehen, können verschiedene Elemente einer WSDL-Datei mit Hilfe von SAWSDL-Referenzen annotiert werden. Für den vorliegenden Ansatz ist vor allem die Annotation der Operationen interessant. Die Operationen werden mit Referenzen auf die jeweiligen Vorbedingungen und Effekte versehen, so dass während einer Suche nach Services diese abgerufen und mit dem vom Nutzer formulierten Bedingungen verglichen werden können. In Abbildung 65 ist eine WSDL-Datei mit beispielhaften SAWSDL Referenzen dargestellt.

```
<wsdl:description
  targetNamespace="http://www.w3.org/2002/ws/sawSDL/spec/wsdl/order#"
  xmlns="http://www.w3.org/2002/ws/sawSDL/spec/wsdl/order#"
  xmlns:wsdl="http://www.w3.org/ns/wsdl"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:sawSDL="http://www.w3.org/ns/sawSDL">

  <wsdl:types>
    <!-- Definierung der Datentypen für In- und Outputs mit entsprechenden
          SAWSDL-Referenzen -->
  </wsdl:types>

  <wsdl:interface name="Abrechnungsservice" sawSDL:modelReference=
    "http://example.org/services/Abrechnungsservice_Billing_v1.5_2011">
    <wsdl:operation name="Sende_Rechnung_an_Kunde"
      pattern=http://www.w3.org/ns/wsdl/in-out
      sawSDL:modelReference= "http://example.org/services/
        Abrechnungsservice_Billing_v1.5_2011#Sende_Rechnung_an_KundeVor1"
      sawSDL:modelReference="http://example.org/services/
        Abrechnungsservice_Billing_v1.5_2011#Sende_Rechnung_an_KundeVor2"
      sawSDL:modelReference="http://example.org/services/
        Abrechnungsservice_Billing_v1.5_2011#Sende_Rechnung_an_KundeEff1"
      sawSDL:modelReference="http://example.org/services/
        Abrechnungsservice_Billing_v1.5_2011#Sende_Rechnung_an_KundeEff2">
      <wsdl:input element="Rechnung" />
      <wsdl:output element="Bestaetigung" />
    </wsdl:operation>
    <wsdl:operation name="Ueberpruefe Rechnung"
      pattern=http://www.w3.org/ns/wsdl/in-out>
      <!-- Operationsdefinition -->
    </wsdl:operation>
    <wsdl:operation name="Erstelle_Kundenrechnung"
      pattern=http://www.w3.org/ns/wsdl/in-out>
      <!-- Operationsdefinition -->
    </wsdl:operation>
    <wsdl:fault name="orderFault"/>
  </wsdl:interface>
```

```
</wsdl:description>
```

Abbildung 65: Beispielcode zur Referenzierung von Vorbedingungen und Effekten in einer WSDL2-Datei mit SAWSDL.

Obwohl es möglich ist, ein Element der WSDL-Datei (wie zum Beispiel die Operation) mit mehreren Referenzen zu versehen, könnten diese nicht explizit voneinander unterschieden werden. Das heißt, dass zur Interpretation der Servicebeschreibung Regeln für die Definition und das Parsen der Vorbedingungen und Effekte formuliert werden müssen. Dies ist zum Beispiel unter Verwendung von Prä- oder Suffixen möglich.

4.2 GPM-getriebene Service Discovery

In diesem Abschnitt wird der Kerngedanke für eine Service Discovery auf Basis der Geschäftsprozessmodellierung vorgestellt. Hierzu soll zunächst die Zielgruppe für die Nutzung einer solchen Service Discovery sowie die durch die Nutzung entstehenden Vorteile dargestellt werden. Anschließend wird diskutiert, welche bestehenden Informationen, insbesondere Kontextinformationen, zu Unterstützung der Suche verwendet werden können. In diesem Zusammenhang soll auch ein möglicher Einsatz des sogenannten Natural Language Processing (NLP) analysiert werden. Abschließend wird eine Integration der Service Discovery in eine Modellierungsumgebung besprochen. Dies dient als Basis für die spätere Umsetzung im Rahmen der prototypischen Implementierung.

4.2.1 Zielgruppe und Vorteile

Die Grundidee einer GPM-getriebenen Service Discovery basiert vor allem auf der Nachfrage einer Automatisierung von Geschäftsprozessen auf Basis serviceorientierter Architekturen. Ein Ziel einer Automatisierung auf Basis von Services ist die Erreichung einer Flexibilisierung der Prozesse unter Verwendung der Zielprinzipien der SOA durch Wiederverwendbarkeit, Komponierbarkeit, Integration und Interoperabilität (siehe auch Abschnitt 2.1.4.2). Grundlage für die effiziente Umsetzung dieser Prinzipien ist unter anderem die Service Discovery, da eine flexible Anpassung eines Geschäftsprozesses und vor allem die Wiederverwendung von Services nur gewährleistet werden kann, wenn auch die Services mit der gewünschten Funktionalität gefunden werden können. Aktuelle Ansätze der Service Discovery (zum Beispiel durch Implementierung der UDDI Spezifikation, siehe Abschnitt 2.1.5.2) bieten nur eine beschränkte Funktionalität zum Auffinden von Services an, meist basierend auf einer Schlüsselwortsuche und der Kategorisierung der Services. Die bereits diskutierte semantische Service Discovery verspricht eine enorme Verbesserung in der Qualität der Suchergebnisse unter dem Einsatz intelligenter Suchanfragen. Aktuelle Implementierungen bieten allerdings nur wenig Unterstützung bei der Formulierung semantischer Suchanfragen, welche durch ihre sehr formalen Sprachen (WSMO, OWL-S, etc.) und die verwendeten logischen Ausdrücke eine hohe Komplexität aufweisen können. Hinzu kommt, dass im Fall des Geschäftsprozessmanagement und der Geschäftsprozessmodellierung, welche als Ausgang-

szenario für den vorliegenden Ansatz dienen, die Kluft zwischen Fachseite und IT zur Umsetzung einer Automatisierung der Prozesse sehr groß ist (Hepp et al. (2005)). Aufbauend auf dem Gedanken des semantischen Geschäftsprozessmanagements (siehe Abschnitt 3.2), welches diese Kluft verringern soll, versucht der vorliegende Ansatz die Suche nach Services während der Geschäftsprozessmodellierung und für die nachfolgende Automatisierung zu unterstützen. Unter Berücksichtigung der durchgeführten Forschungsprojekte mit drei großen Telekommunikations- und Softwareunternehmen ergeben sich folgende mögliche Vorteile beziehungsweise Ziele für die Implementierung:

- (1) Verbesserte Suche nach fachlichen Services durch die Einbeziehung von Geschäftsprozessexperten, Geschäftsprozessmodellierern und Unternehmensarchitekten
- (2) Nutzung bestehender Informationen über die fachlichen Anforderungen an den Service durch bestehende Geschäftsprozessmodelle
- (3) Fachliche Vorauswahl an Services für die spätere Implementierung durch die IT-Abteilung
- (4) Vereinfachte Formulierung von semantischen Suchanfragen durch Wiederverwendung fachlicher Konzepte und ein kontrolliertes Vokabular

Wie bereits in Abschnitt 4.1.2 erwähnt, sind dies die identifizierten Zielgruppen für die Nutzung einer GPM-getriebenen Service Discovery (GPM-SD):

- (1) *Geschäftsprozessexperten*: Diese Zielgruppe besitzt eine hohe fachliche Qualifikation bezüglich der Geschäftsprozesse des Unternehmens. Der Einsatz von Geschäftsprozessexperten im Rahmen einer GPM-SD bezieht sich vor allem auf die Kenntnis von Geschäftsobjekten und Zusammenhängen zur Formulierung von Anforderungen an die Services, welche den Prozess unterstützen beziehungsweise automatisieren sollen.
- (2) *Geschäftsprozessmodellierer*: Das Wissen der Modellierer liegt vor allem im Umgang mit Sprachen und Regeln zur Darstellung der Prozesse. Basierend auf dem Wissen der Geschäftsprozessexperten oder entsprechender Dokumentation erstellen sie Geschäftsprozessmodelle mit allen relevanten Informationen für die spätere Durchführung des Prozesses. Dies ist allerdings nicht gleichzusetzen mit einer Automatisierung desselben. Im Rahmen der GPM-SD können GP-Modellierer ihre Kenntnis über das Modell zur Formulierung von Suchanfragen sowie initiale Vorauswahl und Einbindung von Services nutzen.
- (3) *Unternehmensarchitekten*: Die Gruppe der Unternehmensarchitekten bildet die Schnittstelle zwischen Fachseite und IT-Bereich. Das Wissen über den Aufbau der SOA im Unternehmen ermöglicht es den Architekten weitere technische Anforderungen an den Einsatz der gesuchten Services zu identifizieren, zum Beispiel die Notwendigkeit der Komposition zweier Operationen unter Verwendung von Datenmediatoren.

Für den vorliegenden Ansatz sollte erwähnt werden, dass die Nutzer durchaus mehrere dieser Rollen erfüllen können. Es ist vor allem wichtig, dass während der GPM-SD immer fachliches Wissen verwendet werden kann. Entweder implizit durch die Rolle

der Geschäftsprozessexperten und –Modellierer oder explizit durch Informationen innerhalb der Modelle.

Ein bereits mehrmals erwähnter Kernaspekt der GPM-SD ist die Verwendung von bestehenden Informationen im Rahmen der Modellierung von Prozessen. Welche Informationen verwendet werden können und aus welchen Quellen sowie in welchem Kontext diese entstehen, wird im folgenden Abschnitt beschrieben.

4.2.2 Nutzung von Kontextinformationen

Das Ziel einer GPM-SD ist das Auffinden von Services, welche durch ihre Funktionalität einen vorliegenden Geschäftsprozess unterstützen. Alternativ kann ein solcher Prozess auch in der Entstehung sein und kontinuierlich während der Modellierung mit Services verbunden werden. Dieser Kontext kann genutzt werden, um bestehende und entstehende Informationen für die Formulierung von Suchanfragen zu definieren. Im Folgenden soll untersucht werden, welche Informationen im Kontext der Geschäftsprozessmodellierung zur Verfügung stehen und wie diese unter Umständen für eine Suche nach fachlichen Services verwendet werden können.

Um zu ermitteln, welche Informationen in der Modellierung der Prozesse zur Verfügung stehen, muss zunächst eine Annahme über die Modellierungssprache getroffen werden. Für den vorliegenden Ansatz wird die BPMN in der Version 2.0 (OMG (2011)) als Grundlage unter folgenden Gesichtspunkten ausgewählt:

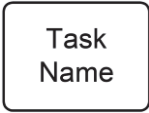
- Sie bietet eine standardisierte und in vielen Werkzeugen umgesetzte Beschreibung der meisten für die Geschäftsprozessmodellierung relevanten Objekte.
- Im Gegensatz zu anderen Modellierungssprachen, insbesondere der EPK, besitzt BPMN ein standardisiertes Metamodell.
- Die Zielgruppen der BPMN sind sowohl fachliche Modellierer als auch Implementierer für die Automatisierung der Prozesse.
- Bestehende Ansätze im Rahmen des semantischen Geschäftsprozessmanagements basieren vornehmlich auf BPMN und können somit möglicherweise mit dem hier vorgestellten Ansatz kombiniert werden (siehe Abschnitt 3.2.1).
- BPMN 2.0 besitzt die Möglichkeit einer direkten Ausführung der modellierten Prozesse, wodurch im Rahmen der GPM-SD gefundene Services direkt weiterverwendet werden können und somit die Kluft zwischen Fach- und IT-Seite weiter verringert wird.

Der Einsatz von BPMN⁶⁶ im vorliegenden Ansatz schließt den Einsatz anderer Modellierungssprachen (wie EPK) nicht aus. Unter Umständen können nur bestimmte Informationen nicht aus dem Kontext abgeleitet werden, von den grundlegenden Objek-

⁶⁶ Im weiteren Verlauf soll bei der Nennung von BPMN immer von der aktuellen Version 2.0 ausgegangen werden.

ten einer Geschäftsprozessmodellierungssprache (Aktivitäten, Datenobjekte, Verbindungselemente) muss aber ausgegangen werden können.

Die Tabelle 7 stellt eine Übersicht über die Konzepte von BPMN, die darin enthaltenen eventuell relevanten Informationen und die Beschreibung eines möglichen Einsatzes in der GPM-getriebenen Service Discovery dar. Es sei erwähnt, dass die Tabelle nur eine Auswahl und keine vollständige Auflistung der Modellierungskonzepte von BPMN bietet. In zukünftigen Erweiterungen oder anderen Szenarien können andere Konzepte der BPMN für die Suche relevant werden.

BPMN Objekt	Enthaltene Informationen	Mögliche Verwendung in der GPM-SD
Aufgabe (generisch) 	Aufgabenname, Kombination aus Verb und Objekt, Aktionsnomen oder Ähnlichem (siehe Mendling et al. (2009), S. 470)	Extraktion von Vorschlägen für Status und Objekt aus Verben, Objekten bzw. Aktionsnomen, Abgleich mit bestehenden Konzepten von Ontologien oder anderem kontrolliertem Vokabular
Aufgabe – Senden und Empfangen  	Meist wie Aufgabe (generisch), jedoch oft ohne oder mit <i>senden</i> bzw. <i>empfangen</i> als Verb, Objekt im Fokus	Extraktion von Objekt, <i>gesendet</i> bzw. <i>empfangen</i> als möglicher Status implizit abzuleiten, Filterung möglicher Services nach Typ
Aufgabe – Service 	Meist wie Aufgabe (generisch), Servicename, bereits annotierter Service	Extraktion von Vorschlägen für Status und Objekt, Weiterverwendung von Effekten für nachfolgende Aktivitäten, falls ein Service (mit Beschreibung) bereits annotiert ist
Aufgabe – Manuell und Benutzer 	Meist wie Aufgabe (generisch), Ausführende Person oder Rolle, Benutzeraufgabeteilautomatisiert (typische Workflowunterstützung), Manuelle Aufgabe nicht automatisiert (Konvention beachten)	Automatisierung unter Umständen nicht gewünscht bzw. irrelevant, Präsentationsservices (siehe Klassifikation) für Benutzerinteraktion denkbar (nicht behandelt im vorliegenden Ansatz), Filterung möglicher Services nach Typ, orchestrierte Services für die Workflowunterstützung relevant

	Aufgabe - Geschäftsregel	Meist wie Aufgabe (generisch) oder Name der Geschäftsregel	Geschäftsregelservices (siehe Klassifikation) primär relevant (nicht behandelt im vorliegenden Ansatz), Filterung möglicher Services nach Typ, ansonsten wie Aufgabe (generisch)
	Datenobjekt (generisch)	Name des Geschäfts- oder Datenobjekts, Zusätzliche Statusinformation (optional), Verknüpfung mit Aufgabe (gerichtet und ungerichtet), Multiple Objekte pro Aufgabe möglich, Multiple Aufgaben pro Objekt möglich	Extraktion von Vorschlägen für Status und Objekt aus Namen und Statusinformation (separate Definition in BPMN), gerichtete Verknüpfungen mit Objekten für Vorbedingungen und Effekte verwendbar (ungerichtete relevant nach Kontext, z.B. Verbindung zu zwei Aufgaben), je nach Modellierungskonvention auch für Suche nach Ein- und Ausgabeparametern einzusetzen, Interpretation als Geschäftsobjekt und nicht als Datenobjekt explizit
	Datenobjekte - Eingabe und Ausgabe	Wie Datenobjekt (generisch), größtenteils gerichtete Verknüpfungen (abhängig von Werkzeug und Modellierungskonventionen)	Extraktion von Vorschlägen für Status und Objekt aus Namen und Statusinformation, Abgrenzung zu generischen Datenobjekten möglich, welche u.U. die Geschäftsobjekte und nicht die Datenobjekte der Ein- und Ausgabe beschreiben, Ein- und Ausgabeinformation für Zuordnung zu Vorbedingung (Eingabe) und Effekt (Ausgabe) nutzbar
 	Nachricht – Initialisierend und nicht initialisierend	Wie Datenobjekt ohne Statusinformation, besitzt immer eine Richtung, kann nicht initialisierend sein (d.h. löst beim Empfänger keine Aktion aus)	Extraktion von Vorschlägen Objekt aus Namen, Richtung und Verknüpfung mit Aufgaben für die Verwendung als Vorbedingung und Effekt nutzbar,
 			

Tabelle 7: Auswahl an BPMN Modellierungskonzepten und Relevanz für die GPM-getriebenen Service Discovery.

Neben Konzepten der BPMN außerhalb von Prozessmodellen (Choreographien und Konversationen) wurden vor allem Ereignisse und Gateways (Entscheidungsknoten) nicht in der Übersicht betrachtet. Gateways besitzen generell eine Entscheidungslogik, welche im Rahmen der Geschäftsprozessengine implementiert ist und nicht durch einen zusätzlichen Service umgesetzt werden muss. Eventuell verwendbare Informationen der Entscheidungsknoten wären Bedingungen, welche an eingehenden und ausgehenden Sequenzen annotiert sind. Welche Informationen vorhanden sind, ist allerdings stark von den verwendeten Modellierungskonventionen abhängig und kann somit nicht immer als gegeben angenommen werden. In den verschiedenen Diagrammen und Ebenen der BPMN existieren zudem 48 mögliche Ereignisse. Diese Form der Variabilität macht es schwer von bestimmten Informationen für das generelle Ereigniskonzept auszugehen und eine Unterscheidung aller Ereignisse würde den Umfang der obigen Auflistung überschreiten. Ein Beispiel für die mögliche Nutzung bestimmter Eventtypen ist die Verwendung von Nachrichtenergebnissen und den damit verbundenen Nachrichteninformationen, welche in den ersten Aufgaben nach dem Ereignis relevant sein können.

Nachdem identifiziert wurde, welche Kontextinformationen potentiell relevant für die Formulierung von Suchanfragen für die GPM-SD sind, soll im Folgenden diskutiert werden, wie diese Informationen extrahiert werden können. Im Rahmen dessen wird im nachfolgenden Abschnitt vor allem der Forschungsbereich des Natural Language Processing (NLP) relevant, um Ähnlichkeiten zwischen bestehenden Informationen und Ontologiekonzepten zu erkennen.

4.2.2.1 Extraktion von Informationen mittels Natural Language Processing

Das Konzept der Nutzung von Kontextinformationen in Geschäftsprozessmodellen wurde bereits von anderen Autoren thematisiert. Mendling et al. (2009) schlagen in ihrer Arbeit vor, die Label von Aktivitäten beziehungsweise Aufgabe eines Prozesses auf ihre syntaktische Korrektheit bezüglich bestimmter Modellierungskonventionen (Verwendung einer Kombination aus Verb und Objekt für das Label) zu überprüfen. Hierzu können bestehende Tokenisierer die Label in ihre Bestandteile zerlegen und andere Werkzeuge (wie WordNet) diese als Verb, Nomen oder etwas anderes klassifizieren. Die Analyse der Labels zielt bei Mendling et al. (2009) allerdings vornehmlich auf die Kontrolle und Unterstützung der Modellierer sowie auf eine mögliche Verbalisierung des Prozessmodells. Die Weiterverwendung der gesammelten Informationen für die Suche nach Services zur Unterstützung des Prozesses wird nicht diskutiert. (Mendling et al. (2009), S. 12)

Die Arbeit von Born et al. (2007), welche bereits als Basis für die Formulierung von Vorbedingungen und Effekten durch Geschäftsobjekte und Statusinformationen verwendet wurde, analysiert Kontextinformationen für die Suche nach Prozessfragmenten. Verwendet werden zum einen die Namen der Prozesselemente und zum anderen den Kontext, in welchem sich ein spezifisches (ausgewähltes) Element befindet. So können Vorschläge für die domänenkonforme Benennung der Elemente generiert werden, indem die Nutzereingabe mittels string-basiertem Matching und der Identifizierung möglicher Synonyme mit den Konzepten der Domänenontologien verglichen werden. Der

Kontext der Elemente ist zudem relevant, um nachfolgende Aktivitäten auf Basis des Lebenszyklus eines verwendeten Konzeptes (zum Beispiel *Auftrag*) vorschlagen zu können. Zusätzlich beschreiben Born et al. (2007) bereits die im vorliegenden Ansatz angestrebte Weiterverwendung von ontologischen (Domänen-)Konzepten für die Formulierung von Zielen beziehungsweise Suchanfragen im Rahmen einer semantischen Service Discovery. (Born et al. (2007), S. 265-266)

Für die Identifizierung und Weiterverwendung von Kontextinformationen wird im vorliegenden Ansatz vorgeschlagen, bestehende Techniken und Werkzeuge des NLP einzusetzen (Jurafsky und Martin (2009)). Die Bezeichnungen der in Tabelle 7 identifizierten Elemente können zunächst, wie auch im Ansatz von Mendling et al. (2009), mittels eines Tokenisierers in ihre Bestandteile aufgeteilt werden. Anschließend erlauben sogenannte Part-of-Speech (POS) Tagger (zum Beispiel auf Basis von Kupiec (1992)) eine Kategorisierung, unter anderem nach Nomen, Verben und Adjektiven. Diese Kategorisierung hilft bei einem späteren Ähnlichkeitsvergleich mit bestehenden Ontologiekonzepten. So können zum Beispiel Nomen nur mit Geschäftsobjekt-konzepten und Verben nur mit Statusobjekt-konzepten verglichen werden, um den Aufwand der Ähnlichkeitsberechnung zu verringern und die Ergebnisqualität zu erhöhen. Unter Umständen versagen bestimmte POS Tagger, wenn nicht genug Informationen in Form eines entsprechenden Korpus existieren. Für einen solchen Fall ist der Einsatz linguistischer Thesauri, wie WordNet, denkbar. Nachdem diese Vorarbeit geleistet ist, können verschiedene syntaktische und semantische Techniken des Wortvergleichs eingesetzt werden (siehe auch Abschnitt 3.3.1.2).

Nachdem eventuelle Übereinstimmungen mit bestehenden Ontologiekonzepten erkannt wurden, gibt es unterschiedliche Möglichkeiten diese Informationen weiterzuverwenden. Eine mögliche Verwendung der Kontextinformationen hängt zusätzlich von dem Zeitpunkt beziehungsweise den Intervallen ab, in welchen ein Matching zwischen den Informationen des Geschäftsprozessmodells und den vorhandenen Ontologiekonzepten durchgeführt wird. So ist ein kontinuierliches Matching während der Modellierung in „Echtzeit“ denkbar, welches allerdings sehr rechenintensiv sein kann (abhängig von der Komplexität der zu vergleichenden Ontologien). Ein Matching in Echtzeit würde in seiner reinsten Form die Möglichkeit der Autovervollständigung von Begriffen während der Modellierung erlauben, wobei Wortfragmente auf eventuelle Ähnlichkeit mit den Ontologiekonzepten überprüft werden müssten. Eine solche Überprüfung ist natürlich auch erst nach Abschluss eines Wortes denkbar und damit weniger komplex. Eine Abschwächung des Echtzeitmatchings ist durch den Einsatz von Ereignissen möglich, welche unterschiedlich ausgelöst werden können. Zum einen kann ein Abgleich der Begriffe aktiv durch den Benutzer ausgelöst werden, indem zum Beispiel eine Vorschlagsfunktion angeboten wird. Zum anderen können Ereignisse wie der Wechsel von einem aktiven Modellierungselement (Aufgabe A) zu einem Anderen (Aufgabe B) ein Matching initiieren und dem Anwender Vorschläge für passende domänenkonforme Begriffe präsentieren.

Für den Ansatz in dieser Arbeit ist diese Informationsermittlung vor allem Mittel zum Zweck für eine vereinfachte Erstellung von Suchanfragen für die Unterstützung des

Prozesses mit passenden Services. In diesem Sinne werden im Folgenden Beispiele aus Tabelle 7 übernommen und die Verwendung für die Generierung von Suchanfragen diskutiert. Ein Vorschlag für die konkrete Integration in eine Modellierungsumgebung folgt im anschließenden Abschnitt.

Ein erstes einfaches Beispiel für die Nutzung von Kontextinformationen basiert auf einer *generischen Aufgabe* mit einer darin enthaltenen Beschreibung aus Objekt und Verb. Sollte eine Serviceanfrage formuliert werden, kann das extrahierte Objekt mit allen Geschäftsobjekt-konzepten und das extrahierte Verb mit allen Statuskonzepten der vorhandenen Ontologien verglichen werden. Eventuelle Treffer können zum einen genutzt werden, um dem Nutzer Bausteine für die Formulierung von Vorbedingung und Effekt zur Verfügung zu stellen. Zum anderen ist bei der Nutzung von Objekt und Verb innerhalb der Aufgabe oft ein bestimmtes Ergebnis beschrieben (zum Beispiel *Rechnung verschicken*), wodurch direkt ein spezifischer Effekt vorgeschlagen werden kann (in diesem Fall eventuell *Rechnung versendet*). In diesem Beispiel wird auch deutlich, wie wichtig bei der Analyse der Beschreibung der Umgang mit den Besonderheiten der natürlichen Sprache ist, da für die Erkennung des Status ein Synonym eingesetzt wurde. Weiterhin sollte immer auch ein Stemming (Ermittlung des Ursprungs) der Begriffe vor einem Vergleich durchgeführt werden, um eventuelle Konjugationen der Verben identifizieren zu können.

Ein zweites etwas komplexeres Beispiel nutzt die formalen Verbindungen zwischen den Elementen der BPMN. Angenommen es existieren zwei *generische Aufgaben*, welche durch einen Sequenzfluss direkt miteinander verbunden sind. Beide sind wie im ersten Beispiel durch eine Kombination von Objekt und Verb (*Auftrag abschließen*, *Rechnung verschicken*) beschrieben und die entsprechenden Begriffe werden extrahiert und mit bestehenden Ontologiekonzepten verglichen. Sollen beide Aufgaben durch Services unterstützt werden, existieren zwei Möglichkeiten. Zum einen kann zunächst ein Service für die erste Aufgabe analog zu Beispiel eins gesucht und vielleicht auch gefunden werden. Wird ein passender Service gefunden so können die Effekte dieses Service als mögliche Vorbedingungen für die zweite Aufgabe verwendet werden. Wird kein passender Service gefunden oder soll zunächst nach einem Service für die zweite Aufgabe gesucht werden, kommt die Verbindung der Informationen über den Sequenzfluss zum Tragen. Die aus der Beschreibung der ersten Aufgabe identifizierten Ontologiekonzepte (zum Beispiel *Auftrag* und *abgeschlossen*) dienen dann als Vorschlag für die Formulierung der Vorbedingung und die Konzepte aus der Beschreibung der zweiten Aufgabe (*Rechnung* und *versendet*) als Vorschlag für die Formulierung des Effektes. Je komplexer der Prozessfluss ist desto komplexer werden allerdings die möglichen Vorschläge für die Suchanfrage. Das heißt aber wiederum auch, dass mehr Informationen zur Verfügung stehen. Zusätzlich kann die Reichweite der semantischen Analyse begrenzt werden, indem zum Beispiel nur maximal eine oder zwei vorhergehende Aufgaben untersucht werden.

Das dritte und letzte Beispiel in diesem Abschnitt nutzt sowohl die Verbindungen zwischen einzelnen BPMN-Elementen sowie verschiedene Typen von Elementen und deren spezifische Semantik. Angenommen es existiert eine *Aufgabe* des Typs *Senden* sowie zwei damit verbundene *Datenobjekte*. Das erste Datenobjekt besitzt eine ausgehende

Verbindung zur Aufgabe, das zweite Datenobjekt eine eingehende Verbindung von der Aufgabe. Die Beschreibung der Aufgabe kann wie im ersten und zweiten Beispiel extrahiert und verwendet werden. Zudem kann die Semantik der Aufgabe, in diesem Fall der Typ *Senden*, für Vorschläge zur Formulierung von Effekten mit den Status *gesendet* oder einem anderen spezifizierten Konzept einer Ontologie führen. Weiterhin können die Benennung der Datenobjekte und deren (optionalen) Statusbeschreibungen als Vorschläge für Vorbedingung (Informationen des eingehenden Datenobjekts) und Effekt (Informationen des ausgehenden Datenobjekts) dienen. Der effektive Einsatz dieser Informationen hängt allerdings stark von den Modellierungskonventionen des jeweiligen Szenarios ab, kann aber bei eindeutigen Konventionen zu sehr guten Ergebnissen führen.

4.2.3 Integration in die Modellierung

Nachdem in den vorangegangenen Abschnitten Ziele, Zielgruppen und die nutzbaren Kontextinformationen besprochen wurden, sollen im Folgenden Anwendungsfälle im Rahmen der Integration einer GPM-getriebenen Service Discovery in eine Modellierungsumgebung dargestellt werden. Diese bauen teilweise auf den Beispielen des vorherigen Abschnittes auf, stellen aber vor allem den Ablauf während einer Suche nach Services bei der Modellierung von Geschäftsprozessen durchgängig dar. Somit werden auch bereits Anforderungen an die spätere Umsetzung durch eine prototypische Implementierung aufgestellt.

Die Idee des vorliegenden Ansatzes basiert auf dem Ziel die fachliche Modellierung der Geschäftsprozesse näher an die Entwicklung unterstützender Services und somit an die Automatisierung dieser heranzubringen. Um eine Service Discovery aufbauend auf der Modellierung anbieten zu können, wird eine Integration in bestehende Modellierungswerkzeuge vorgeschlagen. Es existiert eine Vielzahl an Lösungen für die Modellierung von Geschäftsprozessmodellen in unterschiedlichen Produkten. Beispiele in diesem Bereich sind umfangreiche GPM-Suiten wie die ARIS Plattform⁶⁷ der Software AG, Software-as-a-Service Lösungen wie der Signavio Process Editor⁶⁸ der Signavio GmbH oder Plug-Ins für bestehende Office-Lösungen wie der Process Modeler⁶⁹ für Microsoft Visio der itp commerce AG. Neben diesen kommerziellen Produkten existieren auch Open-Source-Lösungen wie der BPMN Editor des SOA Tools Platform Projekts⁷⁰ auf Basis von Eclipse. Obwohl der in dieser Arbeit vorgestellte Ansatz werkzeugunabhängig konzipiert ist, bestehen einige Grundvoraussetzungen für eine Integration. Zum einen muss das Werkzeug BPMN 2.0 unterstützen, um alle Konzepte des vorgestellten Ansatzes

⁶⁷ http://www.softwareag.com/de/products/aris_platform/

⁶⁸ <http://signavio.com/de/produkte/process-editor-as-a-service.html>

⁶⁹ <http://www.itp-commerce.com/process-modeler/>

⁷⁰ <http://www.eclipse.org/bpmn/>

auch umsetzen zu können. Zum anderen muss ein formales Modell des Prozesses existieren und Methoden für die Manipulation dessen und das Auslesen der Daten der BPMN-Elemente vorhanden sein.

Vorgestellt werden drei grundlegende Anwendungsfälle, welche aufeinander aufbauen. Diese werden im Folgenden anhand von grafischen Abbildungen erklärt:

- (1) Darstellung, Manipulation und Extraktion verwertbarer Informationen
- (2) Aufruf und Formulierung der Service-Anfrage
- (3) Auswahl und Verwendung von Services

4.2.3.1 Darstellung, Manipulation und Extraktion verwertbarer Informationen

Die Abbildung 66 stellt eine Auswahl verschiedener Elemente der BPMN dar, wie sie in einer Modellierungsumgebung auftreten können. Es gibt, wie mit den Verbindungen angedeutet, bereits bei einigen Elementen Referenzen zu einer Ontologie. Dies unterstellt die Möglichkeit, während der Modellierung auf Konzepte einer oder mehrerer Ontologien zugreifen zu können. Eine mögliche Umsetzung ist die Integration eines Ontologienavigators, wie er zum Beispiel im WSMO Studio existiert (Dimitrov et al. (2007a), S. 30-32) (in der Abbildung 66 nur abstrakt angedeutet). So können Referenzen zu ontologischen Konzepten direkt durch den Benutzer oder über Vorschläge durch das Werkzeug hergestellt werden. Für einen konkreten Vorschlag zur Umsetzung dieser Benutzerinteraktion sei auf Abschnitt 4.4 verwiesen. Neben der Darstellung und einer Manipulation der Begriffe und ontologischen Konzepte skizziert die Abbildung auch eine mögliche Extraktion der Information für die Weiterverwendung. Dies beinhaltet sowohl die Extraktion der Information eines Elements als auch die der verbundenen Elemente. In diesem Fall werden beispielhaft die Informationen aus einer Aufgabe, der vorangegangenen Aufgabe und den verbundenen Datenobjekten dargestellt. Ist die Möglichkeit der Extraktion gegeben, können die Informationen weiterverwendet werden, zum Beispiel für den nachfolgenden Anwendungsfall.

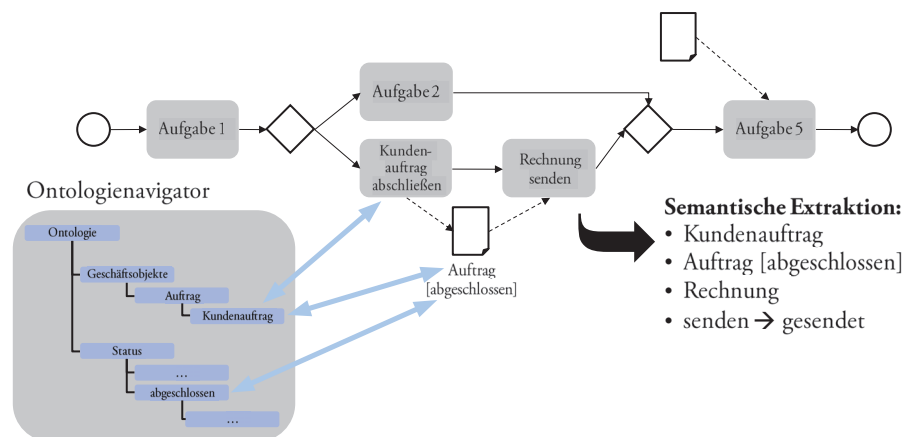


Abbildung 66: Verwertbare Informationen in der Geschäftsprozessmodellierung

4.2.3.2 Aufruf und Formulierung der Service-Anfrage

Die Abbildung 67 kann auf dem vorangegangenen Anwendungsfall aufbauen, erlaubt aber auch ein Szenario ohne vorherige Extraktion von Informationen. Der Anwender kann, zum Beispiel durch Auswahl und eine entsprechende Menüfunktionalität oder Rechts-Klick, einen Dialog zur Erstellung einer Service-Anfrage starten. Der abgebildete Dialog ist nur eine Möglichkeit von vielen, dem Benutzer eine anwenderfreundliche Umgebung für die Formulierung semantischer Vorbedingungen und Effekte für die Servicesuche zu präsentieren. Wichtig ist zum einen, dass der Nutzer keine syntaktischen Fehler machen soll und nicht mit einer reinen Form der logischen Ausdrucksprache konfrontiert wird. Zum anderen muss es möglich sein, bestehende Ontologiekonzepte suchen und verwenden zu können, sowie Vorschläge für passende Geschäftsobjekte und Status (womöglich auch einen Vorschlag zu Kombination) zu erhalten. Die Integration in die Modellierung hilft dabei auf unterschiedliche Weise: (1) Bestehende Informationen des Modells können als Basis für die Suchanfrage dienen (siehe Anwendungsfall 1), (2) Der Anwender muss nicht die Perspektive oder die Anwendung wechseln, um passende Services für die Unterstützung des Prozesses zu suchen, und (3) Konzepte der Domäne können implizit (durch das Verständnis des Nutzers im Kontext des Prozesses) oder explizit (durch die Präsentation der Ontologien in der Modellierungsumgebung) wieder- beziehungsweise weiterverwendet werden. Eventuelle weitere Optionen zur Servicesuche (Filterung, Schlüsselwörter, Kategorien) sind denkbar, aber für den Kern des Anwendungsbeispiels zunächst nicht relevant. Sobald eine Anfrage oder Teile einer Anfrage formuliert wurden, kann die Servicesuche gestartet werden und der letzte vorzustellende Anwendungsfall für die Integration in die Modellierung folgt.

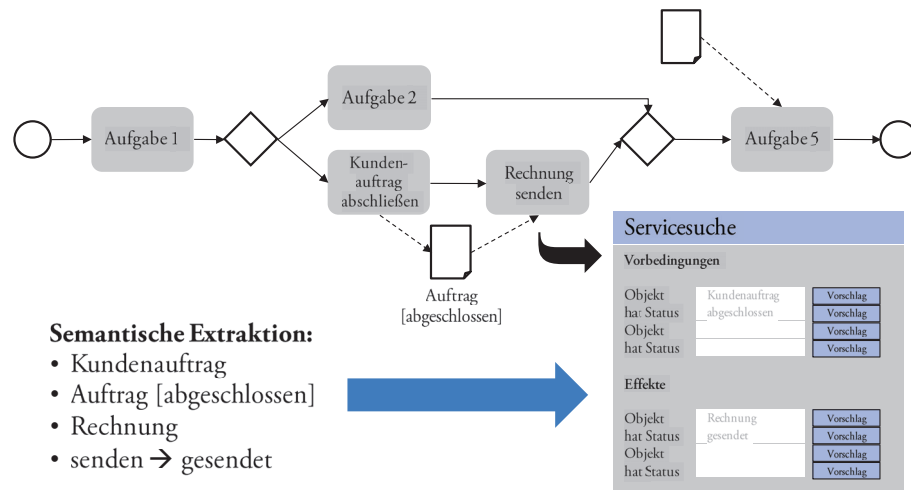


Abbildung 67: Formulierung einer Serviceanfrage mit extrahierten Informationen

4.2.3.3 Auswahl und Verwendung von Services

Die Abbildung 68 beschreibt einen Anwendungsfall nach der erfolgreichen Formulierung und Versendung der Serviceanfrage. Wie auch im vorangegangenen Anwendungsfall wird die Modellierungsumgebung nicht verlassen. Dies erlaubt die Servicekandidaten im Kontext ihres späteren Einsatzes zu sehen. Es kann sowohl überprüft werden, ob ein oder mehrere Services die ausgewählte Aufgabe unterstützen, als auch ob Services für nachfolgende oder vorangegangene Aufgaben in der Ergebnisliste existieren. Dieser Fall kann eintreten, wenn zum Beispiel bestimmte Vorbedingung auch für weitere Aufgaben im gleichen Prozess gelten, allerdings andere Effekte nach sich ziehen. Obwohl eine Reihenfolge der Ergebnisse existiert, welche die potentiell besten Services für die Anfrage priorisiert, liegt es letztlich am Anwender die Auswahl zu treffen. Je nach Szenario können auch mehrere Kandidaten identifiziert werden, um eine endgültige Auswahl in der späteren Implementierung treffen zu können. Vorteile der Integration in die Modellierungsumgebung liegen in der bereits erwähnten Kontextualisierung der Ergebnisse sowie in der Möglichkeit, die Services direkt bestimmten Aufgaben zuweisen zu können. Für die Zuordnung der Services zu den Aufgaben können die Metadaten der BPMN verwendet werden, in denen Referenzen zu Services und Operationen hinterlegt werden können (OMG (2011), S. 158-159).

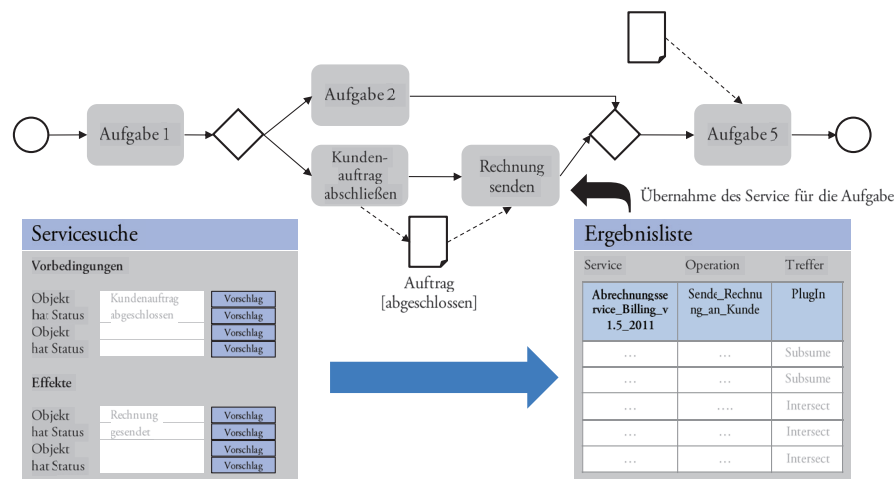


Abbildung 68: Integration der Servicesuche in die Prozessmodellierung

Nachdem Zielgruppen und Vorteile der GPM-getriebenen Service Discovery vorgestellt sowie Beispiele und Anwendungsfälle präsentiert wurden, soll im nächsten Abschnitt die zweite Säule des entwickelten Ansatzes beschrieben werden: die heterogene Service Discovery. In Teilen beeinflusst die Möglichkeit, mehrere Ontologien zur Servicesuche nutzen zu können, auch die Konzeption der GPM-SD sowie die Integration in eine Modellierungsumgebung. Entsprechende Hinweise werden dazu gegeben.

4.3 Heterogene Service Discovery

Neben der geplanten Geschäftsprozessorientierung des Ansatzes steht vor allem die Möglichkeit im Vordergrund, mehr als eine Ontologie zur Beschreibung der Services sowie zur Formulierung der Suchanfragen verwenden zu können. Das Konzept einer solchen heterogenen Service Discovery und eine Möglichkeit zur Umsetzung soll im folgenden Abschnitt beschrieben werden. Hierfür soll zunächst die zugrundegelegte „normale“ Service Discovery beziehungsweise der eingesetzte Algorithmus beschrieben werden. Anschließend wird diskutiert, wie zwischen den Konzepten der Ontologien eine Interoperabilität hergestellt werden kann, um ähnliche Konzepte für die Suche nach Services berücksichtigen zu können. Weiterhin soll betrachtet werden, wie bestehende Ansätze für das Matching und Alignment von Konzepten in den Ansatz der heterogenen Service Discovery integriert werden können. In diesem Rahmen werden auch Vor- und Nachteile der Wiederverwendung bestehender Werkzeuge diskutiert. Abschließend werden Techniken zur kontinuierlichen Verbesserung der Service Discovery vorgestellt. Zudem wird der Zeitpunkt für die Herstellung der Interoperabilität zwischen den Konzepten als Einflussgröße für die Performanz des Ansatzes untersucht.

4.3.1 Grundlegende Service Discovery

Bevor unterschiedliche Ontologien für die Beschreibung von Services und Suchanfragen verwendet werden können, muss die Basis für eine Discovery auf einer homogenen oder auch einzelnen Ontologie beschrieben werden (eine „homogene“ Discovery kann auch auf mehreren Ontologien mittels Mediatoren durchgeführt werden). Der eingesetzte Algorithmus ähnelt bestehenden Ansätzen für das Matching von Services und wird im Folgenden sowohl textuell als auch mit Hilfe eines Ablaufdiagramms beschrieben. Vorab sollen allerdings einige Annahmen über die Servicebeschreibungen und die Informationen getroffen werden, welche als relevant für die Suche beziehungsweise das Matching identifiziert wurden.

Für den vorliegenden Ansatz wird nur eine funktionale Service Discovery beschrieben. Zum einen, da eine nichtfunktionale Service Discovery erst sinnvoll durchgeführt werden kann, wenn mehr als ein Service mit exakt der gleichen oder zumindest einer sehr ähnlichen Funktionalität zur Verfügung steht. Zum anderen da eine Vielzahl bestehender Ansätze für die Suche nach Services auf Basis nichtfunktionaler Eigenschaften existieren (siehe unter anderem Schröpfer et al. (2009) und Wang et al. (2006)). Unter Umständen können die grundlegenden Konzepte des vorliegenden Ansatzes in der Zukunft auch für die nichtfunktionale Servicesuche eingesetzt werden. Dies liegt allerdings außerhalb des Rahmens dieser Arbeit und ist maximal für den Ausblick zum Ende der Ausarbeitung relevant.

Im Bereich der funktionalen Discovery wird eine Suche auf Basis der geschäftsprozessorientierten Servicebeschreibungen durchgeführt (siehe Abschnitt 4.1). Wie bereits erläutert, wird zunächst nur eine Beschreibung in Form von Vorbedingungen und Effekten verwendet. Untersuchungen der Eingangs- und Ausgangsparameter können durch

bestehende Ansätze realisiert werden. Hierbei ist durchaus eine zweistufige Suche denkbar, in welcher zunächst Services mit passender Funktionalität gefunden und anschließend auf passende Ein- und Ausgangsparameter untersucht werden. Eine weitere Grundannahme für den Algorithmus der funktionalen Discovery geht davon aus, dass einzelne Services mehrere Operationen beinhalten können. Jede dieser Operationen kann 0-n Vorbedingungen und 0-m Effekte besitzen, welche die Funktionalität dieser einzelnen Operation beschreiben. Bei der Formulierung einer Anfrage wird genau eine Operation definiert, welche ebenso 0-n Vorbedingungen und 0-m Effekte besitzen kann. In der gewählten Beschreibungssprache (WSML) können pro Service nicht explizit mehrere Operationen definiert werden. Daher wird mit eindeutigen Operationsnamen verbunden mit Suffixen für Vorbedingungen und Effekte gearbeitet (Operationsname+[Pre][Eff]). Die Beschreibungen der Vorbedingungen und Effekte sind im vorgestellten Ansatz auf eine Konjunktion von Geschäftsobjekten mit entsprechenden Status beschränkt. Für die erfolgreiche Ausführung des Algorithmus sind nur die formale Syntax der Bedingungen und nicht die vorgeschriebenen Konventionen relevant. Bei Nichteinhaltung der Konventionen kann die Qualität der Ergebnisse allerdings nicht garantiert werden.

Der Algorithmus (siehe Abbildung 69) arbeitet auf einer Sammlung bestehender Services, das heißt es werden während der Laufzeit keine zusätzlichen Services geladen. Mit den Services werden auch die mit diesen verbundenen Ontologien (in der Beschreibung als *import* definiert) in die Wissensbasis des verwendeten Reasoners geladen. Zu Beginn der Suche werden die Vorbedingungen und Effekte aus der Anfrage extrahiert und als Hilfskonzepte in der Wissensbasis abgelegt. Die jeweiligen Hilfskonzepte der Anfrage bestehen dabei aus einer Konjunktion der Vorbedingungen beziehungsweise Effekte (*Geschäftsobjekt1 mit Status1 AND Geschäftsobjekt2 mit Status2* etc.). Die beiden entstehenden Ontologiekonzepte können somit im Laufe der Suche mit den Beschreibungen der Operationen abgeglichen werden. Anschließend wird eine Iteration über die einzelnen Services der Sammlung durchgeführt. Hierfür werden zunächst für die einzelnen Services auf Basis der Operationsnamen die Vorbedingungen und Effekte pro Operation extrahiert. Wie auch für die Anfrage werden je Operation Hilfskonzepte aus Konjunktionen der Vorbedingungen und Effekte erstellt. Diese können nun mit den Hilfskonzepten der Anfrage verglichen werden. Für den Vergleich wird die Subsumptionslogik des Reasoners auf Basis der Beschreibungslogik der Bedingungen verwendet. Es werden folgende Trefferkategorien unterschieden. Diese Trefferkategorien gelten pro Bedingung und nicht pro Operation:

- | | |
|----------------|---|
| <i>Exact:</i> | Die jeweilige Vorbedingung beziehungsweise der jeweilige Effekt der Anfrage stimmt exakt mit denen der Operation überein. |
| <i>PlugIn:</i> | Die jeweilige Bedingung der Operation $\mathcal{B}_O$ subsumiert die Bedingung der Anfrage $\mathcal{B}_A$ ($\mathcal{B}_O > \mathcal{B}_A$). Für die Vorbedingung bedeutet dies, dass diese auf jeden Fall erfüllt ist ($\mathcal{B}_A$ ist ein Spezialfall von $\mathcal{B}_O$). Im Falle des Effekts bietet der Service eine breitere Funktionalität als gewünscht an. |

- Subsume:* Die jeweilige Bedingung der Anfrage $\mathcal{B}_A$ subsumiert die Bedingung der Operation $\mathcal{B}_O$. Somit existiert eine Umkehrung des PlugIn-Treffers, $\mathcal{B}_O$ wird von $\mathcal{B}_A$ subsumiert. Gleichzeitig entsteht eine Umkehrung der Erfüllung von Vorbedingung und Effekt wie im PlugIn-Treffer beschrieben. Somit ist zum einen möglicherweise nur ein Teil der Vorbedingung erfüllt. Zum anderen kann der Service unter Umständen nur ein Teil beziehungsweise ein Spezialfall der gewünschten Funktionalität erfüllen.
- Intersect:* Ein *Intersect*-Treffer bedeutet nicht unbedingt eine Überschneidung der Konzepte der Bedingungen, sondern wird durch die Bewahrheitung der *entails* Überprüfung des Reasoners erzeugt. Kann die jeweilige Bedingung nicht erfüllt werden (zum Beispiel durch eventuell disjunkte Konzepte) entsteht kein *intersect*.
- Disjoint:* Werden bei der Überprüfung der Bedingungen disjunkte Konzepte gefunden, entsteht ein *Disjoint*-Treffer.

Die unterschiedlichen Trefferkombinationen der Vorbedingungen und Effekte führen zu den jeweiligen Treffern der Operation. Die Tabelle 8 beschreibt die entstehenden Gesamttreffer.

Vorbedingung	Effekt	Gesamt
Exact	Exact	Exact
Exact	PlugIn	PlugIn
Exact	Subsumes	Subsumes
Exact	Intersect	Intersect
Exact	Disjoint	Disjoint ⁷¹
PlugIn	Exact	PlugIn
PlugIn	PlugIn	PlugIn
PlugIn	Subsumes	Subsumes
PlugIn	Intersect	Intersect
PlugIn	Disjoint	Disjoint
Subsumes	Exact	Subsumes
Subsumes	PlugIn	Subsumes
Subsumes	Subsumes	Subsumes
Subsumes	Intersect	Intersect
Subsumes	Disjoint	Disjoint
Intersect	Exact	Intersect

⁷¹ Da der Service nicht die benötigte Funktionalität bieten kann.

Intersect	PlugIn	Intersect
Intersect	Subsumes	Intersect
Intersect	Intersect	Intersect/ Fail ⁷²
Intersect	Disjoint	Disjoint
Disjoint	[All]	Disjoint

Tabelle 8: Mögliche Trefferkombinationen für Vorbedingungen und Effekte der Operationen im Rahmen der Servicesuche

Nachdem die Treffer durch das Subsumption-Reasoning ermittelt und zusätzlich zu einem Gesamttreffer pro Operation aggregiert wurden, können sie zur Ergebnisliste hinzugefügt werden. Darin wird neben den Treffern die eindeutige Identifikation des Service und der jeweiligen Operation abgelegt. Bei der Darstellung der Ergebnisse für den Nutzer kann diese Ergebnisliste nach Relevanz geordnet und bestimmte Trefferkategorien ausgeschlossen werden (zum Beispiel keine Darstellung der *Fail* und *Disjoint* Gesamttreffer, Ordnung der Treffer nach *Exact* > *PlugIn* > *Subsumes* > *Intersect*).

In dem vorgestellten Algorithmus werden jeweils die Vorbedingungen und Effekte einer Operation mit denen der Anfrage verglichen. Für die Überprüfung auf entsprechende Treffer findet eine Konjunktion statt. Dies geschieht unter der Annahme, dass möglichst eindeutige Treffer gefunden werden sollen. Eine Verknüpfung über ein logisches ODER (Disjunktion) würde voraussichtlich zu einer sehr hohen Anzahl an falsch positiven Treffern führen (Bedingungen welche nur in einer Aussage übereinstimmen).

⁷² Zusätzliche Kategorie für die Gesamttreffer, da ansonsten zu viele eventuell mögliche Operationen als Treffer gewertet werden.

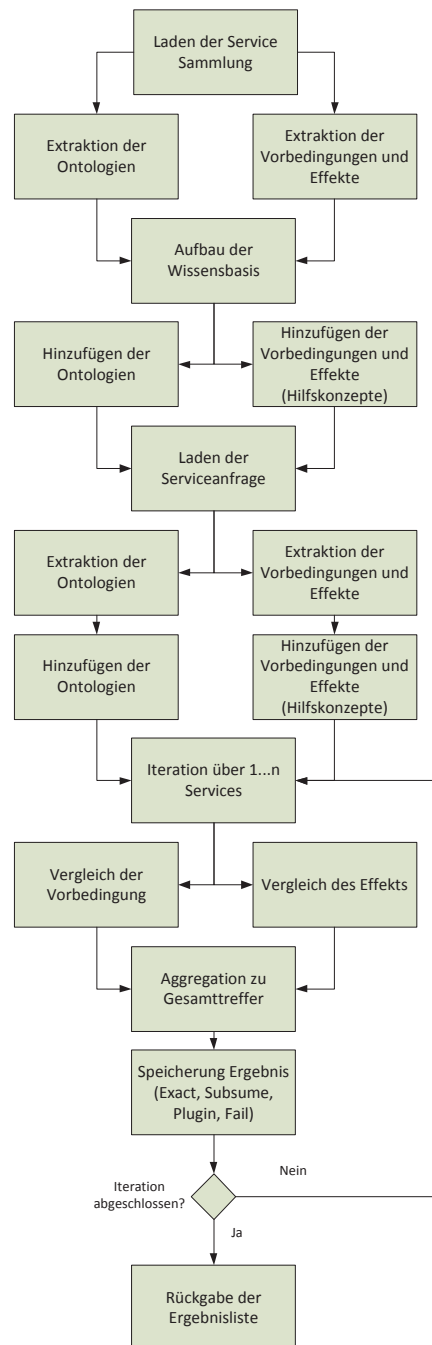


Abbildung 69: Einfacher Algorithmus der Service Discovery

4.3.2 Herstellen von Interoperabilität

Die Problematik beim Einsatz einer herkömmlichen Service Discovery, wie im vorigen Abschnitt beschrieben, liegt in der fehlenden Interoperabilität bei der Verwendung mehrerer Ontologien. Solange Anfragen und Services auf der gleichen Ontologie basieren oder bei der Verwendung von mehreren Ontologien bestimmte Regeln eingehalten werden (z.B. Vorbedingungen werden immer mit O_1 und Effekte mit O_2 beschrieben) kann der bereits vorgestellte Algorithmus verwendet werden. Andernfalls muss über bestimmte Techniken die benötigte Interoperabilität hergestellt werden, das heißt gleiche oder ähnliche Konzepte unterschiedlicher Ontologien müssen erkannt (Matching) und einer Beziehung zugeordnet werden (Alignment).

Es existieren verschiedene Möglichkeiten ein solches Matching durchzuführen und im bestehenden Service Discovery Algorithmus zu verwenden. Diese bieten unterschiedliche Vor- und Nachteile und werden daher im Folgenden kurz diskutiert:

- *Ausschließliches Matching der verwendeten Konzepte:* Diese Variante betrachtet ausschließlich diejenigen Konzepte, welche explizit in der Formulierung von Anfrage und Servicebeschreibungen auftreten. So können zum Beispiel alle Konzepte in der Vorbedingung einer Anfrage mit allen Konzepten in der Vorbedingung eines Service während der Discovery verglichen und die identifizierten Treffer zur Berechnung der jeweiligen Trefferkategorie verwendet werden.
 - *Vorteil:* Diese Variante bedeutet unter Umständen weniger Aufwand bei der Berechnung der Treffer, vor allem bei der Verwendung sehr großer Ontologien, da nur ein geringer Teil der Ontologie betrachtet werden muss. Dies bedingt aber auch gleichzeitig die Nachteile.
 - *Nachteil:* Gerade weil nur wenige Teile der Ontologie (nur die verwendeten Konzepte) mit einbezogen werden, leidet die Qualität des Matchings. So können meist nur *terminologische Techniken* (siehe Abschnitt 3.3.1.2) beziehungsweise unter Umständen *strukturelle Techniken* auf Basis der Eigenschaften der Konzepte eingesetzt werden. Dies kann zu einer Vielzahl an False Positives und False Negatives führen.
- *Matching von Ausschnitten der Ontologien:* Diese Variante ist eine Erweiterung des ausschließlichen Matchings von Konzepten. Hierbei wird neben den verwendeten Konzepten auch deren Umfeld innerhalb der jeweiligen Ontologie mit in das Matching einbezogen. Hierbei existieren unterschiedliche Varianten zur Reichweite der mit einbezogenen Informationen. Zumindest aber werden Super- und Subkonzepte der Ursprungskonzepte betrachtet sowie unter Umständen deren Attribute.
 - *Vorteil:* Der Hauptvorteil dieser Variante liegt offensichtlich in der Verwendung weiterer Informationen, um die Qualität des Matchingergebnisses zu erhöhen. So können neben *terminologischen*

- gezielt *strukturelle Techniken* eingesetzt werden. Je nach Beschreibungslogik können auch Instanzen im Rahmen *extensioneller Techniken* verwendet werden (allerdings beschränkt durch eine voraussichtlich geringe Anzahl an zu betrachtenden Instanzen).
- *Nachteil:* Obwohl mehr Informationen als in der ersten Variante verwendet werden, kann unter Umständen die Qualität der Ergebnisse nicht ausreichend sein. Zudem entsteht ein höherer Aufwand, vor allem in der Extraktion der Ontologiefragmente und bei der Iteration des Matchingprozesses für die einzelnen Servicebeschreibungen.
 - *Matching der Gesamtontologien:* Das andere Extrem zu dem Matching allein auf Basis der verwendeten Konzepte ist die Einbeziehung aller verwendeten Ontologien. Hierbei kann man differenzieren zwischen dem Matching der Ontologien der Anfrage und aller Servicebeschreibungen vor der Iteration während der Discovery und dem Matching der Ontologien der Anfrage mit denen einer einzelnen Servicebeschreibung während der Iteration. Das Matching der gesamten Ontologien kann mit oder ohne Einbeziehung der Instanzen durchgeführt werden.
 - *Vorteil:* Durch die Einbeziehung aller vorhandenen Informationen kann die Qualität der Ergebnisse maximiert werden. Bestimmte Techniken des Matchings, wie zum Beispiel *extensionelle*, können nur sinnvoll ab einer bestimmten Grundmenge an Informationen eingesetzt werden. Weiterhin können solche Matchings bereits vor einer Discovery durchgeführt werden, da nicht relevant ist welche Konzepte im späteren Verlauf verwendet werden.
 - *Nachteil:* Mit einer steigenden Anzahl an Informationen wächst generell die Komplexität und der Aufwand des Matchingprozesses.
 - *Verwendung vorhandener Alignments:* Neben dem Matching während des Discovery Prozesses besteht unter Umständen die Möglichkeit, vorhandene Alignments der eingesetzten Ontologien zu verwenden. Diese können zum Beispiel aus vorherigen Discovery Prozessen oder zentralen Alignment Repositories stammen. Wie auch die Alignments beim Matching der Gesamtontologien vor der eigentlichen Discovery werden vorhandene Alignments zu Beginn des Discovery Prozesses eingebunden und stellen somit eine gewisse Interoperabilität der verwendeten Konzepte her.
 - *Vorteil:* Beim Einsatz bereits vorhandener Alignments entsteht kein Aufwand zur Berechnung der Ähnlichkeit zwischen den jeweiligen Ontologien. Dies kommt allerdings nur zum Tragen, wenn Alignments zwischen allen verwendeten Ontologien existieren.
 - *Nachteil:* Die größten Nachteile liegen in der Ungewissheit über die Qualität der Alignments und deren Verfügbarkeit. Ersteres kann verbessert werden, indem vornehmlich Ergebnisse aus dem „eigenen“

Discovery Prozess wiederverwendet werden. Unter Umständen kann somit auch die Qualität der Ergebnisse durch weitere Iterationen des Matching verbessert werden.

Neben den verschiedenen Varianten zur Herstellung von Interoperabilität muss vor allem die steigende Komplexität bei dem Auftreten einer Vielzahl an unterschiedlichen Ontologien bei Anfragen und Beschreibungen betrachtet werden. Je nach Art und Zeitpunkt des Matchings und der Beschreibung der Anfragen und Services kann mit mehr als zwei Ontologien unterschiedlich umgegangen werden. Werden zu einem Zeitpunkt nur jeweils zwei Konzepte miteinander verglichen, entsteht auch bei der Verwendung von mehreren Ontologien kein Problem. Auch bei Anfragen und Servicebeschreibungen bei denen jeweils die Vor- oder Nachbedingung nur genau eine (nicht-interoperable) Ontologie verwendet, können die verwendeten Ontologien direkt verglichen werden. Die Komplexität steigt offensichtlich, wenn zum Beispiel jeweils Geschäftsobjekt und Status von Anfrage und Beschreibung von unterschiedlichen (in diesem Fall mindestens vier) Ontologien beschrieben werden. Das kann durch die Verwendung von Modellierungsrichtlinien umgegangen werden, indem solche Konstrukte generell nicht erlaubt werden. Dies ist zwar sehr restriktiv, verringert allerdings deutlich die Komplexität und kann gerade in kontrollierten Umgebungen gut umgesetzt werden. In einer „offenen Welt“ kann eine solche Form der Kontrolle unter Umständen nicht realisiert werden. Daher wird vorgeschlagen für diese Fälle entweder alle auftretenden Ontologien untereinander zu Beginn der Discovery zu matchen und eine hohe Komplexität zu riskieren oder Zwischen- beziehungsweise Brückenontologien einzusetzen. Diese Zwischenontologien bieten den zentralen Ankerpunkt für Konzepte gegen welchen die jeweiligen Ontologien verglichen werden. So lässt sich die Komplexität von N Ontologien auf bis zu N Alignments reduzieren. Zwischenontologien können ex-ante bereitgestellt werden oder während des Matchings entstehen, indem die verwendeten Ontologien nacheinander miteinander verbunden werden (sogenanntes *Ontology Merging*). Dieser Exkurs zeigt die zunehmende Komplexität bei dem Einsatz einer Vielzahl von unterschiedlichen Ontologien. Zur Erklärung der folgenden Konzepte innerhalb des hier behandelten Ansatzes wird im Weiteren davon ausgegangen, dass innerhalb der zu vergleichenden Vor- oder Nachbedingung von Anfrage und Servicebeschreibung immer nur eine Ontologie verwendet wird. Dies führt dazu, dass zu einem gegebenen Zeitpunkt maximal zwei Ontologien gleichzeitig miteinander verglichen werden müssen, trotzdem in der Gesamtheit N beliebige Ontologien zum Einsatz kommen können.

Neben der Art und Weise sowie des Zeitpunktes des Matchings muss festgelegt werden, in welcher Form das entstehende Alignment (sei es ausschnittsweise oder eine gesamte Ontologie umspannend) in den vorgestellten Discovery Algorithmus eingebunden werden kann. Teilweise hängt diese Entscheidung von dem Zeitpunkt des Matchings ab. So kann bei Ansätzen, in welchen das Matching direkt zum Zeitpunkt des Vergleichs von Anfrage und Servicebeschreibung durchgeführt wird, auch frühestens zu diesem Zeitpunkt eine Einbindung in die Wissensbasis des Discovery Algorithmus durchgeführt werden. Bei Ansätzen, in welchen größere Bestandteile von Ontologien und ganze

Ontologien miteinander verglichen werden, kann bereits vor der Durchführung der Discovery ein Matching durchgeführt werden. So ist es möglich zunächst alle verwendeten Ontologien zu identifizieren und bereits zu Beginn zu vergleichen. Dies kann allerdings zu ähnlichen Problemen einer erhöhten Komplexität führen wie sie im vorangegangenen Absatz bereits erklärt wurden. Da ex-ante meist nicht bekannt ist, welche Konzepte welcher Ontologie miteinander verglichen werden, muss zunächst pauschal ein Matching aller Ontologien untereinander durchgeführt werden. Sinnvoller ist es, ein Matching zweier Ontologien erst dann durchzuführen, wenn erkannt wurde, dass mindestens zwei Konzepte dieser Ontologien in der Discovery miteinander verglichen werden müssen. Der vorliegende Ansatz implementiert dieses Vorgehen und verwendet bestehende Alignments wieder, welche im Rahmen desselben Discovery Prozesses errechnet wurden. Hierfür werden die Alignments zwischengespeichert oder in der existierenden Wissensbasis belassen. Dies ist sinnvoll unter der Annahme, dass die Ontologien im Laufe des Discovery Prozesses nicht verändert werden (wovon generell nicht auszugehen ist).

Um den Discovery Prozess unter Herstellung von Interoperabilität durch das Ontology Matching zu veranschaulichen ist auf Basis des Ablaufdiagramms aus Abschnitt 4.3.1 der veränderte Prozess in Abbildung 70 dargestellt. Zu erkennen ist, dass pro Vorbedingung und Effekt identifiziert wird, welche Konzepte unterschiedliche Ontologien verwenden. Auf Basis dessen wird ein Matching dieser Ontologien entweder hergestellt oder, falls vorhanden, ein bestehendes Alignment aus vorherigen Schritten wiederverwendet. Nachdem das Alignment zur Verfügung steht wird es zur Wissensbasis hinzugefügt. Der Vergleich von Vorbedingung beziehungsweise Effekt von Anfrage und Servicebeschreibung kann dann unter Interoperabilität der Konzepte durchgeführt werden (sofern ähnliche oder gleiche Konzepte identifiziert werden konnten). Anschließend wird im vorliegenden Fall das verwendete Alignment wieder aus der Wissensbasis entfernt, um Ressourcen zu schonen. Allerdings wird eine Kopie des Alignments mit Verweis auf die verglichenen Ontologien zwischengespeichert, um im weiteren Prozess wie erwähnt wiederverwendet werden zu können.

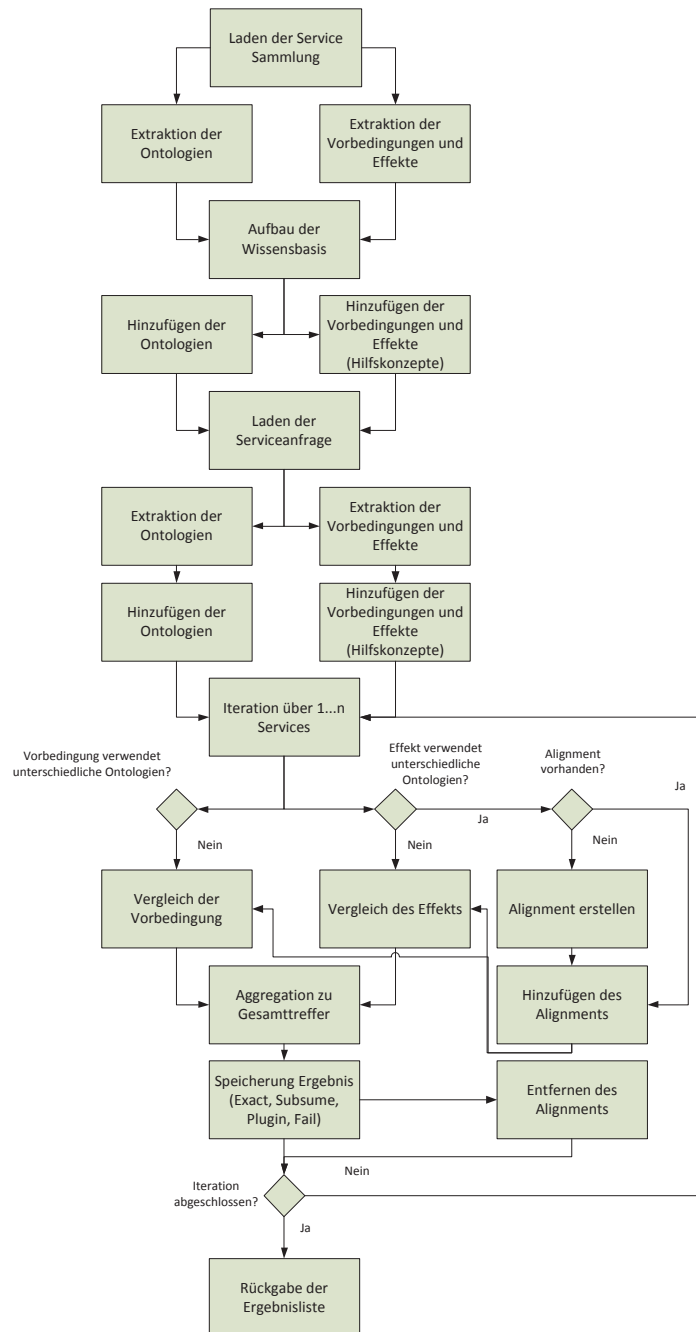


Abbildung 70: Heterogene Service Discovery unter Verwendung von Ontology Alignments

In diesem Abschnitt wurde beschrieben, in welcher Art, zu welchem Zeitpunkt und mit welcher möglichen Komplexität ein Matching von Ontologien zur Herstellung von Interoperabilität in einem Service Discovery Prozess eingebunden werden kann. Weiterhin ist zu klären, unter dem Einsatz welcher Techniken oder bestehenden Werkzeuge das Matching während des Prozesses konkret hergestellt wird. Mit diesem Thema beschäftigt sich der folgende Abschnitt.

4.3.3 Umsetzung des Ontology Matchings

Generell existieren drei Strategien zur Herstellung der Alignments im Rahmen des Service Discovery Prozesses. (1) Es kann ein eigener Matchingalgorithmus, unter Umständen unter Verwendung bestehender Theorien aus Abschnitt 3.3.1, implementiert werden. (2) Es können bestehende Implementierungen verschiedener Techniken kombiniert werden, um eine eigene Matching-Strategie zu implementieren. (3) Es können existierende Alignment-Werkzeuge wiederverwendet und direkt in den Discovery Prozess eingebunden werden. Theoretisch ist es auch denkbar diese drei Alternativen untereinander zu kombinieren. Wichtig ist diese Unterscheidung, da jede einzelne Alternative unterschiedliche Vor- und Nachteile birgt. Diese sollen im Folgenden kurz diskutiert und anschließend die Auswahl für den vorliegenden Ansatz konzeptionell vorgestellt werden.

Die individuelle Implementierung von Matchingtechniken besitzt vor allem den Vorteil eines speziellen für den Anwendungsfall zugeschnittenen und optimierten Matchings. Es können relevante Techniken identifiziert und miteinander kombiniert werden, um ein hohes Maß an Recall und Precision zu erreichen. Für diesen Fall muss allerdings genau bekannt sein, welche Ontologien im vorliegenden Anwendungsfall verwendet werden und die Anforderungen dürfen sich nicht stark ändern. Der Nachteil liegt neben der Bindung an den optimierten Anwendungsfall vor allem im hohen Aufwand für die (Re-) Implementierungen neuer und bestehender Techniken und der Einbindung in den Discovery-Algorithmus. Spätere Anpassungen für weitere Anwendungsfälle oder zusätzliche Optimierung sind zudem mit hohem Aufwand verbunden. Ferner sind bestimmte Techniken aktueller Alignmentwerkzeuge nur unzureichend dokumentiert oder gar nicht publiziert.

Eine Kombination bestehender Implementierungen verschiedener Matchingtechniken verringert den Aufwand aus Alternative eins deutlich. Durch verschiedene Strategien ist eine Optimierung des Matchings für den beschriebenen Anwendungsfall auch hier möglich, allerdings begrenzt auf die zur Verfügung stehenden Implementierungen (auch abhängig von deren Verfügbarkeit, Offenheit und Programmiersprache). Die Kombination an sich sowie die Integration des Matchings in den Discovery-Algorithmus bergen zwar weniger Aufwand als Alternative eins, dieser ist jedoch immer noch beträchtlich. Wie auch bei Alternative eins sind spätere Anpassungen und Optimierung sehr aufwändig.

Die Wiederverwendung bestehender Alignment-Werkzeuge erzeugt den geringsten Aufwand in der Umsetzung des Matchings unter den vorgestellten Alternativen. Durch die Verwendung standardisierter Schnittstellen ist es zudem möglich verschiedene Werkzeuge einzubinden und zu kombinieren. Somit ist auch die Einbindung neuer Varianten oder neuer Werkzeuge mit wenig zusätzlichem Aufwand verbunden. Weiterhin werden durch diese Alternative die Standardisierungsbemühungen im Rahmen des Ontology Alignment unterstützt. Auch unzureichend oder gar nicht dokumentierte Matchingalgorithmen können auf diese Weise eingesetzt werden, insofern sie das Matching optimieren. Durch die Auswahl verschiedener Werkzeuge ist es möglich, diejenigen auszuwählen welche für bestimmte Anwendungsfälle am besten geeignet sind. Der größte Nachteil dieser Alternative ist die unter Umständen mangelhafte Anpassung und Optimierung für den vorgestellten Anwendungsfall. Man ist auf die Auswahl an bestehenden Alignment-Werkzeugen angewiesen und hat keinen direkten Einfluss auf die Umsetzung der Matchingtechniken und -strategien. Bei der Nutzung einheitlicher Schnittstellen ist die Umsetzung dieser bei bestehenden Implementierungen nicht garantiert und führt möglicherweise zu zusätzlichem Aufwand. Bestimmte Werkzeuge besitzen zudem gar keine API zur Einbindung in den Discovery-Prozess.

Das Ziel des bestehenden Ansatzes ist vornehmlich die Machbarkeit einer Umsetzung der heterogenen Service Discovery zu zeigen. Hierbei ist die Optimierung der Alignmentstechniken sekundär. Eine solche Form der Optimierung fällt in das eigenständige Forschungsfeld des Ontology Alignment, in welchem bereits ausreichende Anstrengungen der Optimierung (auch im Rahmen von Ontology Alignment Wettbewerben) durchgeführt werden. Daher wurde im vorliegenden Fall auf eine eigene Implementierung von Techniken und Strategien⁷³ verzichtet. Stattdessen wird auf eine Auswahl der besten (basierend auf den Ergebnissen der OAEI) Alignment-Werkzeuge zurückgegriffen und somit Alternative drei verfolgt.

⁷³ Eine prototypische Implementierung zur Kombination verschiedener Techniken des Ontology Matching und Alignment begleitet die Kernimplementierung der heterogenen Service Discovery als Machbarkeitsstudie. Siehe dazu Abschnitt 5.2.

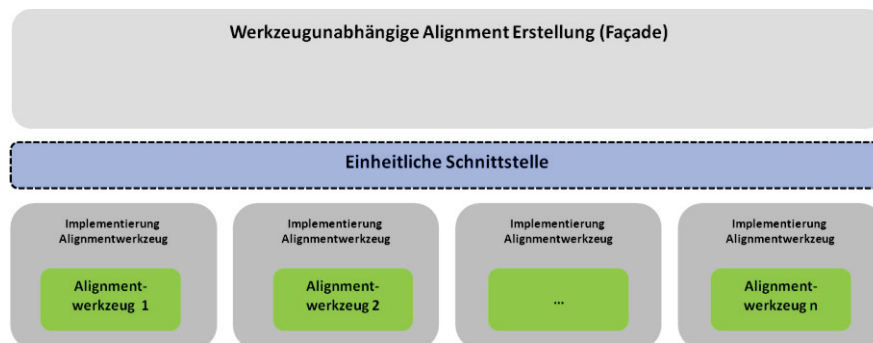


Abbildung 71: Fassadenkonzept zur Einbindung unterschiedlicher bestehender Alignmentwerkzeuge

Die Wiederverwendung von bestehenden Werkzeugen stellt allerdings die Herausforderung der Integration in den bestehenden Service Discovery Prozess dar. Zum Umgang mit der Komplexität bestehender Werkzeuge und der Anforderung möglichst viele unterschiedliche Werkzeuge einbinden zu können, wurde ein modulares Fassadenkonzept entwickelt (siehe Abbildung 71). Hierbei verwendet der Discovery Prozess bei der Erstellung des benötigten Alignments eine werkzeugunabhängige Fassade, welche mittels einer einheitlichen Schnittstelle auf die Implementierungen unterschiedlicher Alignmentwerkzeuge zugreifen kann. Diese Modularität erlaubt neben der einfachen Aktualisierung bereits verwendeter Werkzeuge auch die Einbindung neuer Implementierungen. Eine Detaillierung der Umsetzung einer solchen Fassade findet sich in Abschnitt 5.1 zur prototypischen Implementierung.

4.3.4 Modularisierung zur Verbesserung der Discovery

Für die Erfüllung der Anforderungen aus verschiedenen Anwendungsszenarien ist es unabdingbar, die heterogene Service Discovery kontinuierlich verbessern zu können. Einige Möglichkeiten für diese kontinuierliche Verbesserung wurden bereits kurz in den vorangegangenen Abschnitten angeschnitten. So ermöglicht die modulare Einbindung verschiedener Alignment-Werkzeuge eine einfache Integration neuer Implementierungen und Aktualisierungen bereits eingebundener Werkzeuge. Somit kann je nach Anwendungsfall und zur Verwendung aktueller Erkenntnisse im Forschungsfeld des Ontology Alignment die Discovery verbessert werden. Im Abschnitt zur Umsetzung des Ontology Alignments (siehe Abschnitt 4.3.3) wurde diskutiert, dass neben der Möglichkeit einer Einbindung bereits vollständig implementierter Alignment-Werkzeuge auch eine Kombination von implementierten Matching-Techniken möglich ist. Neben der Verfügbarkeit solcher Implementierungen wurde vor allem der zusätzliche Aufwand bei der Implementierung und der Einbindung in die Discovery genannt.

Aufbauend auf der Idee der modularen Integration von Alignment-Werkzeugen wurde im vorliegenden Ansatz ein Konzept zur Modularisierung von Algorithmen mit wieder-

verwendbaren Bestandteilen entwickelt. Umsetzung von Matchingstrategien durch die Modularisierung von Basistechniken⁷⁴

Die Grundidee dieses Ansatzes ist die Annahme, dass bestehende Ontology Alignment Werkzeuge eine oder mehrere Matching-Strategien einsetzen. Solche Strategien bestehen aus einer Kombination verschiedener Basistechniken des Matchings (siehe Abschnitt 3.3.1.2). Ziel der Modularisierung dieser Techniken und Strategien ist es, diese nach einem Baukastenprinzip kombinieren zu können. Dadurch soll zum einen der Aufwand der Programmierung von Matching-Systemen verringert und zum anderen die Definition solcher Strategien durch Nicht-Programmierer ermöglicht werden.

Die Grundbausteine dieses Konzeptes sind die erwähnten Basisalgorithmen. Dabei existieren drei verschiedene Formen von Basisalgorithmen: (1) Basismatcher, (2) Combiner und (3) Modifier. Basismatcher basieren auf den Grundtechniken des Matchings und untersuchen jeweils ein Artefaktpaar, werden allerdings auf eine gesamte Ontologie angewendet (alle möglichen Artefaktpaare werden untersucht). Zu diesem Artefaktpaar ermitteln sie einen Ähnlichkeitswert (zwischen 0 und 1 oder Boolean). Artefakte können dabei je nach Matchingtechnik Konzepte, Attribute, Instanzen oder andere Bestandteile einer Ontologie sein. Wird ein Basismatcher angewendet, entsteht eine Ähnlichkeitsmatrix. Combiner sind Algorithmen, welche es ermöglichen verschiedene Ähnlichkeitsmatrizen zu kombinieren. Für eine sinnvolle Kombination sollten sich die Matrizen auf gleiche Objekte beziehen (zum Beispiel Konzepte). Eine mathematische Funktion vereint die korrespondierenden Werte der Matrizen. Modifier werden verwendet, wie bereits der Name impliziert, um Ähnlichkeitsmatrizen beziehungsweise deren Werte zu modifizieren. Dabei können unterschiedliche Modifikationen durchgeführt werden, unter anderem eine Extraktion der „besten“ Werte, eine Normalisierung aller Werte oder auch die Erstellung einer Alignment Matrix (1:1 Ähnlichkeitsbeschreibung).

Eine Matchingstrategie besteht nun aus einer Kombination dieser verschiedenen Basisalgorithmen. Generell werden eine oder mehrere Basismatcher auf einem Ontologiepaar angewendet und entweder zwischen den Matchings oder nachträglich durch Modifier modifiziert oder durch Combiner kombiniert. Der Vorteil in der expliziten Definition dieser Basisalgorithmen und einer Standardisierung der verwendeten Schnittstellen liegt vor allem in der Möglichkeit, schnell neue Strategien zu entwickeln und umzusetzen. Zudem können auch Nutzer ohne Programmierkenntnisse diese Strategien modellieren (zum Beispiel durch eine grafische Repräsentation). Die eindeutige Semantik der Strategien erlaubt die Persistenz dieser und einen einfachen Austausch. Eine Beschreibung der prototypischen Implementierung dieses Ansatzes findet sich in Abschnitt 5.2.

⁷⁴ Konzepte und Implementierung dieser Modularisierung wurden unter Anleitung der Autoren in der Diplomarbeit „Konzeption und Implementierung eines Ontology Matchers“ von Georgy Dobrev entwickelt.

4.4 Benutzerinteraktion

Im Rahmen einer geschäftsprozessorientierten Service Discovery und im Speziellen für den Einsatz heterogener Service Beschreibungen ist eine einfache und zielgeführte Benutzerinteraktion unabdingbar. Dies liegt vor allem an:

- der Zielgruppe für die Suche nach Services. Dazu gehören vor allem Geschäftsprozessexperten, -modellierer und Unternehmensarchitekten (siehe Abschnitt 4.2.1). Diese haben unter Umständen keine Ausbildung in der formalen Beschreibung von Services und der Formulierung von Suchanfragen.
- der Verwendung von (mehreren) Ontologien als Einschränkung für die Formulierung von Suchanfragen sowie die Ergänzung der Modellierung. Dabei muss dem generellen Einsatz von eingeschränktem Vokabular sowie dem speziellen Umstand heterogener Ontologien Rechnung getragen werden.
- der Einbettung der Service Discovery in die Modellierung. Da für die Suche nicht ein separates Werkzeug verwendet werden soll, entstehen besondere Ansprüche an die Modellierungsumgebung, auch was die Weiterverwendung von semantischen Informationen betrifft.

In den folgenden Abschnitten werden Konzepte für kontrolliertes Vokabular, kontextorientierte Annotationen, Autovervollständigung und Weiterverwendung als Bestandteile der Benutzerinteraktion für die heterogene GPM-getriebene Service Discovery vorgestellt. Dabei ist zu beachten, dass dies kein vollständiges User Experience Design darstellt, welches in einem finalen Produkt notwendig wäre. Vielmehr geht es darum, die Grundlagen für verbessertes Benutzererlebnis bereits in der prototypischen Implementierung zu schaffen.

4.4.1 Kontrolliertes Vokabular

Der Einsatz eines sogenannten kontrollierten Vokabulars bietet sich im Rahmen des vorliegenden Ansatzes besonders an. Zum einen unterstützt ein eingeschränktes und strukturiertes Vokabular die Modellierung von Geschäftsprozess, Service und Serviceanfragen. Zum anderen sind Ontologien eine spezielle Form eines solchen Vokabulars und für das Reasoning während der Servicesuche unabdingbar. Die Welten der Geschäftsprozessmodellierung und (geschäftsprozessorientierten) Servicebeschreibung können auf diese Weise zusammengebracht werden.

Mendling et al. (2009) (S. 478-479) beschreiben die Bedeutung der Kontrolle von sowohl Objekt- als auch Verb-Vokabular im Kontext der Geschäftsprozessmodellierung. Die Kontrolle von Objekten wird in vielen Fällen durch den Einsatz eines Glossars oder Verzeichnisses von technischen und fachlichen Begriffen einer Organisation vorgenommen. Die Erstellung solcher Verzeichnisse kann eine Vorstufe der eigentlichen Prozessmodellierung sein. Alternativ können bestehende Verzeichnisse wiederverwendet oder verfeinert werden. Diese können entweder aus der Organisation stammen (wie bestehende Datenmodelle) oder Produkte externer Standardisierungen sein (wie die XML

Common Business Library⁷⁵). In bestehenden Modellierungswerkzeugen werden solche strukturierten Vokabulare, welche auch gleichzeitig mit dem Prozessmodell verbunden werden, selten unterstützt. Bestehende Glossare und Verzeichnisse eignen sich durchaus für den Einsatz von Objektbeschreibungen in Prozessen. Wenn wie im vorliegenden Ansatz nach Mendling et al. (2009) die Beschreibung der Aktivitäten durch Verb-Objekt-Paare durchgeführt werden soll, so fehlt ein strukturiertes Verb-Vokabular. Wenige Ansätze, wie zum Beispiel das MIT Process Handbook⁷⁶, schlagen auch Hierarchien der Verben basierend auf WordNet vor. Klassifikationen wie die von Levin (1993) können außerdem genutzt werden. Mendling et al. (2009) merken aber an, dass solche Ansätze nach ersten Untersuchungen nur 44%-68% der Verben in realen Szenarien abdecken können. Für den vorliegenden Ansatz wird daher vorgeschlagen, bestehende Ansätze mit domänenspezifischen Objekten und Verben zu erweitern und so intra- und interorganisatorische Vokabulare zu erhalten.

Kontrollierte Vokabulare werden von Muñoz-Avila et al. (2005) auch als mögliche Lösung des sogenannten „Vocabulary Problems“ nach Furnas et al. (1987) genannt. Dieses Problem wurde ursprünglich im Rahmen der Mensch-Computer-Interaktion geprägt. Nutzer konnten nach Furnas et al. (1987) nicht die Eingabeparameter für ein Computerprogramm ermitteln. Dies zeigte, dass nicht berücksichtigt wurde, dass die Nutzer eine Vielzahl an Begriffen für das gleiche Konzept einsetzen würden. Ein kontrolliertes Vokabular, welches ex-ante vereinbart wird, kann dieses Problem lösen. Allerdings merken Muñoz-Avila et al. (2005) an, dass es nicht immer möglich oder gewollt ist, sich auf ein Vokabular zu einigen. Dies führt zu einer möglichen Alternative, welche auch die Motivation für den vorliegenden Ansatz ist: Der Einsatz von Alignments. So ist es für die Interaktion besonders wichtig, dass der Nutzer verschiedene Vokabulare (Ontologien) für die Modellierung einsetzen kann. Ein späteres Alignment in der Service Discovery erlaubt aber weiterhin den Einsatz von Services, welche nicht nach dem ursprünglichen Vokabular formuliert wurden.

Für die Beschreibung der Beziehungen zwischen Elementen kontrollierter Vokabulare stehen unterschiedliche Ansätze zur Verfügung. So existiert mit SKOS (Simple Knowledge Organisation System) ein Framework für Struktur und Inhalt der Vokabulare (Miles und Perez-Aguera (2007)). Im Weiteren wird für die Beschreibung von WSMO Ontologien ausgegangen, welche zunächst die Basisfunktionalität zur Beschreibung von sowohl Verb- als auch Objektvokabularen und vor allem die Kompatibilität zu den Servicebeschreibungen bietet. In der Zukunft kann über die Integration komplexerer Beschreibungssprachen für kontrollierte Vokabulare nachgedacht werden.

Die Benutzerinteraktion über ein kontrolliertes Vokabular erlaubt die Beschränkung der verwendeten Begriffe sowie die Unterstützung und (unter Umständen) Vereinfachung bei der Modellierung. Der Einsatz solcher kontrollierten Vokabulare wird weiterhin durch

⁷⁵ www.xcbl.org

⁷⁶ process.mit.edu

Techniken der kontextorientierten Annotation sowie einer Autovervollständigung unterstützt, welche in den folgenden Abschnitten näher beschrieben werden sollen.

4.4.2 Kontextorientierte Annotation

Der vorliegende Ansatz unterstützt den Nutzer während der Modellierung und für eine vereinfachte Servicesuche mit Annotation bezüglich des Kontexts, in welchem er sich gerade befindet. Ein Kontext kann im vorliegenden Fall explizit oder implizit angenommen werden. Zudem kann ein Kontext entweder auf Basis ausgewählter Ontologien oder durch Analyse der bereits bestehenden Modellierungselemente entstehen.

Im einfachsten Fall wählt der Nutzer zu Beginn oder auch während der Modellierung bereits eine oder mehrere Domänenontologien aus, welche für das zu erstellende Modell passend erscheinen. Die Konzepte stehen anschließend in der Modellierungsumgebung (zum Beispiel in Form eines Ontologie-Navigators) zu Verfügung und können den Elementen des Modells zugeordnet werden (beispielsweise via Drag-and-Drop). Zusätzlich können die Konzepte der ausgewählten Ontologien auch für eine Annotation mittels einer Autovervollständigung (siehe Abschnitt 4.4.3) zur Verfügung gestellt werden. Eine Erweiterung dieser manuellen Auswahl des Kontexts ist die (halb)automatische Erkennung desselben. Hierzu können bestehende Prozesselemente analysiert werden; entweder im bereits bestehenden Modell oder in benachbarten Modellen (zum Beispiel in Modellen des gleichen Projektes der Modellierungsumgebung). Eine Analyse der Elemente ist mittels einfachen String-Matchings oder auch komplexeren Techniken des NLP denkbar, indem die Beschreibungen der Modellelemente mit Konzepten bekannter Ontologien verglichen werden. Identifizierte Domänenontologien können entweder implizit für die Annotation vorgehalten (etwa für die erwähnte Autovervollständigung) oder explizit dem Nutzer zu Auswahl vorgeschlagen werden. Anschließend stehen diese dann wie im ersten Fall dem Nutzer zur Verfügung. Eine komplexere Variante der kontextorientierten Annotation wird von Born et al. (2007) vorgestellt. Hierbei wird der Prozesskontext analysiert und mit Hilfe von Ontologien vor- oder nachgelagerte Prozesselemente identifiziert. So können Annotation oder auch ganze Modellelemente durch den Kontext erkannt und für die Verwendung im Modell vorgeschlagen werden.

Neben der Annotation der Elemente des Geschäftsprozesses ist der Kontext auch für die Formulierung der Serviceanfragen relevant. Im weiteren Sinne kann auch hier von einer Annotation gesprochen werden, welche auf die bestehenden Informationen des Prozesses zurückgreift. Detaillierte Techniken wurden hierzu bereits in Abschnitt 4.2.2 vorgestellt. Dort wurden vor allem die technischen Grundlagen für die Extraktion von Kontextinformationen dargelegt. Daher sei hier nur noch einmal darauf verwiesen, dass diese Informationen auch benutzerfreundlich präsentiert werden müssen. In Abschnitt 4.2.3.2 wurde bereits vorgestellt, wie Kontextinformationen (durch bereits bestehende Annotationen) für Vorschläge von Vorbedingungen und Effekten verwendet werden können. Eine weitere Technik, dem Nutzer Konzepte zur Annotation einfach präsentieren zu können, ist die Verwendung einer Autovervollständigung. Diese wird im nachfolgenden Abschnitt weiter detailliert.

4.4.3 Autovervollständigung

Die Autovervollständigung (*engl.* autocompletion) ist eine Technik, welche bei der Suche in Datenbanken sowie in Suchmaschinen (Google.com) und Katalogen (Amazon.com) eingesetzt wird. Weitere Einsatzmöglichkeiten finden sich in Editoren (sowohl Text als auch für Programmcode) und Browsern. Bei der Technik werden dem Nutzer während der Texteingabe bereits nach wenigen Symbolen Vorschläge unterbreitet, welche möglichen Wörter oder Symbolkombinationen das potentielle Ergebnis seiner Eingabe sein könnte. Damit wird der Prozess der Eingabe unter Umständen immens verkürzt und der Nutzer unterstützt, die „richtigen“ Begriffe zu wählen. Letzteres kann zum Beispiel bei der Vervollständigung von syntaktischen Begriffen beim Editieren von Programmcode hilfreich sein, wobei nicht nur Vorschläge zur Vervollständigung zum Beispiel von Methodenaufrufen gegeben werden, sondern zusätzlich bereits Informationen über benötigte Ein- und Ausgabeparameter der Methoden. Die Autovervollständigung in Suchmaschinen kann dem Nutzer neben der Beschleunigung des Suchprozesses auch einen Einblick in die Häufigkeit beziehungsweise Beliebtheit bestimmter Suchbegriffe geben. So ordnen Suchmaschinen die vorgeschlagenen Begriffe der Autovervollständigung nach genau diesen Kriterien. Weiterhin ist die Ermittlung von häufigen und beliebten Begriffen eine Methode um eine Grundvoraussetzung für die Autovervollständigung zu erfüllen: Eine endliche Anzahl an Begriffen. Diese kann per Definition des Vokabulars erreicht (endliche Anzahl an Methoden einer Programmiersprache, endliche Anzahl an Einträgen eines Wörterbuchs) oder künstlich durch die eben besprochene Häufigkeit von Begriffen (bei Suchmaschinen die Top 5 möglichen Begriffe einer Stringkombination) erzeugt werden. Ein Beispiel für die Autovervollständigung bei der Suche findet sich im Patent von Ortega et al. (2003), siehe Abbildung 72.

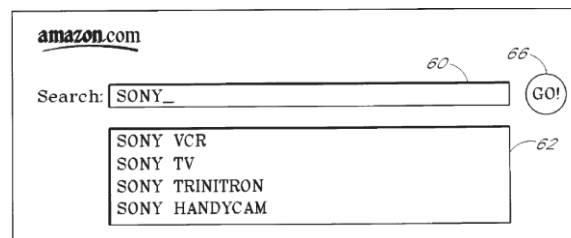


Abbildung 72: Autovervollständigung bei der Suche nach Produkten, nach Ortega et al. (2003), S.1.

Für die Verwendung von kontrollierten Vokabularen im Allgemeinen und Ontologien im Speziellen bietet sich die Autovervollständigung in unterschiedlichen Bereichen der Benutzerinteraktion sinnvoll an. Sie kann zum einen helfen sowohl Prozesse als auch Services mit Ontologiekonzepten zu annotieren. Zum anderen ist die Autovervollständigung eine Möglichkeit formale Suchanfragen durch die Verwendung von bestehenden Konzepten einer Ontologie zu unterstützen. Allerdings besitzen Ontologien die besondere Eigenschaft vordefinierter Hierarchien und unter

Umständen Querverbindungen der Konzepte. Daher wird im vorliegenden Ansatz eine besondere Form der Autovervollständigung nach Sinkkilä et al. (2008) eingesetzt.

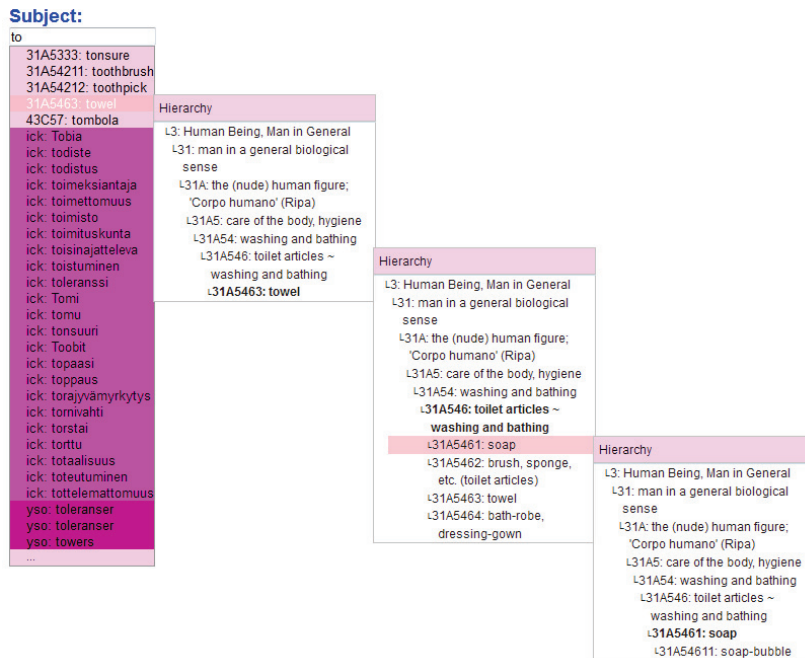


Abbildung 73: "In-Place" Autovervollständigung nach Sinkkilä et al. (2008)⁷⁷.

Die sogenannte „In-Place“ Autovervollständigung basiert auf der Annahme, dass Nutzer zur Annotation bestimmte Konzepte verwenden wollen, unter Umständen aber nicht die genaue Bezeichnung des Konzeptes wissen oder nicht wissen, ob das von ihnen verwendete Konzept auch in der Ontologie existiert. Wenn Ontologien durchsucht werden, können verwandte Konzepte und Hierarchiebeziehungen verwendet werden, um den Nutzer bei der Annotation zu unterstützen. Um einen einfachen Zugang zu Hierarchie und verwandten Konzepten eines Begriffes zu erhalten, schlagen Sinkkilä et al. (2008) vor, die Autovervollständigung mit Ontologiebrowsern zu verbinden, welche eine Navigation durch die verschiedenen Ebenen einer Ontologie erlauben. Das Ergebnis ist in Abbildung 73 zu sehen. Die Anfangsbuchstaben „to“ schlagen eine Reihe von Konzepten zur Auswahl vor. Das System erlaubt es zusätzlich durch einen einfachen Klick ein Konzept in der Ontologiehierarchie anzuzeigen und zu weiteren Konzepten dieser zu navigieren. So kann der Nutzer unterstützt werden, einen für ihn passenden Begriff unter Einhaltung der verwendeten Ontologien auszuwählen. Sinkkilä et al. (2008) merken an, dass je nach Ontologiedomäne auch andere Darstellungsformen als

⁷⁷ Beispiel erstellt unter http://demo.seco.tkk.fi/irma/app/irma_iconclass.html

eine reine hierarchische Form Verwendung finden können. So bietet es sich in einer geografischen Ontologie an, die Autovervollständigung mit Orten „in der Nähe“ oder der Darstellung auf einer Karte zu erweitern.

Im vorliegenden Ansatz wird die Arbeit von Sinkkilä et al. (2008) adaptiert, vor allem um den Nutzern die Möglichkeit zu geben Konzepte aus verschiedenen Ontologien auszuwählen. Zudem wird vorgeschlagen, neben der Einordnung in eine Hierarchie dem Nutzer auch die Beschreibung des jeweiligen Konzeptes zu präsentieren, zum Beispiel unter Verwendung eines „Mouse-over“-Feldes, welches erscheint, wenn der Cursor sich über einem Konzept befindet. So kann vor allem bei semantisch nicht eindeutigen Begriffen die Auswahl erleichtert werden.

4.4.4 Weiterverwendung

Die bisherigen Unterabschnitte zur Benutzerinteraktion bezogen sich vornehmlich auf die Phasen vor beziehungsweise während der Suche nach Services. Ein kontrolliertes Vokabular, kontextorientierte Annotationen und Autovervollständigung können helfen, Suchanfragen zu formulieren und passende Services zur Unterstützung von Geschäftsprozessen zu finden. Zur Benutzerinteraktion gehört aber auch die Weiterverwendung der gefundenen Services.

Im vorliegenden Ansatz ist die Verwendung von Services relevant für die Automatisierung oder Teilautomatisierung von Geschäftsprozessen. Die Suche wird allerdings in einer Umgebung durchgeführt, welche vornehmlich für die Geschäftsprozessmodellierung gedacht ist. Da nicht zwingend von einer einheitlichen Modellierungs- und Entwicklungsumgebung ausgegangen werden kann, wird eine Weiterverwendung von Services für die technische Implementierung der Prozesse betrachtet. Für die spätere Umsetzung bedeutet dies, dass relevante Services aus einer Ergebnisliste ausgewählt, mit Aktivitäten innerhalb des Geschäftsprozesses verbunden und Informationen zu den verbundenen Services in den Aktivitäten persistent gespeichert werden sollen.

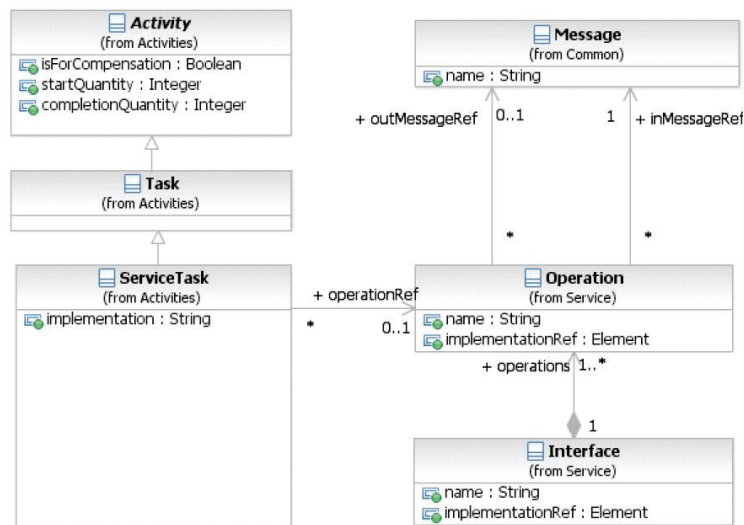


Abbildung 74: Klassendiagramm der „Service Task“ der BPMN nach OMG (2011), S. 159.

Konzeptionell wird für die Weiterverwendung der Serviceinformation die Service Task der BPMN 2.0 (OMG (2011), S. 158-159) eingesetzt. Der Nutzer kann aus der Ergebnisliste seiner Suche eine zu verwendende Serviceoperation auswählen und beispielsweise per Drag-and-Drop mit der gewünschten Aktivität innerhalb des Prozesses verbinden. Diese wird so, wenn nicht schon vorher definiert, zu einer Service Task mit einer „##WebService“-Implementierung als Typisierung (andere Technologien sind denkbar, werden aber hier im Detail nicht betrachtet) umgewandelt. Weiterhin werden die Informationen aus der angewählten Serviceoperation in der Ergebnisliste in die Service Task übernommen. Die URI (Uniform Resource Identifier) der Operation wird in der Operation-Klasse von BPMN abgelegt (im Attribut *implementationRef*), zusammen mit dem Namen der Operation. Somit kann die Service Task über das Attribut *operationRef* mit der Operation-Klasse verknüpft werden. Zusätzlich sollte die Operation mit einer bereits vorhandenen oder zu erstellenden Interface-Klasse verbunden werden, welche den Service der Operation beschreibt. Dieses Interface besitzt auch eine Referenzierung auf die Implementierung des Service (*implementationRef*), für WSDL meist in Form des Porttyps. In der Theorie sind auch mehrere Referenzierungen der Implementierung des gleichen Interface erlaubt. Die entstehende Persistenz im verwendeten BPMN 2.0 Diagramm erlaubt somit die Weiterverwendung der gefundenen Serviceoperationen für die spätere Automatisierung des Prozesses.

5 Prototypische Implementierung

Im folgenden Kapitel wird die prototypische Implementierung der Konzepte dieser Arbeit beschrieben. Der erste Teil der Implementierung (Umsetzung GPM-getriebener heterogener Service Discovery) wurde als Prototyp im Rahmen eines Industrieprojektes realisiert und kann daher nur bis zu einer gewissen Detaillierung beschrieben werden, welche die Rechte der Auftraggeber des Projektes nicht verletzt. Daher können keine Screenshots oder Auszüge aus dem Programmcode dargestellt werden. Allerdings ist es möglich die technischen Komponenten des Modells, deren Architektur und Zusammenspiel zu beschreiben sowie die Interaktion mit dem Benutzer schematisch zu skizzieren. Der erste Teil der Implementierung deckt den Großteil der Konzepte dieser Arbeit ab.

Der zweite Teil der Implementierung (Modulares Ontology Matchmaking System) ist unabhängig von dem oben genannten Projekt entstanden und erlaubt somit eine höhere Detailtiefe. Im Rahmen des Gesamtkonzeptes stellt er eine Ergänzung der Umsetzung der GPM-getriebenen heterogenen Service Discovery dar. Es wird vor allem beschrieben, wie mit der Implementierung Strategien für das Matching im Rahmen der heterogenen Service Discovery einfacher erstellt und angepasst werden können.

5.1 Umsetzung GPM-getriebener heterogener Service Discovery

Für ein besseres Verständnis der Implementierung werden im Folgenden zunächst die relevanten Anforderungen für die Umsetzung beschrieben. Anschließend erfolgt eine Erklärung der Architektur sowie ihrer einzelnen Komponenten. Dies beinhaltet auch eine kurze Beschreibung der Umsetzung der einzelnen Komponenten. Weiterhin wird skizziert, wie die Komponenten in ein bestehendes Modellierungswerkzeug integriert wurden, um die Benutzerinteraktion zu vereinfachen.

5.1.1 Anforderungen

Die prototypische Umsetzung der GPM-getriebenen heterogenen Service Discovery erfolgte unter Anforderungen, welche zum Einen aus den vorgestellten theoretischen Konzepten und zum Anderen aus Gesprächen mit Experten des bereits erwähnten Industrieprojektes entstanden sind. Aus den Anforderungen ergaben sich unterschiedliche Möglichkeiten der Umsetzung einer Architektur des Prototyps. Teilweise werden die Anforderungen bereits durch die theoretischen Konzepte des Kapitels 4 adressiert, dennoch ist eine Umsetzung des Konzepts im Rahmen der Implementierung für einen Nachweis der Möglichkeit der Erfüllung (Proof-of-Concept) unabdingbar. Die letztlich verwendete Architektur wird anschließend in Abschnitt 5.1.2 beschrieben. Die Anforderungen werden nachfolgend tabellarisch aufgelistet und kurz erläutert.

Anforderung	Beschreibung
A1	Formale Darstellung der Funktionalität eines Service Die Funktionalität eines Service soll im Rahmen von semantischen Annotationen darstellbar sein.
A2	Möglichkeit der Speicherung und des Zugriff auf die Meta-Daten der Funktionalität Services sowie deren semantische Annotationen in Form von Meta-Daten müssen persistent und auffindbar sein, z.B. über den Zugriff eines Verzeichnisses. Dies beinhaltet syntaktische und semantische Informationen gleichermaßen.
A3	Suchfunktionalität (einfach und komplex) über den Meta-Daten Die gespeicherten Informationen über die Funktionalität (Meta-Daten) müssen für eine Suche zugänglich gemacht werden. Einfache Suchfunktionalitäten wie eine Schlüsselwortsuche und komplexe Suchfunktionalitäten mittels intelligentem Reasoning müssen möglich sein.
A4	Realisierung einer gemeinsamen Suchbasis für alle Services Sollten Services mit unterschiedlichen Beschreibungen (vor allem in Form von unterschiedlichem Vokabular) existieren, muss gewährleistet sein, dass eine gemeinsame Suchbasis über allen Services entstehen kann.
A5	Gewährleistung von syntaktischer und struktureller Interoperabilität der Services Grundsätzlich können Services unterschiedlicher Anbieter zunächst nur interagieren, wenn deren Syntax und Struktur interoperabel ist, u.a. wenn Datenobjekte einer Domäne auch mit Services einer anderen Domäne kompatibel sind.
A6	Gewährleistung der semantischen Interoperabilität der Services Vor allem die semantischen Modelle bei der Beschreibung der Funktionalität der Services müssen für eine erfolgreiche Suche auf einer heterogenen Suchbasis interoperabel sein.
A7	Integration einer Werkzeugunterstützung zur Lösung noch bestehender semantischer Interoperabilitätsprobleme Ist eine vollständige Interoperabilität der semantischen Modelle ex-ante

	nicht zu gewährleisten so soll ein entsprechendes Werkzeug zur Herstellung dieser Interoperabilität in die Implementierung integriert werden.
A8	Möglichkeit der Erstellung, Speicherung und Ausführung ⁷⁸ von Servicekompositionen
	Aufbauend auf den Konzepten der Service-Suche und Service-Interoperabilität soll es möglich sein, Kompositionen von Services erstellen, speichern und ausführen zu können.
A9	Verwendung einer bestimmten Servicegranularität zur Erstellung sinnvoller Servicekompositionen
	Um Services sinnvoll kombinieren zu können, müssen diese auf ähnlichen Granularitätsebenen interagieren. Es ist meist nicht sinnvoll rein technische Services mit Geschäftsservices verbinden zu wollen.
A10	Etablierung von Voraussetzungen für zukünftige Vorschläge für Servicekompositionen
	Es bestehen bereits Algorithmen, welche Vorschläge zur Komposition von Services liefern können. Voraussetzungen für die Nutzung solcher Algorithmen sollen geschaffen werden.
A11	Auswahl einer semantischen Annotation zur Beschreibung der Funktionalität eines Service
	Dies ist eine Detaillierung von Anforderung A1. Die ausgewählte Beschreibungssprache muss alle möglichen Bestandteile zur Beschreibung der Funktionalität von Services im untersuchten Anwendungsfall beinhalten.
A12	Möglichkeiten zum Ausdruck und Verteilung von Wissen bereitstellen
	Sehr allgemein gehalten wird ausgedrückt, dass ein semantisches Modell hinter den Annotationen erstellt werden kann, meist in Form einer expliziten Wissensbasis.
A13	Möglichkeiten zur Erstellung, Speicherung, Zugang und Suche auf der Wissensbasis bereitstellen
	Analog zu Anforderungen A2 und A3 für die Meta-Daten eines Service müssen auch Möglichkeiten bestehen, Wissen in der Wissensbasis zu

⁷⁸ Obwohl die Ausführung zunächst in den Anforderungen der Implementierung festgehalten wurde, hat sich im Verlauf des Projektes herausgestellt, dass eine solche Umsetzung den Umfang der Implementierung übersteigen würde. Daher wurde dieser Teilaspekt der Anforderung in der späteren Implementierung nicht umgesetzt.

	erstellen, speichern, suchen und abzurufen.
A14	Sicherstellung einer effizienten Methode der Annotation von Service im Hinblick auf Zeit und Kosten für diese Annotation
	Obwohl nicht im Fokus ist es dennoch wichtig, Balance zwischen Ausdrucksfähigkeit und Zeit für die Annotation zu finden. Diese Anforderungen beziehen sich auf die konzeptionelle Form der Annotation und mögliche Werkzeugunterstützung zur Beschleunigung dieser.
A15	Integration der Anforderungen in einen einheitlichen Ansatz
	Für alle formulierten Anforderungen muss geprüft werden, wie diese in einem einheitlichen Ansatz (und Werkzeug) integriert werden können. Diese Einheitlichkeit soll auch eines der Alleinstellungsmerkmale der Implementierung sein.

Tabelle 9: Anforderungen an die prototypische Implementierung einer heterogenen Service Discovery

Die in Tabelle 9 beschriebenen Anforderungen beschäftigen sich vornehmlich mit der Realisierung einer Service Discovery auf Basis heterogener semantischer Beschreibungen. Im Verlauf der Arbeit in Projekten und an der Dissertation hat sich zusätzlich der spezielle Fall der geschäftsprozessorientierten oder auch geschäftsprozessmodellierungsgetriebenen heterogenen Service Discovery herausgebildet. Daher werden die Anforderungen der Vollständigkeit halber noch von Tabelle 10 ergänzt. Diese sind in Teilen bereits konkreter als die initialen Anforderungen in Tabelle 9.

Anforderung	Beschreibung
A16	Ausrichtung der Servicebeschreibung an geschäftsrelevanten Objekten
	Dies ist teilweise eine Verfeinerung von Anforderung A9, indem zur Beschreibung der Funktionalität der Services vornehmlich Geschäftsobjekte und Informationen im Kontext dieser betrachtet werden.
A17	Möglichkeit der Wiederverwendung semantischer Informationen im Rahmen einer Geschäftsprozessmodellierung
	Wird die Service Discovery in den Kontext der GPM eingebettet so erscheint es naheliegend, dass bestehende semantische Informationen einer solchen Umgebung wiederverwendet werden sollen.
A18	Integration der Service Discovery in ein Geschäftsprozessmodellierungswerkzeug
	Für eine benutzerfreundliche Anwendung der GPM-getriebenen

Service Discovery soll diese in einem (wenn möglich bereits bestehenden) Werkzeug zur Modellierung von Geschäftsprozessen integriert werden.

Tabelle 10: Ergänzende Anforderungen im Rahmen der GPM-getriebenen Service Discovery

Nachdem die groben Anforderungen erläutert wurden, wird im folgenden Abschnitt eine Architektur vorgestellt, welche diese (größtenteils) erfüllen kann. Während der Entwicklung musste sich aufgrund von technischen Möglichkeiten und Umfang der Implementierung von wenigen Anforderungen gelöst werden (zum Beispiel der direkten Ausführbarkeit der Servicekompositionen). Zudem sind weitere, vor allem technische, Anforderungen hinzugekommen, welche aber im Weiteren nur am Rande erwähnt werden sollen.

5.1.2 Architektur

Der folgende Abschnitt gibt einen Überblick über die Gesamtarchitektur der Implementierung und detailliert die einzelnen Komponenten einer solchen Architektur. Die Architektur ist in Abbildung 75 zunächst grob skizziert.

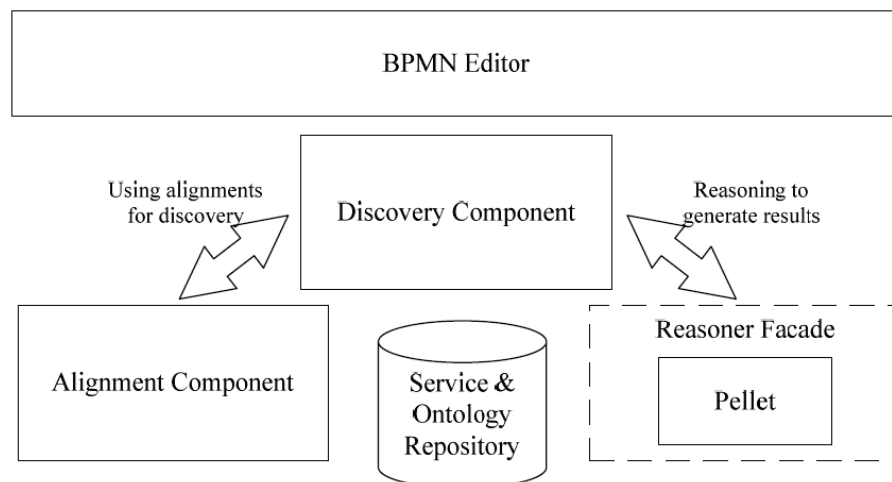


Abbildung 75: Architektur der prototypischen Implementierung

Die generelle Architektur besteht aus fünf grundlegenden Komponenten: Service-Discovery-Komponente, Ontology-Alignment-Komponente, Reasoner (Facade mit entsprechender Implementierung, hier Pellet), Service & Ontology Repository sowie die Integration in einen BPMN Editor. Bevor jede einzelne Komponente im Detail beschrieben wird, soll zunächst das Zusammenspiel erklärt werden.

Der Einstieg für die Suche nach Services geschieht über den BPMN Editor, im Speziellen über eine Ansicht zur Modellierung von Geschäftsprozessmodellen, welche um Komponenten der Service Discovery erweitert wurde. Verschiedene Einstiegspunkte im Modell und der erweiterten Ansicht, welche später genauer erläutert werden sollen, ermöglichen die Formulierung von Suchanfragen und das Initialisieren einer Suche nach Services beziehungsweise Serviceoperationen. Bei der Formulierung von Suchanfragen beziehungsweise der Annotation des Geschäftsprozessmodells wird bereits auf das Ontologie-Repository zugegriffen. Wird eine Suchanfrage gestartet, kommt die Service-Discovery-Komponente zum Einsatz. Diese überprüft zunächst, ob sich Anfrage und Services im Service-Repository auf semantischer Ebene unterscheiden (unterschiedliche Ontologien verwenden). Ist dies der Fall wird außerdem überprüft, ob bereits ein Alignment zwischen den Ontologien zur Verfügung steht. Steht noch kein Alignment zur Verfügung kommt die Alignment-Komponente zum Einsatz. Unter Nutzung verschiedener Werkzeuge werden die passenden Alignments (auf n:m Basis) ermittelt und der Service-Discovery-Komponente übergeben. Diese kann anschließend eine Suche auf den Services des Service-Repositories starten. Hierzu wird in der Reasoner-Komponente eine Wissensbasis mit existierenden Ontologien, den Vorbedingungen und Effekten der Anfrage sowie der Services aufgebaut sowie mögliche Alignments der Ontologien eingebunden. Diese Wissensbasis dient als Grundlage für den Vergleich zwischen Anfrage und angebotenen Funktionalitäten der Services in Form ihrer Vorbedingungen und Effekte. Der Reasoner ermöglicht verschiedene Trefferkategorien, welche für die Anfrage temporär in Verknüpfung mit den Serviceoperationen gespeichert werden können. Auf Basis dieser Treffer generiert die Service-Discovery-Komponente eine Ergebnisliste, welche die Treffer der Vorbedingungen und Effekte kombiniert. Diese Ergebnisliste wird sortiert und mit weiteren nötigen Informationen (Operations- und Servicennamen, Operations- und Servicebeschreibungen, URIs, etc.) der integrierten Sicht im BPMN Editor übergeben. Auf die Anfrage des Nutzers kann nun die Ergebnisliste dargestellt werden. Diese lässt eine Sortierung der Ergebnisse sowie die manuelle Untersuchung der Passgenauigkeit für die initiale Anfrage zu. Wurde eine passende Serviceoperation gefunden, so kann diese in das Geschäftsprozessmodell übernommen werden und steht für eine weitere Verarbeitung zur Verfügung. Vom Nutzer können nun weitere Anfragen im gleichen Modell gestartet werden, welche unter Umständen auch die erste Anfrage verfeinern und ersetzen. Die Architektur erlaubt somit eine kontinuierliche Anreicherung und Verfeinerung des Geschäftsprozessmodells mit Serviceoperationen, welche zum Ziel hat den Prozess zumindest teilautomatisieren zu können.

Dieser grobe Überblick über die verwendeten Komponenten soll nachfolgend pro Komponente weiter detailliert und mit Informationen zu Implementierungstechnologien ergänzt werden. Die folgenden Abschnitte basieren auf der bereits erfolgten Publikation des Ansatzes in Rake et al. (2009).

5.1.2.1 Service & Ontologie Repository

Das Repository beinhaltet alle Servicebeschreibungen, auf welchen eine Suche während des Discovery-Prozesses durchgeführt werden kann. Weiterhin sind alle nötigen Onto-

logien für die Servicebeschreibung und, sobald sie von den ersteren abweichen, weitere Ontologien für die Formulierung von Serviceanfragen enthalten. Die Servicebeschreibungen sind durch WSMO Web Services formalisiert, wobei eine für die spätere Verarbeitung wichtige Anpassung beziehungsweise Annahme realisiert wurde: Viele bestehende Ansätze verwenden pro Service eine oder mehrere Vorbedingungen und Effekte. Näher an der Realität ist allerdings die Annahme, dass Services meist aus mehr als einer Operation bestehen und somit unterschiedliche Vorbedingungen und Effekte pro Operation benötigen. Diesem Umstand wird im vorliegenden Ansatz Rechnung getragen, indem Vorbedingungen und Effekte pro Operation durch die Verwendung von Suffixe formalisiert wurden (siehe hierzu auch Abschnitt 4.1.4). Das Repository lässt eine Speicherung dieser im Rahmen der normalen WSMO Web Service Beschreibungen zu, allerdings muss die Discovery Komponente diese Suffixe interpretieren können.

Generell sind mehrere Vorbedingungen und Effekte pro Operation zulässig, in der vorliegenden Referenzimplementierung kann aber nur jeweils eine Vorbedingung und ein Effekt interpretiert werden. Komplexere Vorbedingungen und Effekte können aber dennoch durch die Verkettung logischer Ausdrücke entstehen. Die vorliegende Implementierung erlaubt den Ausdruck der Vorbedingungen und Effekte in Form von Description Logic mit der zusätzlichen Richtlinie, dass zur Beschreibung der Funktionalität immer eine Kombination aus mindestens einem Geschäftsobjekt und zugehörigem Status zu verwenden ist. Für die Anfragen besteht eine solche Beschränkung nicht.

Im Rahmen der Implementierung wurden 22 reale Web Services von zwei unterschiedlichen Serviceanbietern zum Testen bereitgestellt und in Form von WSMO Web Service im Ontologie-Repository hinterlegt. Da der Ansatz auf Operationsebene arbeitet wurden für insgesamt 40 Operationen der 22 Web Services Vorbedingungen und Effekte in Absprache mit den Experten der Serviceanbieter erstellt. Diese basieren auf der textuellen Beschreibung der Semantik, den verwendeten Geschäfts- und Datenobjekten sowie auf dem Verständnis der Experten über die Semantik der Services. Dem Autor ist bewusst, dass dies nur ein kleiner Ausschnitt der zukünftigen Services in einer SOA darstellt. Allerdings waren diese Services ausreichend um verschiedene Szenarien darstellen und testen zu können. Weiterhin mussten auch vor allem Services ausgewählt werden, welche zumindest eine gewisse Überschneidung zwischen den Anbietern besaßen, um eine heterogene Service Discovery überhaupt zu ermöglichen⁷⁹. Für beide Anbieter wurde jeweils eine umfangreiche Domänenontologie erstellt. Die zwei verwendeten Domänen waren Enterprise Services (vor allem im Bereich des Order Management) und Telecommunication Services (vor allem Call Center Management und Präsenzdienste)⁸⁰.

⁷⁹ Bei der Verwendung von Services ohne semantische Überschneidung würde der Discovery Algorithmus nicht anders als in traditionellen Ansätzen arbeiten und der Mehrwehrt einer heterogenen Service Discovery könnte nicht demonstriert werden.

⁸⁰ Beispiele dürfen aufgrund von Vertraulichkeitsvereinbarungen nicht gegeben werden.

Der Zugriff sowohl auf Service- als auch Ontologie-Repository wird über eine integrierte Sicht im BPMN Editor ermöglicht. Hierbei wird neben einer Übersicht über die geladenen Ontologien und Services dem Nutzer die Möglichkeit gegeben, neue Ontologien und Services hinzuzufügen, zu entfernen und Details dieser anzeigen zu lassen. Die Möglichkeit zur Editierung wurde dem Nutzer explizit nicht gegeben, da dies unter einer zentralen Stelle durchgeführt werden sollte, um Probleme unterschiedlicher Versionierung zu verhindern. Details zur Benutzerinteraktion mit dem Repository werden im Abschnitt zur BPMN Editor Komponente gegeben.

Die Implementierung des Repository wurde unter Verwendung der WSMO4j API in Java durchgeführt. Zugriffe sind über die Methoden der Repository-Klasse oder über eine Kapselung in Form eines Web Services möglich, um die Einbindung in eine serviceorientierte Architektur zu erleichtern. Eine Verwendung bestehender WSMO Repository ist möglich, da keine Anpassungen an den WSML Servicebeschreibungen oder Ontologien gemacht werden, welche nicht auch durch ein standardisiertes WSMO Repository abgedeckt werden können. Zu Beginn der Implementierung wurde versucht das Repository des WSMX (Haller et al. (2005)) wiederzuverwenden. Aufgrund von Problemen mit der Stabilität dieser Lösung wurde der Ansatz wieder verworfen.

5.1.2.2 Reasoning-Komponente

Die Reasoning-Komponente ist ein integraler Bestandteil für die Durchführung der Service Discovery. Sie bietet die Möglichkeit eine Wissensbasis aus Ontologien sowie logischen Ausdrücken (extrahiert aus Vorbedingungen und Effekten) aufzubauen. Auf Basis einer solchen Wissensbasis können Anfragen über die Ähnlichkeit zwischen logischen Ausdrücken und ontologischen Konzepten durchgeführt werden. Je nach verwendeter Regelsprache ist es so möglich, neben exakten Übereinstimmungen auch Subsumptions- und Plugin-Treffer zu generieren. Die Steuerung des Reasoners erfolgt über fest definierte Schnittstellen durch die Discovery Komponente, was die Nutzung unterschiedlicher Reasoner möglich macht.

Für die prototypische Umsetzung wurde die Java-basierte WSML2Reasoner⁸¹ Implementierung verwendet. Die API des WSML2Reasoner ermöglicht die Nutzung unterschiedlicher Reasoner, welche mittels Facaden eingebunden und somit für die WSML angewandt werden können. Dies ist vor allem nötig um die verschiedenen Dialekte der WSML (DL, Flight, Rule, Core, Full) bedienen zu können. Da ein ausgereifter nativer Reasoner für die im vorliegenden Ansatz verwendete Description Logic zum Zeitpunkt der Implementierung nicht existierte, kommt der WSML2Reasoner mit Pellet⁸² (Sirin et al. (2007)) als OWL Reasoner für Java zum Einsatz.

⁸¹ <http://tools.deri.org/wsml2reasoner>

⁸² <http://clarkparsia.com/pellet>

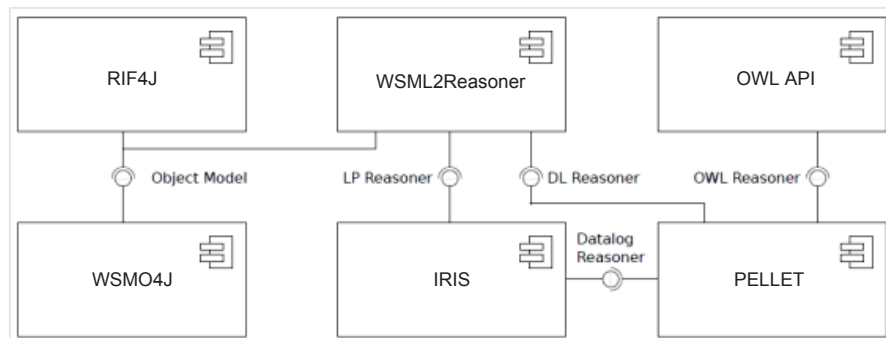


Abbildung 76: Architektur des WSM2Reasoner Framework, angepasst für Pellet nach Krummenacher et al. (2010)

Die WSM2Reasoner stellt eine *WSM2ReasonerFactory* Schnittstelle bereit, welche einen Reasoner für einen gewünschten Dialekt (in diesem Fall DL über die Methode `createDLReasoner`) produziert. Dieser Reasoner erlaubt die Abfrage verschiedener Inferenzen, zum Beispiel ob zwei Konzepte der Wissensbasis äquivalent sind (Methode `isEquivalentConcept`). Für die interne Verarbeitung des Reasoners unter Nutzung von Pellet werden Konzepte und Ausdrücke der Ontologien temporär in OWL-Repräsentationen umgewandelt. Für die im Prototyp zum Einsatz kommenden Beschreibungen bedeutet dies allerdings keine Einschränkung.

In der Architektur existiert bewusst keine Verknüpfung zwischen Reasoner und Alignment Komponente. Sollte für die Erstellung eines Alignments ein Reasoner benötigt werden, so muss das jeweilige Alignment-Werkzeug einen separaten Reasoner bereitstellen. Somit können Probleme unterschiedlicher Versionen des Reasoners verhindert werden.

5.1.2.3 Alignment-Komponente

Die Implementierung der Alignment-Komponente ermöglicht die Umsetzung einer der Hauptanforderungen des Systems: Identifikation von ähnlichen oder gleichen Konzepten in unterschiedlichen Repräsentationen (Ontologien), welche zur Beschreibung der Services und Serviceanfragen verwendet wurden. Erst auf diese Weise wird der Umgang mit heterogenen semantischen Servicebeschreibungen möglich.

Konzeptionell beruht die Umsetzung der Komponente auf den Ausführungen in den Abschnitten 4.3.2 und 4.3.3. Zum besseren Verständnis ist in Abbildung 77 nochmals das Prinzip der Facade vor verschiedenen Alignment-Werkzeugen dargestellt, welches bereits in Abbildung 71 zu sehen war. Wie bereits erwähnt überprüft die Discovery-Komponente, ob für die verwendeten Ontologien ein bestehendes Alignment vorliegt (zum Beispiel im Cache). Ist dies nicht der Fall wird eine Referenz auf beiden Ontologien an die Facade der Alignment-Komponente gesendet. In der aktuellen Implementierung ist das Alignment von maximal zwei Ontologien gleichzeitig möglich, dies ist auch abhängig von dem verwendeten Alignmentwerkzeug. Je nach Einstellung oder An-

forderung in einem solchen Methodenaufwurf wird die Anfrage über eine einheitliche Schnittstelle an das ausgewählte Alignment-Werkzeug weitergeleitet. Nachdem ein Alignment erstellt wurde (welches auch leer sein kann), wird dieses für die Ausführung des Service-Discovery-Prozesses der Discovery-Komponente zur Verfügung gestellt.

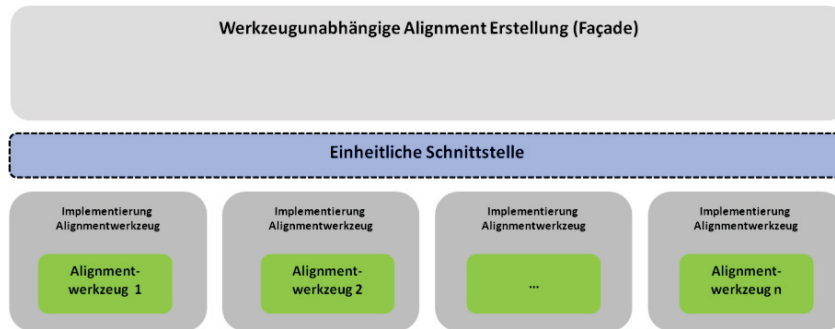


Abbildung 77: Fassadenkonzept zur Einbindung unterschiedlicher Alignment-Werkzeuge

Im Rahmen der prototypischen Implementierung war es wichtig zu zeigen, dass dieses Konzept umgesetzt werden kann. Ein wichtiges Ziel war zudem mehrere Alignmentwerkzeuge zur Auswahl zu haben und für die Zukunft die Einbindung weiterer Werkzeuge zu ermöglichen. Als ersten gemeinsamen Nenner stand das OAEI Framework mit einem standardisierten Alignment Format sowie einer API in Java zu Verfügung (Euzenat (2004), in einer neueren Version nach Euzenat (2006)). Somit stand zum Einen ein möglicher Rückgabeparameter der Schnittstelle fest (Alignment im OAEI Alignment Format) als auch eine Auswahl an Alignment-Werkzeugen welche die Alignment API ganz oder in Teilen bereits implementierten, da sie an der jährlichen OAEI Kampagne teilgenommen hatten. In Abschnitt 3.3.3 wurde bereits erwähnt dass bei der Auswahl an Alignment-Werkzeugen Lily und Falcon-AO selektiert wurden. Beide ermöglichen die Ausgabe eines Alignments im OAEI Alignment-Format.

Die Umsetzung des Fassadenkonzeptes mit den Werkzeugen Lily und Falcon-AO ist in Abbildung 78 dargestellt. Die Facade bildet die Klasse *AlignmentCreator*, welche nach ihrer Instanziierung Zugriff auf die `align()` Methode liefert. An diese werden genau zwei WSML-Ontologien übergeben, für welche ein Alignment durchgeführt werden soll. Der *AlignmentCreator* überwacht die Weitergabe der Anfrage an eines der registrierten Werkzeuge (Falcon-AO als Default), welche als Auflage das Interface *AlignmentTool* implementieren müssen. Die Modulklassen für die Implementierung der jeweiligen Alignment-Werkzeuge sind einmalig manuell zu erstellen. Dies ist notwendig, da viele Werkzeuge zwar das AOEI Alignment Format erzeugen und zum Teil auch eine standardisierte `align` Methode bereitstellen (über das OAEI API Interface *AlignmentProcess* (Euzenat (2004), S. 702)), aber in unterschiedlicher Weise initialisiert werden und Ontologien registrieren. So ist es zum Beispiel für Falcon-AO nötig die verwendeten Ontologien entweder als URI bereitzustellen oder zunächst lokal zu serialisieren, um diese Speicherorte dann referenzieren können. Weiterhin verwenden beide bereits implementierten Werkzeuge OWL-Ontologien für die Erstellung des Alignments. Zur Erfüllung

dieser Anforderung kommt eine Konvertierung von WSMML zu OWL aus der WSMO4j Bibliothek zum Einsatz.

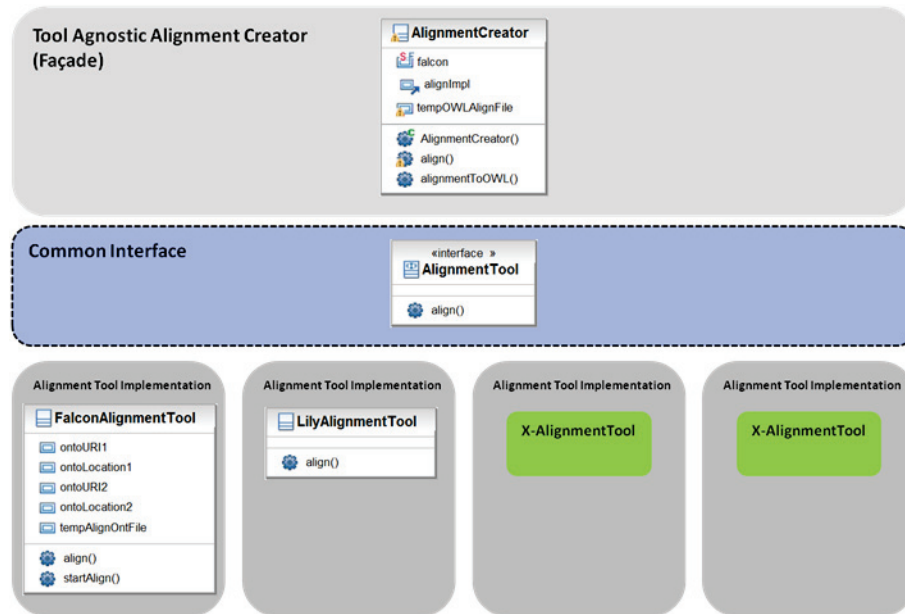


Abbildung 78: Umsetzung des Fassadenkonzepts in der prototypischen Implementierung

Durch die Verwendung der OAEI API ist es zudem möglich, das entstandene Alignment zu verändern. Dies kann zum einen in Form von Grenzwerten (*Thresholding*, mittels `Alignment.cut(double)` (Euzenat (2004), S. 702)) oder Operationen zur Modifikation der Ähnlichkeitswerte geschehen (zu ähnlichen Techniken siehe auch Abschnitt 5.2 im Rahmen des modularen Ontology Matchmaking Systems). Für die erste Iteration der Implementierung ist jedoch keine weitere Modifikation der Alignmentergebnisse der jeweiligen Werkzeuge vorgesehen. Allerdings ist der *AlignmentCreator* geeignet in der Zukunft auch Ergebnisse unterschiedlicher Werkzeuge zusammenzuführen. Für die weitere Nutzung des Alignments im Rahmen der heterogenen Service Discovery kommt eine Methode der API zum Einsatz, welche das OAEI Alignment Format in eine OWL Ontologie mit entsprechenden Regeln umwandelt (siehe Abbildung 78, Methode `alignmentToOWL()`, über die `Alignment.render(visitor)` Methode der OAEI API mittels des `OWLAXiomsRendererVisitor`, (Euzenat (2004), S. 707-708). Da in den aktuellen Werkzeugen nur echte Gleichheit (mit einer entsprechenden Konfidenzaussage) zum Einsatz kommt, beschränken sich die verwendeten Regeln auf `owl:equivalent Class`. Generell sind aber für den Fall der OWL-Umwandlung auch Subsumption und Exklusivität möglich. Ein beispielhaftes Ergebnis dieser Umwandlung ist in Abbildung 79 zu sehen. Die Erstellung einer OWL Ontologie ist hilfreich, da diese über die Discovery Komponente in die Wissensbasis des Reasoner übertragen werden kann. Somit entstehen Gleichheitsbeziehungen für ähnliche Konzepte der verwendeten Ontologien,

welche im Rahmen der Service Discovery verwendet werden können. Auf eine Umwandlung in WSMML Regeln beziehungsweise Ontologien wird verzichtet, da zurzeit standardmäßig der OWL Reasoner Pellet zum Einsatz kommt. Eventuelle Fehlerquellen bei der Umwandlung und Rückumwandlung können so verhindert werden.

```
<rdf:RDF
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"

  <owl:Ontology rdf:about="">
    <rdfs:comment>Aligned ontologies</rdfs:comment>
    <owl:imports rdf:resource="http://www.example.org/ontology1"/>
    <owl:imports rdf:resource="http://www.example.org/ontology2"/>
  </owl:Ontology>

  <owl:Class rdf:about="http://www.example.org/ontology1#reviewedarticle">
    <owl:equivalentClass
      rdf:resource="http://www.example.org/ontology2#article"/>
  </owl:Class>

  <owl:Class rdf:about="http://www.example.org/ontology1#journalarticle">
    <owl:equivalentClass
      rdf:resource="http://www.example.org/ontology2#journalarticle"/>
  </owl:Class>
</rdf:RDF>
```

Abbildung 79: Beispielhafte OWL-Ontologie mit Gleichheitsaxiomen, erstellt aus einem OAEI Alignment, nach Euzenat (2004), S. 708.

5.1.2.4 Service-Discovery-Komponente

Obwohl die Integration der einzelnen Komponenten vornehmlich über den BPMN Editor ermöglicht wird, bildet die Service-Discovery-Komponente das Herzstück des Prototyps. Sie ermöglicht den Vergleich von Service-Anfragen mit den Service-Beschreibungen des Repository. Für die Initialisierung des Prozesses, spätestens zu Beginn der ersten Anfrage, werden die relevanten Ontologien aus dem Repository geladen und am Reasoner registriert. Weiterhin wird überprüft, ob bestehende Alignments für die zu verwendenden Ontologien im Cache der Komponente existieren. Ist dies nicht der Fall, so wird eine Anfrage an die Alignmentkomponente geliefert, welche die bereits erwähnte OWL Ontologie mit Axiomen liefert. Diese wird zusätzlich am Reasoner registriert.

Für die Implementierung wurde der in Abschnitt 4.3.1 beschriebene Algorithmus in Java umgesetzt. Als unterstützende Bibliothek kam hauptsächlich das bereits mehrfach erwähnte WSMO4j (Dimitrov et al. (2006)) zum Einsatz. Wichtig für die Umsetzung ist die Auswahl einer der Varianten zur Herstellung der Interoperabilität nach Abschnitt 4.3.2. Dabei wurden die verschiedenen Vor- und Nachteile abgewogen und eine pragmatische und im Rahmen des Projektumfangs umsetzbare Variante ausgewählt. Zum Einsatz kommt das Alignment der gesamten Ontologien, um möglichst viele existierende Informationen für die Ähnlichkeitsbestimmung verwenden zu können. Wie bereits erwähnt, kann dieses Verfahren zu Performanzproblemen bei vielen und großen Ontologien führen. Da der Prototyp nur mit einer beschränkten Anzahl an Ontologien umgehen kann (aktuell werden maximal zwei Ontologien miteinander verglichen) und die-

se einen moderaten Umfang haben (jeweils <100 Konzepte), bot sich diese Variante für eine reine Machbarkeitsbetrachtung stark an. Um in einem gewissen Rahmen auch zukünftig mit einer größeren Anzahl an umfangreicheren Ontologien umgehen zu können wurde zusätzlich der Einsatz bestehender Alignments implementiert. Um die Qualität der bestehenden Alignments zu garantieren wird nur auf solche zurückgegriffen, welche in vorangegangenen Service-Discovery-Prozessen erstellt wurden. So entsteht ein lokaler (in zukünftigen Versionen eventuell zentraler) Cache von Alignments, auf welche ohne Performanzeinbußen zurückgegriffen werden kann. Falls für zwei Ontologien ein Alignment im Cache existiert, wird dieses auch verwendet, es sei denn es wurden in der Zwischenzeit Änderungen an den Ontologien vorgenommen. Dies setzt ein kontrolliertes Versionsmanagement der Ontologien voraus. Ein solches Caching kann weiterhin nützlich sein falls, in Zukunft neben den Alignmentwerkzeugen Lily und Falcon-AO Ansätze zum Einsatz kommen, welche bereits bestehende Alignments verfeinern können. Mit der Übergabe eines Alignments aus dem Cache können unter Umständen genauere Ergebnisse mit einer fortschreitenden Nutzung der Discovery generiert werden.

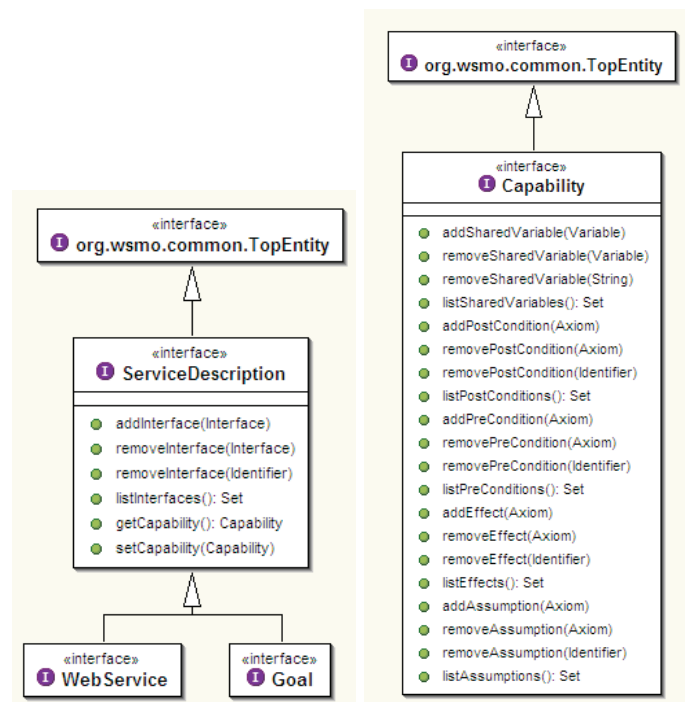


Abbildung 80: Relevante Schnittstellen von WSMO4j für die Extraktion von Vorbedingungen und Effekten nach Dimitrov et al. (2006), S. 30 und 31.

Für die erfolgreiche Durchführung des Service-Discovery-Prozesses ist eine Extraktion der Vorbedingungen und Effekte der Serviceoperationen notwendig, um diese mit denen der Anfrage zu vergleichen. Hierzu kommen vornehmlich die in Abbildung 80 dargestellten Schnittstellen von WSMO4j zum Einsatz. Wichtig ist hier zu erwähnen, dass

die bereits vorgestellten Annahmen zur Definition von Vorbedingungen und Effekten der Services, insbesondere des Einsatzes von Suffixes für die Identifikation von Serviceoperationen, besondere Anforderungen an die Extraktion stellen. Für die interne Verarbeitung wird zunächst die *Capability* eines Service mittels `getCapability()` extrahiert. Anschließend werden die Vorbedingungen und Effekte über die Methoden `listPreconditions()` und `listEffects()` ausgelesen. Durch die Suffixe *Pre* und *Eff* können die Bedingungen klassifiziert und aufgrund ihrer eindeutigen Benennung geordnet werden. Somit ist eine Iteration auf allen Serviceoperationen möglich. Für die Anfrage wird eine analoge Extraktion durchgeführt, wobei maximal je eine Vorbedingung und ein Effekt vorhanden sind. Die Voraussetzungen für die Service Discovery sind somit geschaffen.

Der detaillierte Programm- oder auch Pseudocode für den Algorithmus der Service Discovery kann aus urheberrechtlichen Gründen in dieser Arbeit leider nicht zur Verfügung gestellt werden. Generell ist es aber vorstellbar, dass der theoretische Ansatz zur Einbindung von Ontology Alignments in einen Service-Discovery-Prozess auch in anderen Umgebungen (zum Beispiel eine Discovery im Rahmen von WSMX (Bussler et al. (2005)) oder WSMO-MX (Klusch und Kaufer (2009))) möglich ist.

5.1.2.5 BPMN-Editor-Komponente

Der BPMN Editor ist zum Einen zentraler Integrationspunkt für die Komponenten der vorherigen Abschnitte und zum Anderen eine Möglichkeit den Nutzer durch verschiedene Techniken bei der Durchführung einer GPM-getriebenen Service Discovery zu unterstützen. Obwohl die Services im Rahmen des Anwendungskonzepts auch für die Automatisierung andere prozessorientierte Modelle (wie z.B. EPK) eingesetzt werden können, wurde BPMN als mächtige und weit verbreitete Sprache gewählt.

Als Grundlage für die Umsetzung dient das Eclipse Open Source IDE⁸³, im Speziellen das BPMN-Unterprojekt⁸⁴ des SOA Tools Platform Project (STP)⁸⁵. Ein Bildschirmfoto der unmodifizierten STP BPMN Sicht in Eclipse ist in Abbildung 81 zu sehen. Zur Zeit der Implementierung war das STP BPMN Projekt das beste offene und am einfachsten modifizierbare BPMN Werkzeug. Zudem wird die Einbindung von zusätzlichen Funktionen in Java über neue Plugins sehr gut unterstützt.

Zunächst soll beschrieben werden, wie die einzelnen Komponenten in der Eclipse-Umgebung integriert sind. Anschließend werden die Modifikationen dargestellt, welche die Benutzerfreundlichkeit des Ansatzes auf Basis der Eclipse-Umgebung verbessert haben.

⁸³ www.eclipse.org

⁸⁴ <http://www.eclipse.org/bpmn/> (seit Januar 2012 archiviert)

⁸⁵ <http://eclipse.org/stp/>

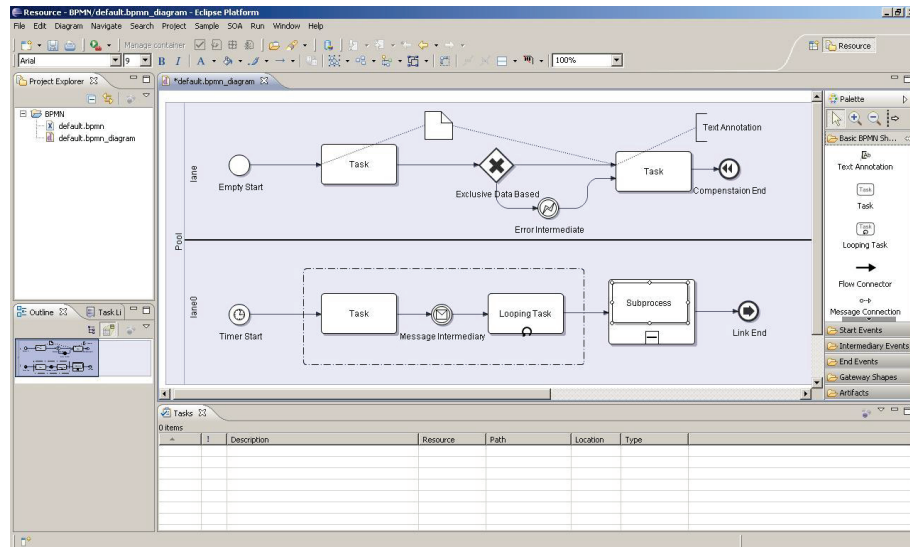


Abbildung 81: Unmodifizierte Oberfläche des BPMN Editors der SOA Tools Platform in Eclipse

Wie bereits mehrfach erwähnt ist der BPMN Editor der Integrationspunkt für die einzelnen Komponenten. Das **Service & Ontology Repository** wird über zwei verschiedene Ansichten eingebunden. **Services** können in einem Servicebrowser ausgewählt, angezeigt, hinzugefügt und entfernt werden. Da im aktuellen Anwendungsfall mit WSML Services gearbeitet wird wurden zu Demonstrationszwecken auch vordefinierte WSML Goals bereitgestellt. Aufgrund der Ähnlichkeit der Syntax von WSMO Services und Goals wurden das Repository und die Sicht im Editor um Goals erweitert. **Ontologien** werden über einen Ontology Navigation Plugin bereitgestellt. In diesem können Ontologien hinzugefügt und entfernt, allerdings nicht editiert werden. Die Darstellung mehrerer Ontologien wird durch die Nutzung einer Baumstruktur möglich, in der die einzelnen Elemente (Ontologien, Konzepte und deren Subkonzepte, Instanzen, Axiome, Attribute und Relation) expandierbar und reduzierbar sind. Dieser basiert auf dem Ontology Navigator des WSMO Studio Dimitrov et al. (2007b), welcher mit einer beispielhaften Ontologie in Abbildung 82 dargestellt ist.

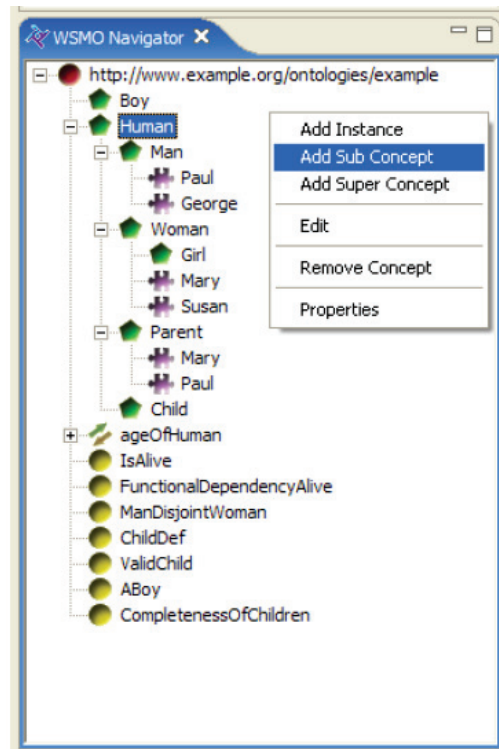


Abbildung 82: WSMO Navigator für WSMO Ontologien des WSMO Studio Werkzeug, nach Dimitrov et al. (2007a), S. 31.

Zur Initialisierung des **Service & Ontology Repository** existiert eine Ansicht für die Einstellungen des Prototyps. Hier können Endpunkt des Repository sowie weitere Ordner für die standardmäßige Suche nach Ontologien definiert werden. Das Werkzeug speichert diese Einstellungen in einer Initialisierungsdatei.

Reasoning und Alignment Komponenten haben keine direkte Integration innerhalb der Eclipse-Umgebung. Sie werden über die Discovery Komponente miteinander verbunden. Einstellungen für die Auswahl, welches Werkzeug für die Durchführung eines Alignment verwendet werden soll, sind in der Einstellungsansicht (siehe oben) vorhanden. In dieser kann auch manuell der Alignmentprozess gestartet werden. Dies ermöglicht die Messung der Zeit, welche für die Erstellung des Alignments nötig ist. Somit existiert eine einfache Form der Performanzmessung integriert in der Benutzeroberfläche.

Die engste Integration findet mit der **Discovery-Komponente** statt. Für den Beginn des Service Discovery Prozesses existieren drei Einstiegspunkte: (1) eine einfache Schlüsselwortsuche existiert als Plugin in einem Reiter des Service Browsers, (2) durch das Rechtsklicken auf eine Aktivität in der BPMN Modellierungssicht wird ein Dialog für die semantische Suche aufgerufen und (3) durch vorformulierte Goals im Service Brow-

ser kann eine semantische Suche ohne weitere Interaktion gestartet werden. Im Rahmen von (2) erscheint ein Dialog, welcher den Nutzer bei der Formulierung einer Service-suchanfrage unterstützen soll. Dieser bietet separate Bereiche für die Formulierung von Vorbedingung und Effekt (je Anfrage ist zurzeit nur eine Vorbedingung und ein Effekt möglich). Innerhalb der Bereiche wird der Nutzer aufgefordert Geschäftsobjekte und dazugehörige Status einzutragen, um somit die Funktionalität des gewünschten Service beziehungsweise der Serviceoperation zu beschreiben. Um relevante Geschäftsobjekte und Status zu finden, sind zwei Mechanismen implementiert. Zum einen die bereits beschriebene ontologische Autovervollständigung nach Sinkkila et al. (2008) (siehe Abschnitt 4.4.3) sobald der Nutzer beginnt, in die Felder für Geschäftsobjekt beziehungsweise Status etwas einzutragen. Zum anderen wird dem Nutzer nach dessen Anforderung ein Vorschlag für eine mögliche Kombination aus Geschäftsobjekt und Status präsentiert. Dieser basiert auf Annotationen innerhalb des Geschäftsprozessmodells, welche sich in Verbindung mit der zuvor ausgewählten Aktivität befinden. Solche Annotationen können bereits vorhanden oder zuvor vom Nutzer zu Verfeinerung des Modells hinzugefügt worden sein. Findet der Mechanismus keine Annotationen, so kann kein Vorschlag geliefert werden. Findet der Mechanismus allerdings unvollständige Annotationen (nur Vorbedingung/ nur Effekt; nur Geschäftsobjekt/ nur Status), so werden standardisierte Superkonzepte der jeweiligen Ontologie vorgeschlagen.

Wenn der Discovery Prozess durch einen der drei Einstiegspunkte angestoßen wurde, erfolgt die Verarbeitung durch die Discovery Komponente. Sobald eine Iteration über allen Serviceoperationen der Services des Service Repository durchgeführt wurde, wird eine Liste der Treffer an die Eclipse-Umgebung übergeben. Dem Nutzer werden die Ergebnisse in einer separaten Ergebnisansicht präsentiert. Neben Meta-Informationen über die Ergebnisse der Suche (durchsuchte Operationen, gefundene Treffer, Zeit für die Ausführung) erhält der Nutzer eine sortierte Liste nach vordefinierten Trefferkriterien (siehe Abschnitt 4.3.1). Zusätzlich kann der Nutzer mit der Ergebnisliste auf verschiedene Arten interagieren. Zum einen können die Ergebnisse nach Gesamtreffer, Vorbedingungstreffer und Effektreffer sortiert werden. Zum anderen werden durch Auswahl der einzelnen Ergebnisse weitere Beschreibungen zu Service und Serviceoperation präsentiert. Für eine bessere Übersicht können die Services in der Liste expandiert und reduziert werden, um deren Operation und die damit verbundenen Vorbedingungen und Effekte ein- beziehungsweise auszublenden.

Nachdem durch die bereits beschriebenen Implementierungen in der Eclipse-Umgebung die Grundfunktionalität für die Durchführung einer GPM-getriebenen heterogenen semantischen Service Discovery bereitgestellt ist, soll kurz auf weitere Funktionalitäten für eine einfachere Interaktion mit dem Nutzer eingegangen werden:

(1) **Annotation von BPMN Datenobjekten**

BPMN Datenobjekte in der Modellierungssicht der Umgebung können durch Konzepte der Ontologien einfach annotiert werden. Der Nutzer kann mittels Drag-and-Drop Konzepte aus dem Ontology Navigation Plugin mit den entsprechenden Datenobjekten verbinden. Dabei wird nachgefragt, ob ein Konzept als Geschäftsobjekt oder Status annotiert werden soll. Die annotierten

Konzepte stehen daraufhin für die spätere Service Discovery zur Verfügung.

(2) **Navigation innerhalb der ontologischen Autovervollständigung**

Die ontologische Autovervollständigung nach Sinkkila et al. (2008) wurde erweitert, so dass durch Nutzung der Maus oder Pfeiltasten innerhalb der Ontologie der vorgeschlagenen Konzepte navigiert werden kann. So können sowohl Super- als auch Subkonzepte ausgewählt werden. Zusätzlich wird dem Nutzer eine Beschreibung der Konzepte durch ein einfaches Mouse-Over angezeigt.

(3) **Weiterverwendung der Suchergebnisse**

In Abschnitt 4.4.4 wurde bereits das theoretische Konzept dieser Funktionalität beschrieben. Der Nutzer kann aus der Ergebnisansicht der Umgebung einzelne Serviceoperationen auswählen und mittels Drag-and-Drop mit den gewünschten BPMN Aktivitäten verbinden. Informationen über den Service (Name, Endpunktreferenz, Interface, etc.) werden BPMN 2.0 konform im Datenmodell des Diagramms hinterlegt. Hierfür kommt das BPMN Domain Model des Eclipse BPMN Modeler Projektes zum Einsatz. Dieses kann durch weitere Implementierungen in ausführbaren Code (zum Beispiel BPEL) umgewandelt werden. Eine solche Umwandlung ist jedoch nicht Teil der aktuellen Implementierung.

5.2 Modulares Ontology Matchmaking System

Neben der prototypischen Implementierung der heterogenen Service Discovery wurde ein weiteres Konzept zur Verbesserung der Qualität des Matchings erarbeitet. Dieses wurde zur Demonstration seiner Machbarkeit implementiert. Da die Umsetzung des Prototyps der GPM-getriebenen heterogenen Service Discovery im Rahmen eines Industrieprojektes realisiert wurde, konnten das modulare Matchmaking System aus lizenzrechtlichen Gründen nicht integriert werden. Aus diesem Grund wird die Implementierung des modularen Ontology Matchmaking Systems separat vorgestellt.

Im Weiteren wird die Architektur (inklusive Rollen) des Systems, die Implementierung der Komponenten sowie abschließend die grafische Benutzerinteraktion beschrieben.

5.2.1 Architektur

Zur Illustration der Architektur sind in Abbildung 83 der konzeptionelle Aufbau des Systems sowie mögliche Rollen für die Nutzung der Systemkomponenten dargestellt. Wie bereits in der Konzeption in Abschnitt 4.3.4 beschrieben müssen die Basisalgorithmen von ausgebildeten Programmierern geschrieben werden. Diese können natürlich auch durch Übernahme bestehender Algorithmen beziehungsweise Algorithmusimplementierungen entstehen. Sobald ein Basisalgorithmus zur Verfügung steht kann er zu den zugehörigen Repositories (Combiner, Modifier, Matcher) hinzugefügt werden. Die Aufgabe der Erweiterung der Repositories übernimmt im vorgeschlagenen System

die Strategieentwicklerin. Auf Basis ausgewählter Algorithmen der Repositories kann die Strategieentwicklerin zudem eine oder mehrere Matchingstrategien definieren. Ein Beispiel einer solchen Strategie durch die Verwendung von Präfixoperatoren ist in Abbildung 83 zu sehen. Eine oder eine Kombination von Strategien kann abschließend verwendet werden, um ein Alignment zwischen zwei Ontologien zu erstellen.

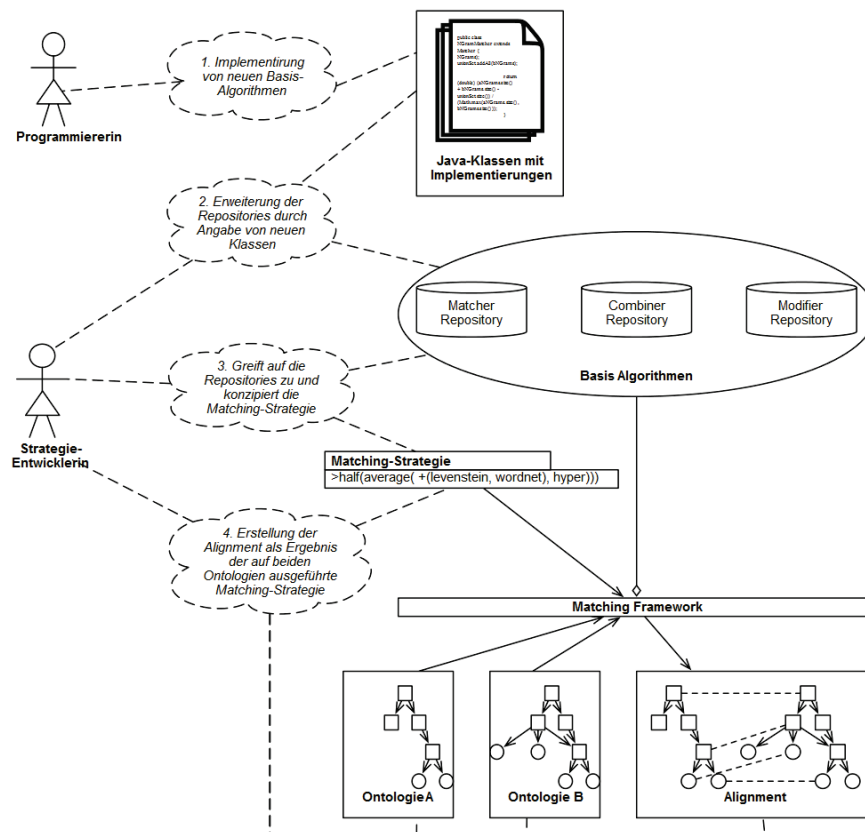


Abbildung 83: Architektur des modularen Ontology Matching Systems

Die hier beschriebenen vier Anwendungsfälle bilden die Basis für die Implementierung der benötigten Komponenten für das Ontology Matching Framework, welche im nächsten Abschnitt beschrieben werden.

5.2.2 Implementierung der Komponenten

Für die Umsetzung der Anwendungsfälle ist der Einsatz unterschiedlicher Technologien und Ansätze erforderlich. Dabei sollen sowohl bestehende Implementierungen wiederverwendet als auch neue Konzepte implementiert werden. Bereits existierende Ansätze die übernommen werden können sind die Beschreibungen der Ontologien nach

WSMO in der der WSM. Weiterhin werden Alignmentbeschreibungen nach den Definitionen der OAEI definiert. Um Ontologien verarbeiten zu können wird die wsmo4j API⁸⁶ eingesetzt. Da für die erste Implementierung des Matchingframeworks nicht alle Informationen einer WSMO Ontologie verwendet werden sollen, wird teilweise von dieser abstrahiert und eine sogenannte Simple Ontology API definiert (siehe Abbildung 84).

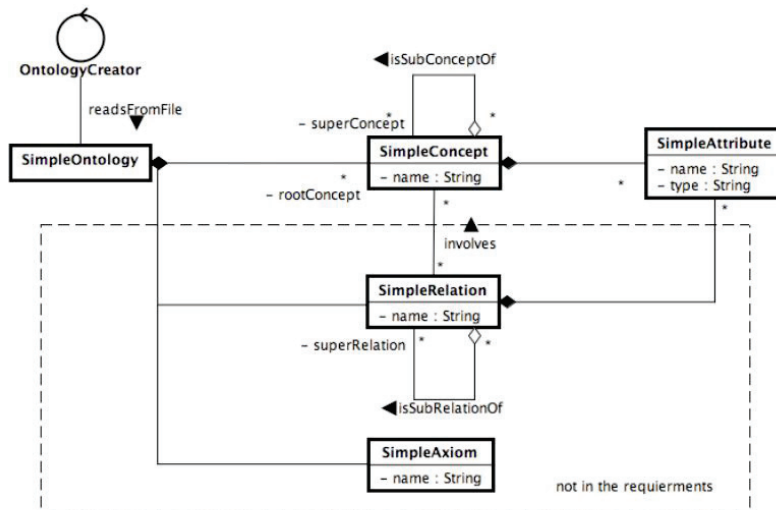


Abbildung 84: Bestandteile der Simple Ontology API

In der Abbildung ist zu erkennen, dass eine Simple Ontology aus Konzepten, Relationen und Axiomen besteht. Konzepte und Relation besitzen Super- und Subrelationen. Weiterhin können Konzepte bestimmte Attribute innehaben. In der aktuellen Implementierung liegt der Fokus allerdings auf dem Vergleich der Konzepte.

Matcher
+ match(a : SimpleArtifact, b : SimpleArtifact) : double
+ proceed(o1 : SimpleOntology, o2 : SimpleOntology) : SimilarityMatrix

MatrixCombiner
+ proceed(m1 : SimilarityMatrix, m2 : SimilarityMatrix) : SimilarityMatrix
+ function(a : double, b : double) : double

⁸⁶ <http://wsmo4j.sourceforge.net/>

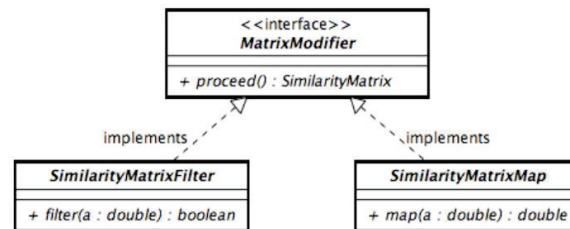


Abbildung 85: Klassen zur Definition von Basisalgorithmen

Sobald die Ontologien zur Verfügung stehen ist es nötig verschiedene Matchingstrategien für einen Vergleich definieren zu können. Dafür stehen die bereits erwähnten Basisalgorithmen zur Verfügung. In Abbildung 85 sind die abstrakten Klassen zur Implementierung dieser Algorithmen dargestellt. Durch die Standardisierung der Schnittstellen wird eine modulare Struktur ermöglicht, welche die einfache Definition von Matchingstrategien erlaubt. Für die Implementierung eines Matchers muss die Methode **match** implementiert und unter Umständen die Methode **proceed** überschrieben werden. Die Methode **proceed** ist schon in der abstrakten Klasse implementiert und führt bei Ausführung die **match** Methode auf allen Artefaktpaaren der Ontologien aus. Dadurch wird eine Ähnlichkeitsmatrix erstellt, welche in der Strategie weiterverwendet werden kann. Matcher erlauben Parametrisierungen, das heißt verschiedene Ausprägungen eines Matchingalgorithmus können durch unterschiedliche Parameter definiert werden (zum Beispiel das n eines n -gram Matchers).

Ähnlich aufgebaut ist die Combiner Klasse, sie implementiert die Methoden **function** und **proceed**. Die Methode **proceed** arbeitet mit zwei Ähnlichkeitsmatrizen als Eingangsparameter. Dabei führt sie für die gleich positionierten Wertepaare der Matrize die **function** Methode aus. Diese berechnet je nach ihrer Implementierung einen neuen Wert durch eine Kombination der alten Werte, zum Beispiel den Mittelwert dieser. Generell muss für einen Combiner nur **function** implementiert werden. Unter Umständen muss **proceed** überschrieben werden, wenn zum Beispiel die Ähnlichkeitsmatrizen unterschiedliche Strukturen aufweisen.

Zur Modifikation genau einer Matrix kommt ein Modifier zum Einsatz. Es sind drei Typen zu unterscheiden: *Common Modifier*, *Filter Modifier* und *Map Modifier*. Filter überprüfen jeden Wert einer Ähnlichkeitsmatrix auf eine Bedingung. Wird die Bedingung erfüllt bleibt der Wert bestehen, andernfalls wird er auf 0 gesetzt. Die Methode **filter** der abstrakten Klasse **SimilarityMatrixFilter** überprüft den Wert, die vererbte Methode **proceed** wendet den Filter auf alle Werte der Matrix an. Der Map Modifier hingegen wendet eine mathematische Funktion auf den Wert einer Matrix an, umgesetzt in der **map** Methode der abstrakten Klasse **SimilarityMatrixMap**. Wieder wird die vererbte Methode **proceed** verwendet, um diese Funktion auf alle Werte der Matrix anzuwenden. Der Common Modifier hingegen erlaubt die freie Modifikation der Matrix, so

können auch ganze Reihen oder Spalten oder die Matrix als Ganzes betrachtet werden. Dafür ist allerdings eine Überschreibung von `proceed` notwendig.

Durch die Existenz der beschriebenen Klassen können eine Vielzahl von Strategien definiert werden (gesetzt den Fall es existieren genug Implementierungen der Basisalgorithmen). Eine mögliche Strategie könnte somit wie folgt definiert werden:

$$threshold(average(ngram(o1;o2); levenstein(o1;o2)))$$

Auf die Ontologien *o1* und *o2* werden jeweils die Matcher *levenstein* und *ngram* angewendet. Anschließend werden die ermittelten Ergebnisse durch den *average*-Combiner verbunden, indem der Durchschnitt ermittelt wird. Abschließend wird der Filter Modifier *threshold* verwendet, welcher zum Beispiel alle Werte über 0,7 extrahiert. Auf diese Weise wird ersichtlich, dass relativ einfach eine Vielzahl von Matchingstrategien schnell und unkompliziert umgesetzt werden kann.

Die Umsetzung der Strategien wird nochmals durch den Einsatz einer grafischen Benutzeroberfläche vereinfacht, welche im nachfolgenden Abschnitt beschrieben ist.

5.2.3 Benutzerinteraktion

Durch den Einsatz des Eclipse Frameworks ist es möglich, einfache Strukturen zur benutzerfreundlichen Definition von Strategien zu schaffen. Die grafische Benutzeroberfläche lag nicht im Fokus der Implementierung, daher ist die Umsetzung rudimentär und sicherlich auch an vielen Stellen noch verbesserungsfähig. Nichtsdestotrotz ist es möglich, Strategien in dieser Umgebung ohne Programmierkenntnisse zu definieren. Ein Bildschirmfoto der Oberfläche ist in Abbildung 86 zu sehen.

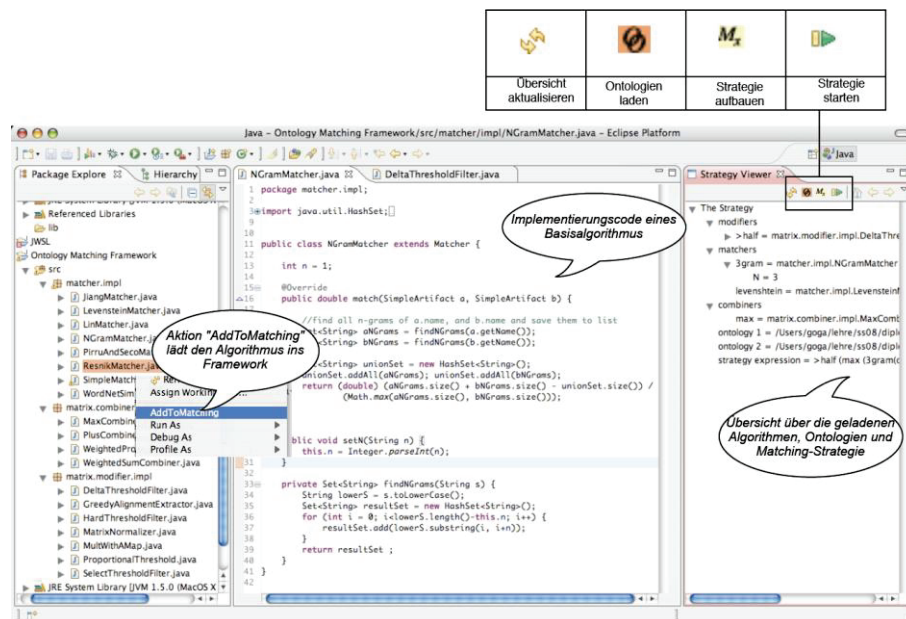


Abbildung 86: Benutzeroberfläche des modularen Ontology Matchmaking Systems

Für die Definition der Strategie wird auf die Implementierungen der Basisalgorithmen zurückgegriffen, welche im linken Teil des Bildes im Package Explorer zu sehen sind. Algorithmen können im Rahmen der Java-Eclipse Umgebung wenn gewünscht auch gleichzeitig angepasst werden. Zum Aufbau von Strategien kommt der Strategy Viewer im rechten Teil des Bildes zum Einsatz. Dieser gibt eine Übersicht über die geladenen Algorithmen, Ontologien und Strategien. Gleichzeitig können Strategien aufgebaut und verändert sowie Ontologien für ein Matching geladen werden. Sobald eine Strategie erstellt wurde, kann die Ausführung über den Menüpunkt „Strategie starten“ begonnen werden. Solange die Strategie valide ist und keine Fehler enthält wird nach der Ausführung ein Alignment der verwendeten Ontologien konform zum OAEI-Alignment-Format ausgegeben. Aus bereits genannten Gründen existiert keine Integration mit der prototypischen Implementierung der heterogenen GPM-getriebenen Service Discovery. Allerdings ist es möglich in der Zukunft das beschriebene Ontology Matchmaking System durch die Schnittstelle für Alignmentwerkzeuge anzubinden, und somit relativ einfach und schnell anwendungsfallabhängige Matchingstrategien implementieren und testen zu können.

6 Evaluierung

Zum Abschluss der vorliegenden Arbeit werden in den folgenden Abschnitten das Konzept geschäftsprozessorientierter Servicebeschreibung und die Implementierung einer heterogenen Service Discovery evaluiert. Letztere wird auf Basis einer quantitativen Auswertung mittels eines Testsets durchgeführt. Das Konzept der Beschreibungen hingegen wird vornehmlich durch eine qualitative Evaluierung im Rahmen zweier kurzer Fallstudien reflektiert. Zudem wird eine Abgrenzung zu bestehenden Ansätzen der heterogenen Service Discovery durchgeführt.

6.1 Quantitative Auswertung

Es erfolgt eine quantitative Auswertung der prototypischen Implementierung aus Abschnitt 5.1. Zunächst werden kurz die Metriken zur Bewertung der Güte der erzielten Ergebnisse und das verwendete Testset vorgestellt. Anschließend werden die Ergebnisse präsentiert und diskutiert. Die Ergebnisse wurden bereits in Rake et al. (2009) veröffentlicht.

6.1.1 Metriken und Testdaten

Für die Bewertung der Nutzung automatischer Alignments für eine heterogene Service Discovery werden unter Verwendung der prototypischen Implementierung zwei beispielhafte Experimente durchgeführt:

Experiment 1: Service Discovery ohne Alignment

Experiment 2: Service Discovery mit automatischem Alignment

Für die Messung der Ergebnisqualität der Experimente werden Standardmetriken aus dem Bereich des Information Retrieval verwendet: *Precision*, *Recall* und *F-Measure*.

- *Precision*: Anzahl der relevanten Operationen⁸⁷ in der Ergebnisliste geteilt durch die Anzahl der gefundenen Operationen in der Ergebnisliste
- *Recall*: Anzahl der relevanten Operationen in der Ergebnisliste geteilt durch die Anzahl aller relevanten Operationen in der Sammlung⁸⁸

⁸⁷ Die Evaluierung wird pro Operation durchgeführt, da jeder Services mehrere potentiell auch passende Operationen beinhalten kann. Relevant sind exakte sowie auch Subsume- Plugin-Treffer.

⁸⁸ Als Sammlung wird die gesamte Anzahl an Services in der Testumgebung bezeichnet.

- *F-Measure*: Kombination aus *Precision* und *Recall* durch die Verwendung eines gewichteten harmonischen Mittelwertes⁸⁹

In diesem Sinne kann neben korrekten Treffern differenziert werden in Falsch Positive (nicht korrekter Treffer), Falsch Negative (Fehlende Treffer) und, wenn auch weniger relevant, korrekte Nicht-Treffer.

Das automatische Alignment in Experiment 2a wird mit dem Werkzeug *Lily* (Wang und Xu (2009)) und in Experiment 2b mit dem Werkzeug *Falcon-OA* (Hu und Qu (2008)) erzeugt. Die Verwendung der erzeugten Alignments geschieht wie in Abschnitt 5.1.2.4 beschrieben.

Die größte Herausforderung bei der Evaluierung war eine fehlende Sammlung an Test-services. Obwohl Testsammlungen für andere Matchmaker existieren (wie die OWL-S Test Collection⁹⁰) stehen diese weder in der korrekten Sprache (WSMO) noch in der korrekten Form der Beschreibung (Anhand von Geschäftsobjekten und deren Status). Eine Konvertierung hat sich zudem nicht als sinnvoll herausgestellt. Stattdessen wurden die im bereits erwähnten Industrieprojekt entstandenen Services als Grundlage für den Aufbau einer Testsammlung verwendet. Genutzt wurden insgesamt 22 Services mit 41 Operationen von zwei Serviceprovidern (entsprechend zwei Domänenontologien). Auf Basis dieser Services und Ontologien wurden weitere 44 Services mit 82 Operationen auf Basis von vier Ontologien erstellt. Dabei wurden analog zu den Benchmarks der OAEI⁹¹ jeweils zwei Ontologien verfremdet. Das erste Set verwendet Synonyme der Konzepte (analog zu OAEI Benchmark Test 205) und das zweite Set extrahiert alle Vokale der Konzepte beider Ontologien (ähnlich zu OAEI Benchmark Test 201/202⁹²). Auf diesen Daten werden vier Testanfragen formuliert. Auf diese Art und Weise entstehen pro Experiment sechs verwertbare Metriken, insgesamt 18 Ergebniswerte. Die Auswertung der Experimente erfolgt im nächsten Abschnitt.

6.1.2 Ergebnisse

Die Ergebnisse der oben beschriebenen Experimente unter Verwendung der Werkzeuge *Falcon-OA* und *Lily* sind in Abbildung 87 beziehungsweise Abbildung 88 dargestellt. Die Evaluierung der Werte für Experiment 1 ohne Alignment ist verständlicherweise gleich. Hervorzuheben ist die *Precision* für das Experiment ohne Alignment. Diese liegt bei 1 (100%). Dies ist durch fehlende falsch positive Treffer zu erklären, da im Rahmen der Service Discovery nur nach Konzepten einer einzelnen Ontologie gesucht wird. Da

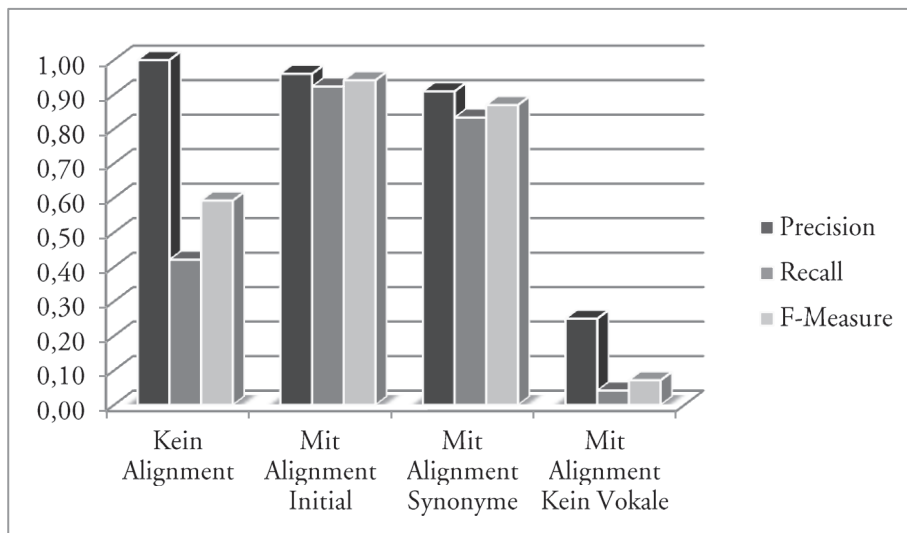
⁸⁹ Für die aktuellen Ergebnisse werden Werte von *Precision* und *Recall* gleich gewichtet,

⁹⁰ Aktuell in der Version 4.0, <http://projects.semwebcentral.org/projects/owls-tc/c>

⁹¹ <http://oaei.ontologymatching.org/2011/benchmarks/>

⁹² Ein Test komplett ohne Namen wurde aufgrund der geringeren Komplexität der Ontologien verworfen.

es aber keine Verfremdung durch ein Alignment gibt, werden genau alle Operationen gefunden, welche mit den Konzepten dieser einen Ontologie beschrieben wurden. Dies ist auch zu erkennen im *Recall*-Wert, welcher bei 0,42 liegt. Das heißt, dass vor einem Alignment nur 42% aller relevanten Operationen der Sammlung gefunden wurden. Dadurch ergibt sich ein kombiniertes *F-Measure* von 0,59.



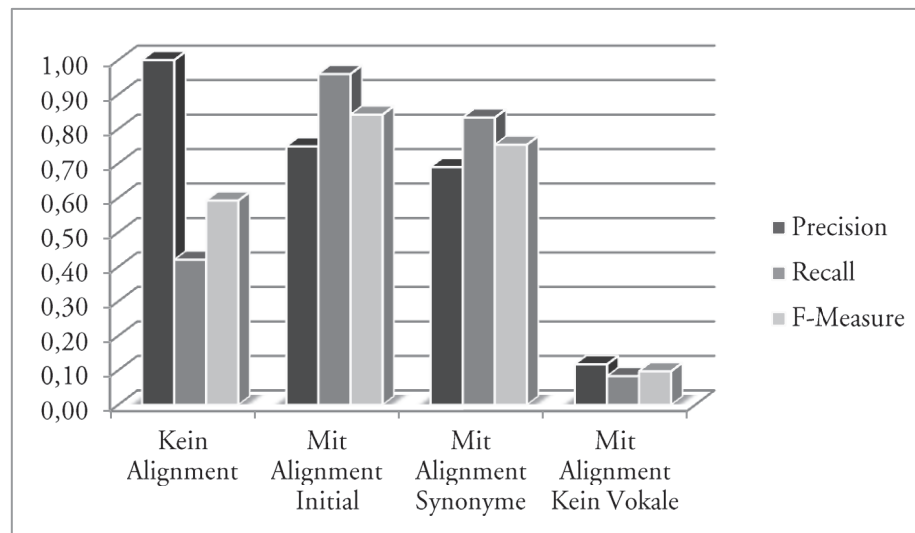
	Kein Alignment	Mit Alignment Initial	Mit Alignment Synonyme	Mit Alignment Kein Vokale
Precision	1,00	0,96	0,91	0,25
Recall	0,42	0,92	0,83	0,04
F-Measure	0,59	0,94	0,87	0,07

Abbildung 87: Auswertung der Service Discovery mit und ohne automatischem Alignment unter Verwendung von Falcon-OA

In Experiment 2 wird die Service Discovery mittels eines zuvor erstellten Alignments durchgeführt. Dies wird zunächst für die initialen Servicebeschreibungen und Operationen durchgeführt und anschließend für die Varianten mit zusätzlichen⁹³ Synonymen und Konzepten ohne Vokale. Falcon-OA (Abbildung 87) schneidet im initialen Szenario bedeutend besser ab als Lily, mit einem *F-Measure* von 0,94 zu 0,84. Ein entscheiden-

⁹³ Im initialen Szenario sind bereits aus dem Projekt bestimmte Konzepte unter den Ontologien synonym.

der Unterschied in diesem Szenario ist allerdings der *Recall*: Durch das Lily-Alignment werden 96% aller relevanten Operationen der Sammlung erkannt. Dieses Ergebnis wird jedoch getrübt durch eine Vielzahl an falsch positiven Treffern in der Ergebnisliste, erkennbar durch die *Precision* von 0,75 (Abbildung 88) im Vergleich zu 0,96 von Falcon-OA. Falcon-OA generiert also nur 4% falsch positive Treffer.



	Kein Alignment	Mit Alignment Initial	Mit Alignment Synonyme	Mit Alignment Kein Vokale
Precision	1,00	0,75	0,69	0,12
Recall	0,42	0,96	0,83	0,08
F-Measure	0,59	0,84	0,75	0,10

Abbildung 88: Auswertung der Service Discovery mit und ohne automatischem Alignment unter Verwendung von Lily

Im nächsten Szenario wird der Umgang der Alignment-Werkzeuge mit Synonymen getestet. In diesem Fall liefert Falcon-OA ein klar besseres (bezüglich *Precision* und *F-Measure*) beziehungsweise gleiches (bezüglich *Recall*) Ergebnis im Vergleich zu Lily. Mit 0,87 erzielt Falcon-AO ein deutlich besseres *F-Measure*-Ergebnis als Lily mit 0,75. Allerdings liefert Lily selbst mit Synonym-Verfremdung ein besseres Ergebnis als die Service Discovery ohne Alignment (*F-Measure* 0,59). Erwähnenswert in diesem Szenario ist zudem, dass Falcon-OA im Vergleich zum initialen Alignment ohne Verfremdung sowohl mehr falsch positive Treffer erzeugt (insgesamt 9%, *Precision* 0,9) als auch insgesamt weniger der möglichen Treffer erkennt (*Recall* 0,83). Lily erkennt prozentual ge-

nauso viele relevante Operationen, erzeugt im Gegensatz aber 31% falsch positive Treffer (*Precision* 0,69).

Im letzten Szenario scheitern beide Alignment-Werkzeuge. Durch die Extraktion der Vokale aus den Konzepten wird vor allem das strukturelle Matching getestet. Hierbei erweist sich immerhin Lily mit einem *F-Measure* von 0,10 ein wenig besser als Falcon-OA mit 0,07. Beide Ergebnisse sind für eine Verwendung in einem echten Szenario unakzeptabel. Durch eine weitergehende Analyse wurde festgestellt, dass dieses Ergebnis teilweise mit fehlenden Strukturinformationen innerhalb der gewählten Ontologien zu erklären ist. Die verwendeten Ontologien sind im Rahmen eines Industrieprojektes entstanden, erreichen hier jedoch nicht die Komplexität großer und ausgereifter Ontologien anderer Anwendungsfälle. Für eine weitere Auswertung eines solchen Szenarios werden weitere Anwendungsfälle und zusätzliche, zumindest strukturell komplexere Szenarien benötigt. Weiterhin ist zu erwähnen, dass dieses letzte Szenario ein Extremfall darstellt, welcher in der Realität in dieser Form wahrscheinlich nicht auftreten würde.

Die durchgeführte Evaluierung demonstriert zwei Aussagen:

1. Mit beiden Werkzeugen werden, unter Ausnahme eines Extremfalls, bessere Ergebnisse erzielt als bei einer Service Discovery ohne Alignment.
2. Für den bestehenden Anwendungsfall ist das Werkzeug Falcon-OA besser geeignet als Lily.

Eine Generalisierung dieser Aussagen ist nur in weiteren Experimenten mit zusätzlichen Daten möglich, welche im Rahmen dieser Arbeit leider nicht zur Verfügung stehen. Daher können die erzielten Ergebnisse nur ein Indiz für die Qualität einer heterogenen semantischen Service Discovery geben.

6.2 Qualitative Auswertung

In der folgenden qualitativen Auswertung des Ansatzes sowie der prototypischen Implementierung werden zum einen zwei Fallstudien zum Einsatz und der Reflektierung in der Praxis beschrieben. Zum anderen erfolgt eine kurze Abgrenzung zu ähnlichen bestehenden Ansätzen und Implementierungen.

6.2.1 Fallstudien

Der grundlegende Ansatz sowie die prototypische Implementierung wurden in einem Industrieprojekt mit einer weltweit führenden Unternehmenssoftware-Firma und einer internationalen Telekommunikations-Firma entwickelt und evaluiert. Durch ein weiteres Industrieprojekt mit einem internationalen Telekommunikationsanbieter wurden weitere Erkenntnisse für die Beschreibungen der Services gesammelt. In diesem zweiten Projekt wurden weder Ansatz noch Prototyp eingesetzt, da dies rechtlich nicht möglich war. Die gewonnen Erkenntnisse sind allerdings in die vorliegende Arbeit eingeflossen und haben somit den Ansatz erweitert.

6.2.1.1 Praxisprojekt Telekommunikation und Unternehmenssoftware

In diesem ersten Projekt kamen Ansatz und Prototyp zum Einsatz um die Software-Services der beiden Projektpartner im Rahmen mehrerer komplexer Geschäftsprozess-szenarios kombinieren zu können. Untersucht wurden drei Szenarien: (1) Management des Außendienstes im Rahmen eines Call Center Prozesses, (2) Überprüfung der Kreditwürdigkeit für eine Auftragserfüllung und (3) Ausführung einer Serviceanfrage auf der Fertigungsebene. In allen drei Szenarien konnte gezeigt werden, dass der Ansatz die verwendeten Services beschreiben und sinnvoll mit dem Prozess verbinden konnte. Zusätzlich war es möglich Services der beiden Anbieter zu kombinieren, um eine spätere Prozessautomatisierung vorzubereiten.

Unter Berücksichtigung der drei erwähnten Szenarien wurden unterschiedliche Nutzungsmuster und Anwendungsfälle identifiziert und implementiert. Der erste und wichtigste Anwendungsfall war die heterogene Service Discovery. Hier konnte gezeigt werden, dass durch Anfragen eines Anbieters auch die relevanten Services des anderen Anbieters gefunden wurden, ohne den Einsatz eines manuellen Matchings zwischen den Anbieterontologien. Teil dieses Anwendungsfalls waren die Umsetzung der Service Discovery rein auf Basis von entweder Vorbedingung oder Effekt. In beiden Fällen war es möglich eine Anfrage auf Basis bestehender semantischer Beschreibungen des Geschäftsprozesses oder durch die Nutzung der jeweiligen Ontologie zu formulieren. Für die Nutzung von Ontologiekonzepten wurden weitere Anwendungsfälle entwickelt. Zum einen die Auswahl und Verwendung von Konzepten über einen Ontologie-Navigator. Zum anderen die Suche nach Konzepten unter Nutzung einer Autovervollständigung und anschließenden integrierten Navigation zu den relevanten Konzepten. Weiterhin wurde es ermöglicht die Geschäftsprozesse der Szenarien unter Verwendung des Ontologie-Navigators semantisch zu annotieren, unter anderem um damit die verschiedenen Varianten der Service Discovery zu realisieren.

Neben den Anwendungsfällen kamen unterschiedliche Nutzungsmuster zum Tragen. Diese sind auch über die Szenarien hinweg relevant für die Verwendung der GPM-getriebenen Service Discovery und Automatisierung der identifizierten Prozesse. Insgesamt wurden fünf verschiedene Nutzungsmuster betrachtet: (1) Aufbau von Telekonferenzen unter Verwendung von Präsenz, (2) Weiterleitung von Anrufen an verschiedene Service-Level, (3) Aktualisierung von Präsenzinformationen, (4) Manipulation und Erstellung von Geschäftsobjekten und (5) Kommunikationsunterstützende Geschäftsobjekte. Obwohl diese Nutzungsmuster nicht immer direkt in den Ansatz einfließen konnten, haben sie wichtige Erkenntnisse zur Entwicklung des Beschreibungsansatzes und der Implementierung des Prototyps geliefert.

Eines der wichtigsten Ergebnisse des Projektes, neben der technischen Machbarkeit der Implementierung, war die Darstellung und Beschreibung realer Services der zwei Anbieter mittels des entwickelten GPM-zentrierten Ansatzes. Somit konnte eine Praxisnähe realisiert werden, welche sich auch für die Auswertung im zweiten Projekt als hilfreich erwiesen hat. Weitere Details zu dieser Fallstudie können aus Gründen der Vertraulichkeit gegenüber den Projektpartnern nicht geliefert werden.

6.2.1.2 Praxisprojekt Telekommunikationsanbieter

Das zweite Praxisprojekt, welches relevante Erkenntnisse für die vorliegende Arbeit lieferte, wurde bei einem internationalen Telekommunikationsanbieter durchgeführt. Im Projekt wurden unter Anderem verschiedene Aspekte der Governance einer serviceorientierten Architektur untersucht. Im Speziellen wurden auch Servicebeschreibungen analysiert und bewertet. Diese Bewertung führte zu Rückschlüssen für die geschäftsprozessorientierten Servicebeschreibungen des vorliegenden Ansatzes. Anonymisiert werden im Folgenden einige Erkenntnisse des Projekts in diesem Zusammenhang dargestellt.

Für die Beschreibung der Services ist vor allem die Einordnung in ein Domänenmodell relevant, realisiert durch eine sogenannte fachlogische Architektur. Die Domänen innerhalb dieser Architektur bündeln logische Funktionen. So entsteht ein Modell welches bestehende Applikationen kartographiert und so auch die Evolution hin zu einer serviceorientierten Architektur unterstützt, indem verschiedene Funktionen der Altsysteme genauso wie neue Services einzelnen Domänen zugeordnet werden (dargestellt in Abbildung 89).

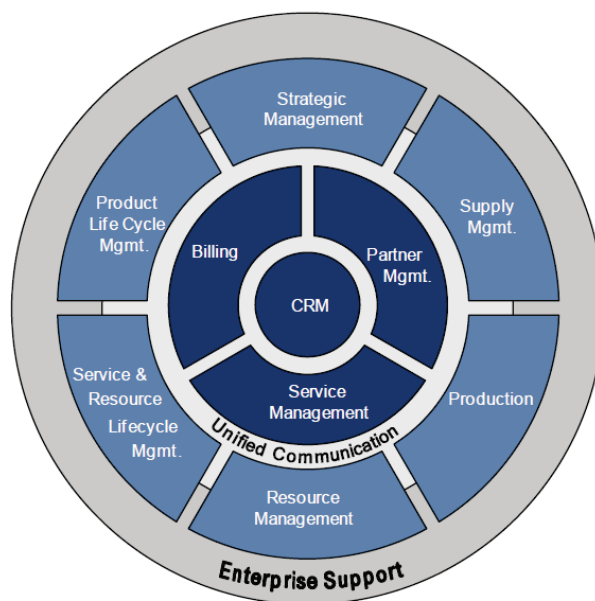


Abbildung 89: Domänenmodell des Telekommunikationsanbieters

Eine direkte Zuordnung der Applikationen zu den Domänen und funktionalen Komponenten ist möglich. Wichtig ist vor Allem, dass den Domänen aggregierte Geschäftsobjekte zugeordnet werden können. Diese bestehen aus relevanten Geschäftsobjekten des Telekommunikationsanbieters. Über die Modellierung dieser Geschäftsobjekte ist auch eine Verbindung mit Services möglich, welche diese manipulieren. Durch die Modellierung der Geschäftsobjekte soll zudem eine einheitliche Semantik zur Kommu-

nikation von IT- und Fachseite entstehen. Hierzu trägt weiterhin die eindeutige Zuordnung der Geschäftsobjekte zu einzelnen Bereichen des Domänenmodells bei, wodurch (vor allem fachliche) Verantwortlichkeiten festgehalten werden können.

Vor und während der Projektlaufzeit wurden unterschiedliche Pilotprojekte zur Erstellung und Einsatz von Services im Sinne einer serviceorientierten Architektur durchgeführt. Für die Zukunft wurde eine einheitlich Governance einer SOA entwickelt. Für den in dieser Arbeit beschriebenen Ansatz konnten in diesem Kontext wichtige Erkenntnisse gesammelt werden:

- Die Verwendung eines unternehmens- oder domänenweiten Geschäftsobjektmodells ist möglich und kann als Grundlage für Ontologien verwendet werden.
- Eine Kombination aus Geschäftsobjekt(en) und Verben zur Manipulation wurde in der SOA Governance als Grundlage für die einheitliche Benennung von Services verwendet. Dies ist der erste Schritt zu einer formalen funktionalen Beschreibung durch Tupel aus Geschäftsobjekten und Status, wie im vorliegenden Ansatz beschrieben.
- Die Geschäftsobjekte zur Beschreibung der Services (und indirekt ihrer Funktionalität) stammen aus dem erwähnten einheitlichen Modell. Zusätzlich sieht die SOA Governance eine Begrenzung der Verben zur Benennung der Services vor. Dies unterstützt den Vorschlag des vorliegenden Ansatzes zur Verwendung kontrollierter Vokabulare für Geschäftsobjekte und Status.
- Für die zukünftige Weiterentwicklung des Ansatzes sind neben den funktionalen Beschreibungen auch nichtfunktionale Eigenschaften nötig, wie sich in diesem Projekt gezeigt hat. Vor allem eine Einordnung in ein formales (zum Beispiel auf einer Ontologie basierenden) Lebenszyklusmodells und somit der Verfeinerung der Suchparameter erscheint sinnvoll. Die Ausarbeitung und Implementierung nichtfunktionaler Eigenschaften übersteigen allerdings den Umfang dieser Dissertation.

6.2.2 Vergleich mit bestehenden Ansätzen

Im vorliegenden Abschnitt wird eine kurze Abgrenzung des vorgestellten Ansatzes für eine heterogene semantische Service Discovery im Bereich der Geschäftsprozessmodellierung zu Ansätzen aus Abschnitt 3.4 durchgeführt.

Zunächst sei erwähnt, dass dem Autor eine Kombination heterogener semantischer Service Discovery im speziellen Bereich der Geschäftsprozessmodellierung durch andere Ansätze nicht bekannt ist. Daher wird eine Abgrenzung nur für andere Ansätze der heterogenen semantischen Service Discovery durchgeführt.

Der Ansatz von Usanavasin et al. (2005) (siehe Abschnitt 3.4.1) liefert gute Konzepte für die Nutzung unterschiedlicher Ontologien für die Servicebeschreibung. Im Vergleich zum vorliegenden Ansatz werden allerdings nur Eingabe und Ausgabeparameter mit den Ontologiekonzepten beschrieben, nicht komplexe logische Ausdrücke zur

Formulierung von Vorbedingungen und Effekten. Für das Matching der Ontologien wird ein eigener Algorithmus entwickelt, dessen Performanz auch evaluiert wird. Leider beschreiben die Autoren weder die eigentliche Einbindung des Matchings in die Service Discovery noch die Implementierung eines Prototypen. Die Evaluierung nach Einbindung des Ontology Alignment ist ein wichtiger Faktor für die Bewertung eines solchen Ansatzes.

Einen dem vorliegenden Ansatz in der Architektur sehr ähnlichen Ansatz beschreiben Kim und Lee (2010) (siehe Abschnitt 3.4.2). In ihrem Ansatz wird vor allem von einer sehr großen Anzahl unterschiedlicher Ontologien ausgegangen, wodurch auch eine sehr große Anzahl an Alignments vorhanden sein oder generiert werden muss. Hier ist anzumerken, dass Kim und Lee (2010) weder beschreiben, wie die verwendeten Alignments entstehen (automatisch oder manuell), noch wer für die Erstellung und eine eventuelle Überprüfung der Korrektheit verantwortlich ist. Zudem werden keine Vor- und Nachbedingungen oder komplexe semantische Beschreibungen, sondern „nur“ einfache Ontologiekonzepte miteinander verglichen. Von Bedeutung ist aber vor allem die Erzielung einer hohen Performanz im Umgang mit einer Vielzahl von Ontologien, deren Erkenntnisse auch für die zukünftige Entwicklung des hier vorgestellten Ansatzes relevant sein könnten.

Ein relativ statischer aber auch praktikabler Ansatz zum Umgang mit Heterogenität bei der Service Discovery ist der Einsatz von Mediatoren, welche bereits in der WSMO Spezifikation vorgesehen sind (siehe Abschnitt 3.4.3). Della Valle et al. (2005) implementierten einen Ansatz zur Service Discovery unter Nutzung von Mediatoren in Form des *Glue*-Systems. Wie bereits beschrieben benötigt der erfolgreiche Einsatz dieses Systems allerdings die manuelle Erstellung und Überprüfung von Mediatoren zwischen allen eingesetzten Ontologien. Nach einer erfolgten Erstellung agiert die Service Discovery mit ähnlicher Qualität wie in einer homogenen Umgebung. Der vorliegende Ansatz dagegen opfert einen Teil der Ergebnisqualität, um den entstehenden Aufwand der manuellen Erstellung von Mediatoren drastisch zu reduzieren.

Abschließend präsentieren Oundhakar et al. (2005) einen sehr umfangreichen Ansatz, um Konzepte heterogener Ontologien für die Service Discovery einsetzen zu können. Wie auch andere Ansätze vor ihnen betrachten die Autoren nur das Matching von Ein- und Ausgabeparametern und nicht wie im vorliegenden Ansatz logische Ausdrücke von Vorbedingungen und Effekten. Weiterhin fließen nur begrenzt strukturelle Informationen der Ontologien in das Matching der Konzepte ein. Ein Umstand der durch den Einsatz unterschiedlicher Matching-Werkzeuge im vorliegenden Ansatz behoben werden konnte.

7 Zusammenfassung und Ausblick

In diesem abschließenden Kapitel der Arbeit soll zunächst eine Zusammenfassung der einzelnen Kapitel präsentiert werden. Diese hat zum Ziel, die erarbeiteten Ergebnisse zu rekapitulieren und aufbauend darauf einen Ausblick auf die Weiterentwicklung des Ansatzes und der Implementierung zu geben.

7.1 Zusammenfassung

In der vorliegenden Arbeit wurde ein Ansatz für die geschäftsprozessorientierte Service Discovery auf Basis heterogener semantischer Beschreibungen erarbeitet und im Rahmen einer prototypischen Implementierung deren Machbarkeit demonstriert sowie evaluiert.

Die Motivation der Notwendigkeit eines solchen Ansatzes ergibt sich aus der generellen Problematik des sogenannten *Business-IT-Gaps*. In Unternehmen existieren oft Probleme zwischen der fachlichen Spezifikation sowie der gewünschten Implementierung von Prozessen und der Umsetzung durch die IT. Aktuelle monolithische Architekturen liefern nicht die von Fachabteilungen gewünschte Flexibilität und somit kann das Unternehmen nicht schnell genug auf wechselnde Bedingungen seiner Umwelt reagieren. Dies kann im schlimmsten Fall zu Wettbewerbsnachteilen führen. Das Aufkommen serviceorientierter Architekturen als neues Paradigma verspricht eine höhere Flexibilität durch feingranulare, austauschbare und wiederverwendbare Services, welche die Geschäftsprozesse eines Unternehmens direkt unterstützen. Weitere potentielle Vorteile dieser neuen Architektur sind eine Verringerung von Implementierungskosten und eine Beschleunigung der Umsetzung durch erhöhte Wiederverwendung der Software-Services. Diese Vorteile führen jedoch gleichzeitig zu einer Erhöhung der Komplexität in der Verwaltung beziehungsweise *Governance* der SOA. Ein praktischer Teil dieser ist die Suche nach Services bestimmter Funktionalität, welche anschließend miteinander kombiniert werden können, um komplexe Geschäftsprozesse zu unterstützen. Weiterhin können die verwendbaren Services von einer Vielzahl unterschiedlicher Anbieter stammen. Semantische Servicebeschreibungen helfen zunächst die Funktionalitäten von Services mit einer erweiterten Logik wie Vorbedingungen und Effekten zu spezifizieren. Dabei ist allerdings nicht davon auszugehen, dass alle Anbieter in Zukunft die gleiche Wissensbasis für ihre Beschreibungen verwenden werden. Das Ziel dieser Arbeit war somit die semantische Suche nach Services unter der besonderen Problematik heterogener Beschreibungen zu betrachten. Aus dem Kontext einer SOA wurde zudem die Geschäftsprozessorientierung als wichtiger Treiber für die Service Discovery selektiert, wodurch weitere Anforderungen entstanden sind.

Kapitel 1 detailliert die bereits geschilderte Motivation und beschreibt konkrete Anwendungsfälle. Folgendes Ziel für die Dissertation wird formuliert:

„... [E]inen Ansatz zu entwickeln, welcher es ermöglicht fachlich relevante Services in einer SOA effektiv und effizient aufzufinden. Weiterhin gilt die zusätzliche Bedingung, dass diese Services ihre Funktionalität in Form von heterogenen semantischen Beschreibungen bekanntmachen können.“ (siehe Abschnitt 1.1.2)

Auf Grundlage dieses Ziels wird eine Abgrenzung durchgeführt. Weiterhin wird die Forschungsmethodik auf Basis des Information Systems Research Framework nach Hevner et al. (2004) vorgestellt und dargelegt, dass die Dissertation allen definierten Richtlinien folgt. Zudem werden durch die Arbeit die folgenden Beiträge für Wissenschaft und Praxis geliefert: (1) Modell zur geschäftsprozessorientierten Servicebeschreibung, (2) Konzept einer heterogenen Service Discovery, (3) Machbarkeitsstudie, (4) Fallstudien, (5) Testset und quantitative Auswertung.

Kapitel 2 definiert die grundlegenden Begriffe und Theorien für das Verständnis der Arbeit. Im Zentrum stehen dabei Service- und Geschäftsprozessorientierung. Unter dem Aspekt der Unternehmensarchitektur wird der Begriff serviceorientierte Architektur nach Offermann et al. (2007) in einer ganzheitlichen Sicht vorgestellt. Neun Grundprinzipien bilden die Basis für Zielprinzipien einer SOA: (1) Wiederverwendbarkeit, (2) Komponierbarkeit, (3) Integration und (4) Interoperabilität. Anschließend erfolgt eine Beschreibung des Semantic Webs und der Semantic Web Services, welche durch erweiterte semantische Beschreibungen auf Basis von Ontologien die Bedeutung (Semantik) eines Service definieren und nicht nur die Syntax. Unter Annahme der bereits erwähnten Heterogenität der Ontologien werden Begriffe und Theorien des Ontology Matchings und Alignments erläutert, um Möglichkeiten darzulegen diese Heterogenität zu überbrücken. Das Kapitel schließt mit grundlegenden Definitionen der Geschäftsprozessorientierung und des Geschäftsprozessmanagements sowie der Modellierung von Prozessen.

Kapitel 3 diskutiert den aktuellen Stand der Forschung in den für die Arbeit relevanten Forschungsbereichen. Auf Basis einer Unterscheidung dreier Ebenen – Schlüsselwort, „leichtgewichtig“ und „schwergewichtig“ – werden bestehende Ansätze der semantischen Service Discovery vorgestellt. Im Fokus stehen dabei die drei vorherrschenden Ansätze OWL-S, SAWSDL und WSMO. Anschließend werden Konzepte und Implementierungen zur Modellierung, Implementierung, Ausführung und Analyse semantischer Geschäftsprozesse untersucht, welche den Bereich des semantischen Geschäftsprozessmanagements umfassen. Die wichtigsten Techniken, Strategien und Werkzeuge im Bereich des Ontology Matchings werden in Abschnitt 3.3 diskutiert. Im Fokus der Betrachtung der Werkzeuge stehen Implementierungen, welche im Rahmen der Evaluierung durch die Ontology Alignment Evaluation Initiative herausragende Ergebnisse erzielt haben. Der finale Abschnitt des Kapitels 3 widmet sich bestehenden Ansätzen der heterogenen (semantischen) Service Discovery. Die darin beschriebenen Konzepte bilden zum einen die Grundlage für Anforderungen an den entwickelten Ansatz und zum anderen das Maß für die Abgrenzung beziehungsweise Evaluierung in Kapitel 6.

Kapitel 4 beschreibt den entwickelten Ansatz der geschäftsprozessorientierten Service Discovery. Zunächst wird auf Basis einer Identifizierung von Zielgruppen und einer Klassifizierung von Services abgeleitet, welche Konzepte zur Beschreibung der Funktionalität eines Service dienen können. Es wird dargelegt wie Geschäftsobjekte und Status dieser im Rahmen von Vorbedingungen und Effekten eingesetzt werden, um die Funktionalität zu beschreiben. Weiterhin wird erläutert, wie bestehende formaler Beschreibungen (WSMO, OWL-S und SAWSDL) zur Umsetzung des entwickelten Ansatzes eingesetzt werden können. Auf Basis der entwickelten geschäftsprozessorientierten Servicebeschreibungen ist eine Service Discovery aus dem Kontext der Geschäftsprozessmodellierung möglich. Diese GPM-getriebene Service Discovery kann zum einen Kontextinformationen der Prozessmodelle nutzen und zum anderen direkt in die Modellierung(swerkzeuge) integriert werden. Eines der Alleinstellungsmerkmale des Ansatzes ist die in Abschnitt 4.3 vorgestellte Möglichkeit einer heterogenen Service Discovery, das heißt die Suche nach Services, welche mit Konzepten unterschiedlicher Ontologien beschrieben sind. Dabei wird auf Basis eines grundlegenden Algorithmus semantischer Service Discovery ein Modell zur Integration von Ontology Matching Werkzeugen entwickelt. Es kommt eine Modularisierung des Matchings zum Einsatz, welche es ermöglicht, die jeweils passenden Werkzeuge für unterschiedliche Anwendungsfälle integrieren zu können. Anschließend werden Konzepte für eine verbesserte Benutzerinteraktion innerhalb des Ansatzes diskutiert, realisiert durch kontrolliertes Vokabular, kontextorientierte Annotationen, Autovervollständigung sowie Weiterverwendung.

Kapitel 5 stellt die auf dem Ansatz entstandene prototypische Implementierung vor. Diese basiert auf einer fünf-Komponenten Architektur bestehend aus Service & Ontologie Repository, Reasoning-Komponente, Alignment-Komponente, Service-Discovery-Komponente und BPMN-Editor-Komponente. Beschrieben werden die Funktionalität und die Integration der Komponenten zur Umsetzung einer geschäftsprozessorientierten semantischen Service Discovery auf Basis heterogener Beschreibungen. Neben der Umsetzung des Ansatzes wird ein System zur Verbesserung des Matchmakings bei der Überwindung von Heterogenität präsentiert. Diese Implementierung ist separat, aber unter der Berücksichtigung der Anforderungen des vorgestellten Ansatzes entstanden, da eine Integration im Rahmen der Forschungsprojekte nicht möglich war. Architektur, Implementierung und Benutzerinteraktion des entstandenen modularen Ontology Matchmaking Systems werden in Abschnitt 5.2 detailliert.

Kapitel 6 führt eine quantitative und qualitative Evaluierung des Ansatzes und der prototypischen Implementierung durch. Die quantitative Auswertung auf Basis der Metriken *Precision*, *Recall* und *F-Measure* testet eine Discovery ohne sowie mit Alignment realisiert durch die Werkzeuge *Lily* und *Falcon-AO*. Dabei zeigt sich, dass sich außer im Rahmen eines Tests eine Service Discovery mit Alignment immer positiv auf die Ergebnisqualität auswirkt. Weiterhin schneidet *Falcon-AO* im getesteten Anwendungsfall deutlich besser ab als der Konkurrent *Lily*. Ergänzt wird die quantitative Auswertung um eine qualitative Analyse in Form von Fallstudien und der Abgrenzung des Ansatzes zu bestehenden Implementierungen. Dabei kann gezeigt werden, dass die gewählte Beschreibungslogik zur Abbildung realer Services unterschiedlicher Anbieter durchaus ge-

eignet ist. Weiterhin existiert nur ein Ansatz der heterogenen semantischen Service Discovery, welcher die gleiche Tiefe und Automatisierung erreicht und welcher auch von den Autoren evaluiert wurde. Dieser bewegt sich allerdings nicht im Kontext der Geschäftsprozessorientierung, wodurch eine ausreichende Abgrenzung entsteht.

7.2 Ausblick

Ein Ausblick auf mögliche zukünftige Entwicklungen des vorgestellten Ansatzes lässt sich am besten nach einer Diskussion der Limitationen führen. Anschließend können weitere, über die Limitation hinausgehende Verbesserungen und Forschungsgebiete identifiziert werden.

Der vorliegende Ansatz kann in der Theorie mit unterschiedlichen Beschreibungssprachen für Semantic Web Services umgesetzt werden. Beispiele wurden neben WSMO als Basis für die Implementierung auch für OWL-S und SAWSDL gegeben. Obwohl gleiche oder ähnliche Beschreibungen der Services in anderen Beschreibungssprachen als WSMO möglich sind, kann die prototypische Implementierung nur mit dieser umgehen. In der Zukunft ist es also wünschenswert, dass neben WSMO auch OWL-S und SAWSDL eingesetzt werden können.

Die Service Discovery basiert zurzeit auf Vorbedingungen und Effekten, welche mit Geschäftsobjekten und deren Status beschrieben werden. Die Ein- und Ausgabeparameter können mit den Geschäftsobjekten in Verbindung stehen, dies ist jedoch nicht immer gegeben. Eine Erweiterung der Service Discovery um Ein- und Ausgabeparameter in der zukünftigen Entwicklung ist denkbar. Vor allem, wenn diese auch mit Ontologiekonzepten beschrieben werden und so ein Matching zwischen heterogenen Beschreibungen hergestellt werden kann. Alternativ können bestehende syntaktische und semantische Matcher auf Datenmodellebene eingesetzt werden, wie zum Beispiel durch Elhadad et al. (2008) beschrieben.

Der Ansatz sowie die prototypische Implementierung konnte anhand zweier Praxisprojekte qualitativ evaluiert werden. Allerdings standen in den Projekten nur eine begrenzte Anzahl an Szenarien und Prozessen zur Verfügung. Für die Zukunft ist eine Evaluierung von Prozessen und Services unterschiedlicher Domänen notwendig, um eine domänenunabhängige Aussage zur Effektivität und Effizienz des Ansatzes treffen zu können. Eine solche Evaluierung ist sehr einfach möglich, da weder die Implementierung noch die verwendete Form der Servicebeschreibung domänenspezifisch sind.

Die quantitative Auswertung wurde anhand realer Services und deren Beschreibung durch die entwickelten Vorbedingungen und Effekte durchgeführt. Die verwendete Testsammlung der Services muss in der Zukunft erweitert werden. Dies kann anhand mehrerer Dimensionen geschehen. Bereits erwähnt wurde die Erweiterung um zusätzliche Domänen und Einsatzszenarien. Weiterhin kann die Komplexität einzelner Servicebeschreibungen durch mehrere Vorbedingungen und Effekte sowie die Komplexität der logischen Ausdrücke erhöht werden. Dies würde weitere Aussagen sowohl über die

Qualität des Service Discovery Algorithmus als auch des Ontology Matching Algorithmus erlauben. Ein direkter Vergleich mit anderen Ansätzen ist zurzeit nicht möglich. In der Zukunft entstehen unter Umständen alternative Ansätze mit einem ähnlichen Fokus, welches eine solche Vergleichbarkeit ermöglichen könnte.

Die Implementierung bietet an bestimmten Stellen Raum für Verbesserungen. Zurzeit kann die Alignment-Komponente nur mit zwei Ontologien gleichzeitig umgehen, was die für die Demonstration der Machbarkeit einer heterogenen Service Discovery ausreicht. Der entwickelte Ansatz sieht aber bereits vor, eine Vielzahl von Ontologien zu vergleichen, wofür eine Erweiterung der Alignment-Komponente notwendig wird. Weiterhin muss in Zukunft die Interaktion mit dem Benutzer in der Eclipse-Umgebung die bereits erwähnte Möglichkeit bieten, mehr als eine Vorbedingung beziehungsweise einen Effekt für die Formulierung der Serviceanfrage zu unterstützen. Dies kommt auch bei der Einbeziehung von Vorbedingungen und Effekten zum Tragen, welche nicht direkt mit der betrachteten Aktivität für die Service Discovery verbunden sind. Es sollte möglich sein, eine Vielzahl von Vorbedingungen im Prozess vor der ausgewählten Aktivität sowie eine Vielzahl von Effekten nach der ausgewählten Aktivität dem Nutzer zur Formulierung einer Serviceanfrage vorschlagen zu können. Nach einer erfolgreichen Serviceanfrage werden dem Nutzer die Ergebnisse in Form einer sortierbaren Liste präsentiert. Für zukünftige Versionen ist es wünschenswert, eine einfache Verfeinerung der Suchanfrage bereits in der Benutzerinteraktion vorzusehen. Im Moment müssen Nutzer manuell eine neue Anfrage starten.

Weitere Forschung zur Verbesserung des Gesamtansatzes sieht der Autor vor allem im Bereich der semantischen Geschäftsprozessmodellierung, des Einsatzes von Machine-Learning Techniken zur Erhöhung der Matchingqualität unter der Einbeziehung der Nutzer sowie der Umsetzung der Prozesse in ausführbare Sprachen. Letzteres ist ein Aspekt, welcher im Moment zwar möglich, aber stark an der Qualität des automatischen Mappings der Eingangs- und Ausgangsdaten abhängig ist, wobei bestehende Ansätze noch keine ausreichende Performanz für die Vollautomatisierung leisten.

Abbildungsverzeichnis

Abbildung 1: Heterogenität von Service Beschreibungen unterschiedlicher Service-Anbieter	3
Abbildung 2: Rahmenwerk des Information Systems Research (aus dem Englischen, nach Hevner et al. (2004), S. 80).....	5
Abbildung 3: Bestandteile der Unternehmensarchitektur nach Aier (2007), S. 135....	11
Abbildung 4: SL&I-Modell einer ganzheitlichen SOA nach Offermann et al. (2007)	14
Abbildung 5: Mögliche Komponenten einer serviceorientierten Architektur, nach Schröpfer (2009), S. 27.	21
Abbildung 6: Semiotisches Dreieck	23
Abbildung 7: Formen von Ontologien nach Euzenat und Shvaiko (2007) in Anlehnung an Uschold und Gruninger (2004).....	25
Abbildung 8: Übersicht aktueller Semantic Web Sprachen, nach Grimm et al. (2007), S. 80.....	27
Abbildung 9: Oberste Ebene der Service Ontologie von OWL-S, nach Martin et al. (2007), S. 247.....	31
Abbildung 10: Kernelemente von WSMO, nach Roman et al. (2005), S. 80.....	32
Abbildung 11: SAWSDL Konzept nach Kopecký et al. (2007), S. 61.....	34
Abbildung 12: Der Matching-Prozess nach Euzenat und Shvaiko (2007), S. 44.....	36
Abbildung 13: Klassifizierung von Matching-Techniken nach Euzenat und Shvaiko (2007), S. 65.....	37
Abbildung 14: Beispiel für das Alignment zweier Ontologien, nach Ehrig (2007), S. 21	39
Abbildung 15: Beispielcode für ein Ontology Alignment nach McIlraith et al. (2004)	40
Abbildung 16: Prozesslebenszyklus nach zur Mühlen (2004), S. 86.....	44
Abbildung 17: Beispiele verschiedener Geschäftsprozessmodellierungssprachen, nach Mendling (2007), S. 9.	47
Abbildung 18: Service Discovery Prozess nach Keller et al. (2007), S. 295.....	51

Abbildung 19: Beispiel von Treffern des Werkzeuges OWLS-MX, nach Klusch et al. (2009), S. 131.	55
Abbildung 20: Benutzeroberfläche von OWLS-MX 1.1c, nach Klusch et al. (2009), S. 132.	56
Abbildung 21: Architektur des DAML-S/UDDI Matchmaker nach Sycara et al. (2003), S. 38.	57
Abbildung 22: Semantische Annotation, Publikation und Discovery, nach Sivashanmugam et al. (2003), S. 401.	61
Abbildung 23: Schnittstellenbasiertes Matching mit SAWSDL-MX.	62
Abbildung 24: Service-Derivat in WSMX-MX, nach Klusch und Kaufer (2009), S. 26.	65
Abbildung 25: Interaktion zwischen set-theoretischen Kriterien, Intentionen und intuitiven Verständnis des Matchings, nach Keller et al. (2007), S. 307.	68
Abbildung 26: Erweitertes Servicemodell, nach Keller et al. (2007), S 316.	69
Abbildung 27: WSMX Architektur, siehe www.wsmx.org	70
Abbildung 28: Die bi-direktionale Kluft zwischen Fachseite und IT, nach Hepp et al. (2005), S. 537.	72
Abbildung 29: Semantisches Geschäftsprozessmanagement, nach Hepp et al. (2005), S. 538.	73
Abbildung 30: Der Semantic Business Process Modeling Lifecycle, nach Wetzstein et al. (2007).	73
Abbildung 31: Erweitertes semiotisches Dreieck, nach Thomas und Fellmann (2009), S. 508.	76
Abbildung 32: Beispielhafte semantische Annotation einer EPK, nach Thomas und Fellmann (2009), S. 511.	77
Abbildung 33: Bestandteile der sBPMN (nicht vollständig), nach Jędrzejczak et al. (2008), S. 9.	78
Abbildung 34: Realisierung semantischer Geschäftsprozessmodellierung, nach Born et al. (2007), S: 263.	79
Abbildung 35: BPMO Editor Ansicht im WSMO Studio, nach Dimitrov et al. (2007c), S. 103.	80
Abbildung 36: Komponentenarchitektur zur Komposition von semantischen Geschäftsprozessen, nach Weber et al. (2007), S. 58.	83

Abbildung 37: Semantischer asynchroner Aufruf eines BPEL4SWS-Prozesses, nach van Lessen et al. (2009), S. 204.....	87
Abbildung 38: Architektur einer BPEL4SWS-Engine, nach van Lessen et al. (2009), S. 207.....	88
Abbildung 39: Die <i>Process Mining Ontology</i> als Aktivitätsdiagramm in UML, nach Pedrinaci und Domingue (2007), S: 85.	93
Abbildung 40: Sequentielle Komposition von Matchern auf Basis von Ähnlichkeitsmatrizen, nach Euzenat und Shvaiko (2007), S. 119..	108
Abbildung 41: Parallele Komposition von Matchern auf Basis von Ähnlichkeitsmatrizen, nach Euzenat und Shvaiko (2007), S. 120..	109
Abbildung 42: Precision/Recall Graph für den Benchmark Track der OAEI 2010, nach Euzenat et al. (2010), S. 96.....	114
Abbildung 43: Systemarchitektur von Falcon-AO, nach Hu und Qu (2008), S. 238.	116
Abbildung 44: Architektur von Lily, nach Wang und Xu (2009), S. 187.....	118
Abbildung 45: Block-Diagramm des ASMOV-Algorithmus, nach Jean-Mary et al. (2009), S. 238.....	120
Abbildung 46: Prototypische Implementierung zum Umgang einer Vielzahl heterogener Ontologien, nach Kim und Lee (2010), S. 1540.....	125
Abbildung 47: Beispiel der Nutzung von WSMO Mediatoren, nach Roman et al. (2005), S. 97.....	127
Abbildung 48: Set-up zur Erstellung von Mediatoren für den Umgang mit heterogenen Ontologien in der semantischen Service Discovery, nach Della Valle et al. (2005), S. 39.....	128
Abbildung 49: SOA in der Automobilindustrie.....	131
Abbildung 50: Klassifizierung von Services nach Offermann (2009), S. 31.....	133
Abbildung 51: Ableitung relevanter Informationen aus dem GPM Kontext.....	134
Abbildung 52: Informationen in Varianten der Prozessmodellierung am Beispiel BPMN.	135
Abbildung 53: Relevanz verschiedener Serviceklassen für die GPM.....	136
Abbildung 54: Nutzung von Vorbedingungen und Effekten zur Servicebeschreibung	136
Abbildung 55: Beispielhaftes "Deal" Geschäftsartefakt nach Cohn und Hull (2009), S. 5.....	138

Abbildung 56: Beispiel zur Verwendung von Geschäftsobjekten und deren Semantik.....	139
Abbildung 57: Verwendung von Datenobjekten und deren Status für die Beschreibung von semantischen Geschäftsprozessmodellen, nach Born et al. (2007), S. 265.	143
Abbildung 58: Beispiel für den Einsatz von Vorbedingung und Effekt in einem Abrechnungsservice.....	146
Abbildung 59: Beispielcode für einen Abrechnungsservice mit Vorbedingungen und Effekt als WSMO Web Service	149
Abbildung 60: Beispielcode für eine Ontologie zur Darstellung des Abrechnungsservice als WSMO Ontology	149
Abbildung 61: Beispielcode für die Anfrage eines Service zur Rechnungsstellung nach einem Fertigungsauftrag als WSMO Goal.....	150
Abbildung 62: Detaillierte Darstellung des OWL-S Profile, nach Martin et al. (2007), S. 250.....	151
Abbildung 63: Beispielcode für einen Abrechnungsservice mit Verweis auf Vorbedingungen und Effekt (Result) als OWL-S Service Profile....	152
Abbildung 64: Beispielcode die Definition von Vorbedingungen und Effekt als OWL-S Service Process unter Verwendung des Knowledge Interchange Format (KIF)	152
Abbildung 65: Beispielcode zur Referenzierung von Vorbedingungen und Effekten in einer WSDL2-Datei mit SAWSDL	154
Abbildung 66: Verwertbare Informationen in der Geschäftsprozessmodellierung.....	163
Abbildung 67: Formulierung einer Serviceanfrage mit extrahierten Informationen ...	164
Abbildung 68: Integration der Servicesuche in die Prozessmodellierung.....	165
Abbildung 69: Einfacher Algorithmus der Service Discovery.....	170
Abbildung 70: Heterogene Service Discovery unter Verwendung von Ontology Alignments.....	175
Abbildung 71: Fassadenkonzept zur Einbindung unterschiedlicher bestehender Alignmentwerkzeuge.....	178
Abbildung 72: Autovervollständigung bei der Suche nach Produkten, nach Ortega et al. (2003), S.1.	183
Abbildung 73: "In-Place" Autovervollständigung nach Sinkkilä et al. (2008).....	184
Abbildung 74: Klassendiagramm der „Service Task“ der BPMN nach OMG (2011), S.	186

Abbildung 75: Architektur der prototypischen Implementierung.....	191
Abbildung 76: Architektur des WSMML2Reasoner Framework, angepasst für Pellet nach Krummenacher et al. (2010)	195
Abbildung 77: Fassadenkonzept zur Einbindung unterschiedlicher Alignment- Werkzeuge.....	196
Abbildung 78: Umsetzung des Fassadenkonzepts in der prototypischen Implementierung	197
Abbildung 79: Beispielhafte OWL-Ontologie mit Gleichheitsaxiomen, erstellt aus einem OAEI Alignment, nach Euzenat (2004), S. 708.....	198
Abbildung 80: Relevante Schnittstellen von WSMO4j für die Extraktion von Vorbedingungen und Effekten nach Dimitrov et al. (2006), S. 30 und 31.....	199
Abbildung 81: Unmodifizierte Oberfläche des BPMN Editors der SOA Tools Platform in Eclipse.....	201
Abbildung 82: WSMO Navigator für WSMO Ontologien des WSMO Studio Werkzeug, nach Dimitrov et al. (2007a), S. 31.....	202
Abbildung 83: Architektur des modularen Ontology Matching Systems	205
Abbildung 84: Bestandteile der Simple Ontology API	206
Abbildung 85: Klassen zur Definition von Basisalgorithmen	207
Abbildung 86: Benutzeroberfläche des modularen Ontology Matchmaking Systems	209
Abbildung 87: Auswertung der Service Discovery mit und ohne automatischem Alignment unter Verwendung von Falcon-OA.....	212
Abbildung 88: Auswertung der Service Discovery mit und ohne automatischem Alignment unter Verwendung von Lily.....	213
Abbildung 89: Domänenmodell des Telekommunikationsanbieters.....	216
Abbildung 90: Beispielcode für eine Ontologie zur Darstellung des Abrechnungsservice als OWL Ontology.....	260

Tabellenverzeichnis

Tabelle 1: Übersicht von SWS Ansätzen	29
Tabelle 2: Semantische Relationen in WordNet, nach Miller (1995), S. 40.....	102
Tabelle 3: Ergebnisse der OAEI 2007 für Falcon-AO, nach Hu und Qu (2008), S. 239.	117
Tabelle 4: Entwicklung der Ansätze mit den besten Ergebnissen im Benchmark- Test der OAEI, nach Caracciolo et al. (2008), S. 82.	119
Tabelle 5: Resultate des Benchmark Tests der OAEI für ASMVO 2010 und 2009, nach Jean-Mary et al. (2010), S. 129.	122
Tabelle 6: Konzeptionelle Bestandteile der geschäftsprozessorientierten Servicebeschreibung	145
Tabelle 7: Auswahl an BPMN Modellierungskonzepten und Relevanz für die GPM-getriebenen Service Discovery.....	158
Tabelle 8: Mögliche Trefferkombinationen für Vorbedingungen und Effekte der Operationen im Rahmen der Servicesuche	169
Tabelle 9: Anforderungen an die prototypische Implementierung einer heterogenen Service Discovery	190
Tabelle 10: Ergänzende Anforderungen im Rahmen der GPM-getriebenen Service Discovery	191

Abkürzungsverzeichnis

A

ACID	Atomicity, Consistency, Isolation und Durability
AI	Artificial Intelligence
API	Application Programming Interface
ARIS	Architektur integrierter Informationssysteme
ASMOV	Automated Semantic Matching of Ontologies with Verification

B

BO	Business Object
BP	Business Process
BPE	Business Process Expert
BPEL	Business Process Execution Language
BPEL4WS	Business Process Execution Language for Web Services
BPEL4SWS	Business Process Execution Language for Semantic Web Services
BPM	Business Process Management / Business Process Modeling
BPMI	Business Process Management Initiative
BPML	Business Process Modeling Language
BPMN	Business Process Model and Notation

C

CRUD	Create Read Update Delete
------	---------------------------

D

DAML	Darpa Agent Markup Language
DAML-S	Darpa Agent Markup Language for Services
DARPA	Defense Advanced Research Projects Agency

DL	Description Logic
----	-------------------

E

EAI	Enterprise Application Integration
ebXML	Electronic Business XML
EDV	Elektronische Datenverarbeitung
EON	Evaluation of Ontology-based Tools (Workshop)
EPK	Ereignisgesteuerte Prozesskette
eEPK	Erweiterte Ereignisgesteuerte Prozesskette
ESB	Enterprise Service Bus
eTOM	Enhanced Telecom Operations Map
EVO	Events Ontology

F

F-Logic	Frame Logic
FOL	First Order Logic

G

GMO	Graph Matching for Ontologies
GO	Geschäftsobjekt
GOM	Generic Ontology Matching
GPE	Geschäftsprozessexperte
GPM	Geschäftsprozessmodellierung / Geschäftsprozessmanagement
GPM-SD	Geschäftsprozessmodellierungsgetriebene Service Discovery
GSD	Geschäftsprozessorientierte Service Discovery
GUI	Graphical User Interface

H

HTTP	Hypertext Transfer Protocol
------	-----------------------------

I

ID	Identifier
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IOPR	Input Output Precondition Effect
IS	Informationssystem (engl. information system)
ISBN	International Standard Book Number
ISO	International Organization for Standardization
ISR	Information Systems Research
ISWC	International Semantic Web Conference
IT	Informationstechnologie (engl. information technology)

J

JEE	Java Enterprise Edition
-----	-------------------------

K

KIF	Knowledge Interchange Format
KPI	Key Performance Indicator

L

LOI	Loss of Information
LOM	Large Scale Ontology Matching

M

MWSDI	METEOR-S Web Services Discovery Infrastructure
MXML	Mining XML

N

NCI	National Cancer Institute
-----	---------------------------

NFP	Non-functional property
NLP	Natural Language Processing

O

OAEI	Ontology Alignment Evaluation Initiative
OASIS	Organization for the Advancement of Structured Information Standards
ODE	Orchestration Director Engine
OIL	Ontology Inference Layer
OMG	Object Management Group
OWL	Web Ontology Language
OWL-S	Web Ontology Language for Services

P

PBM	Partion-based Block Matching
PMO	Process Mining Ontology
ProM	Process Mining

Q

QoS	Quality of Service
-----	--------------------

R

RACER	Renamed Abox and Concept Expression Reasoner
RBE	Reverse Business Engineering
RDF	Ressource Description Framework
RDFS	Ressource Description Framework Schema
RiMOM	Risk Minimization based Ontology Mapping

S

SA-MXML	Semantically Annotated Mining XML
---------	-----------------------------------

SAWSDL	Semantic Annotations for WSDL and XML Schema
sBPM	Semantic BPM
SDD	Semantic Description Document
SEALS	Semantic Evaluation at Large Scale
SKOS	Simple Knowledge Organisation System
SOA	Service-Oriented Architecture/ Serviceorientierte Architektur
SOAP	Simple Object Access Protocol
SOM	Semantic Ontology Matching
SSP	Similarity Propagation Matcher
STP	SOA Tools Platform
SUMO	Suggested Upper Merged Ontology
SWRL	Semantic Web Rule Language
SWS	Semantic Web Services / Semantische Web Services
 U	
UDDI	Universal Description, Discovery and Integration
UI	User Interface
UML	Unified Modeling Language
ULO	Upper Level Ontology
URI	Uniform Ressource Identifier
 V	
v-DOC	Virtual Document
 W	
W3C	World Wide Web Consortium
WfML	Workflow Management Coalition
WS-BPEL	Web Services Business Process Execution Language
WSDL	Web Services Description Language

WSDL-S	Web Services Description Language with semantic extension
WSFL	Web Services Flow Language
WSML	Web Service Modeling Language
WSMO	Web Service Modeling Ontology
WSMX	Web Service Execution Environment
WWW	World Wide Web

X

XLANG

XML	Extensible Markup Language
-----	----------------------------

Y

YAWL	Yet Another Workflow Language
------	-------------------------------

Verzeichnis eigener Veröffentlichungen

- Rake, J., Holschke, O., und Levina, O. (2009). Enhancing Semantic Service Discovery in Heterogeneous Environments, Business Information Systems. In: Lecture Notes in Business Information Processing. W. Aalst, J. Mylopoulos, M. Rosemann, M. J. Shaw und C. Szyperski, Springer Berlin Heidelberg, **21**: 205-216.
- Rake, J. und Schönherr, M. (2009). Decoupling of Heterogeneous Semantic Service Annotations and Their Semantic Models – Towards an Integrated Approach in a Multi-Service-Provider-Scenario. In: Service-Oriented Computing - ICSOC 2007 Workshops. Springer Berlin / Heidelberg. 4907: 188-199.
- Holschke, O., Rake, J., Offermann, P., und Bub, U. (2010). "Steigerung der Softwareflexibilität bei Geschäftsprozessänderungen."
WIRTSCHAFTSINFORMATIK 52(1): 3-15.
- Rake, J., Holschke, O., Offermann, P., und Bub, U. (2010). "Flexible Softwareänderungen in Platform-as-a-Service-Angeboten."
WIRTSCHAFTSINFORMATIK Management 2: 52-59.

Literatur

- Abramowicz, W., A. Filipowska, et al. (2007). "Semantically enhanced business process modelling notation." Proceedings of the Workshop SBPM 2007, Innsbruck, April 7, 2007 251.
- Aguilera, U., J. Abaitua, et al. (2007). "A Semantic Matching Algorithm for Discovery in UDDI." Semantic Computing, 2007. ICSC 2007. International Conference on: 751-758.
- Aier, S. (2007). Integrationstechnologien als Basis einer nachhaltigen Unternehmensarchitektur - Abhängigkeiten zwischen Organisation und Informationstechnologie. Berlin, Gito.
- Aier, S. and R. Winter (2009). "Virtual decoupling for IT/Business Alignment - Conceptual Foundations, Architecture Design and Implementation Example." Business & Information Systems Engineering 1(2): 175-191.
- Akkiraju, R., J. Farrell, et al. (2005). "Web Service Semantics - WSDL-S." Retrieved 28.07.2007, from <http://www.w3.org/Submission/WSDL-S/>.
- Allweyer, T. (2009). BPMN 2.0 - Business Process Model and Notation: Einführung in den Standard für die Geschäftsprozessmodellierung. Norderstedt, Books on Demand GmbH.
- Alves, A., A. Arkin, et al. (2007). "Web Services Business Process Execution Language Version 2.0." OASIS Standard Retrieved 21.04.2011, from <http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html>.
- Alves de Medeiros, A. and W. van der Aalst (2009). Process Mining towards Semantics. In: Advances in Web Semantics I. T. Dillon, E. Chang, R. Meersman und K. Sycara, Springer Berlin / Heidelberg, 4891: 35-80.
- Baader, F., D. Calvanese, et al. (2003). The description logic handbook: theory, implementation, and applications. New York, N.Y., Cambridge University Press.
- Bajaj, S., D. Box, et al. (2006). "Web Services Policy 1.2 - Attachment (WS-PolicyAttachment)." W3C Member Submission Retrieved 26.04.2011, from <http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-20060425/>.
- Bechhofer, S., F. van Harmelen, et al. (2004). "OWL Web Ontology Language Reference." W3C Recommendation 10 February 2004.
- Becker, J. and D. Kahn (2005). Der Prozess im Fokus. In: Prozessmanagement. J. Becker, M. Kugeler und M. Rosemann, Springer Berlin Heidelberg: 3-16.

- Bellahsène, Z., M. Léonard, et al. (2008). On the Duality of Information-Centric and Activity-Centric Models of Business Processes. In: Advanced Information Systems Engineering, Springer Berlin / Heidelberg, 5074: 32-47.
- Benatallah, B., M. S. Hacid, et al. (2003). "Request Rewriting-Based Web Service Discovery." International Semantic Web Conference: 242-257.
- Berardi, D., D. Calvanese, et al. (2005). "Automatic composition of transition-based semantic web services with messaging." Proceedings of Proceedings of the 31st international conference on Very Large Data Bases: 613-624.
- Berge, C. (1976). Graphs and hypergraphs. Amsterdam, London, North-Holland Publishing Company.
- Berners-Lee, T., J. Hendler, et al. (2001). "The semantic Web." Scientific American 284(5): 28-37.
- Bieberstein, N., S. Bose, et al. (2006). Service Oriented Architecture Compass: Business Value, Planning, and Enterprise Roadmap, IBM Press.
- Born, M., C. Brelage, et al. (2009). Auto-completion for Executable Business Process Models. In: Business Process Management Workshops. D. Ardagna, M. Mecella und J. Yang, Springer Berlin Heidelberg. 17: 510-515.
- Born, M., F. Dörr, et al. (2007). User-Friendly Semantic Annotation in Business Process Modeling. In: Web Information Systems Engineering – WISE 2007 Workshops. M. Weske, M.-S. Hacid und C. Godart, Springer Berlin / Heidelberg, 4832: 260-271.
- Born, M., J. Hoffmann, et al. (2008). Semantic Annotation and Composition of Business Processes with Maestro. In: The Semantic Web: Research and Applications. S. Bechhofer, M. Hauswirth, J. Hoffmann und M. Koubarakis, Springer Berlin / Heidelberg. 5021: 772-776.
- Brickley, D. and R. V. Guha. (2004). "RDF Vocabulary Description Language 1.0: RDF Schema." W3C Recommendation Retrieved 15.03.2011, from <http://www.w3.org/TR/2004/REC-rdf-schema-20040210/>.
- Bussler, C., E. Cimpian, et al. (2005). "Web Service Execution Environment (WSMX)." W3C Member Submission Retrieved 30.03.2011, from <http://www.w3.org/Submission/2005/SUBM-WSMX-20050603/>.
- Calado, P., M. Herschel, et al. (2010). An Overview of XML Duplicate Detection Algorithms. In: Soft Computing in XML Data Management. Z. Ma und L. Yan, Springer Berlin / Heidelberg. 255: 193-224.
- Caracciolo, C., J. Euzenat, et al. (2008). "Results of the Ontology Alignment Evaluation Initiative 2008." Proceedings of the Ontology Matching Workshop at the International Semantic Web Conference (ISWC 2008): 73-119.

- Cardoso, J. and A. Sheth (2003). "Semantic E-Workflow Composition." Journal of Intelligent Information Systems **21**(3): 191-225.
- Celino, I., A. K. A. d. Medeiros, et al. (2007). "Semantic Business Process Analysis." Proceedings of Workshops on Semantic Business Process and Product Lifecycle Management (SBPM) at the 4th European Semantic Web Conference (ESWC) **251**: 44-47.
- Cerbah, F. and J. Euzenat (2001). "Traceability between models and texts through terminology." Data & Knowledge Engineering **38**(1): 31-43.
- Cervantes, H. and R. S. Hall (2004). "Autonomous adaptation to dynamic availability using a service-oriented component model." Proceedings of the 26th International Conference on Software Engineering (ICSE): 614-623.
- Christos, F., S. M. Kevin, et al. (2004). "Fast discovery of connection subgraphs." Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining: 118-127.
- Clement, L., A. Hatelly, et al. (2004). "UDDI Version 3.0.2." UDDI Spec Technical Committee Draft, Dated 20041019.
- Clemons, E. K. (1986). "Information Systems for Sustainable Competitive Advantage." Information and Management **11**(3): 131-136.
- Cohn, D. and R. Hull (2009). "Business artifacts: A data-centric approach to modeling business operations and processes." IEEE Data Engineering **32**(3): 3-9.
- Connolly, D., F. v. Harmelen, et al. (2001). "DAML+OIL (March 2001) Reference Description." W3C Note Retrieved 15.03.2011, from <http://www.w3.org/TR/2001/NOTE-daml+oil-reference-20011218>.
- Cyert, R. M. and J. G. March (1963). A Behavioral Theory of the Firm. New Jersey, USA, Prentice-Hall Inc.
- Davenport, T. H. (1993). Process Innovation: Reengineering Work through Information Technology. Boston, Mass., Harvard Business School Press.
- de Bruijn, J., H. Lausen, et al. (2006). The Web Service Modeling Language WSM: An Overview. In: The Semantic Web: Research and Applications. Y. Sure und J. Domingue, Springer Berlin / Heidelberg. **4011**: 590-604.
- Della Valle, E., D. Cerizza, et al. (2005). The mediators centric approach to Automatic Web Service Discovery of Glue. In: Proceedings of the First International Workshop on Mediation in Semantic Web Services (MEDIATE 2005). M. Hepp, A. Polleres, F. v. Harmelen und M. Genesereth. **168**: 35-50.
- Di Francescomarino, C., C. Ghidini, et al. (2009). Semantically-Aided Business Process Modeling. In: The Semantic Web - ISWC 2009. A. Bernstein, D. Karger, T. Heath et al, Springer Berlin / Heidelberg. **5823**: 114-129.

- Dimitrov, M., V. Momtchev, et al. (2006). wsmo4j Programmers Guide, Ontotext Lab. / Sirma Group.
- Dimitrov, M., A. Simov, et al. (2007a). WSMO Studio User Guide, Ontotext.
- Dimitrov, M., A. Simov, et al. (2007b). WSMO Studio – A Semantic Web Services Modelling Environment for WSMO In: The Semantic Web: Research and Applications, Springer. 4519: 749-758.
- Dimitrov, M., A. Simov, et al. (2007c). "A BPMO Based Semantic Business Process Modelling Environment." Proceedings of Workshops on Semantic Business Process and Product Lifecycle Management (SBPM) at the 4th European Semantic Web Conference (ESWC) 251: 101-104.
- Duke, A., J. Davies, et al. (2005). "Enabling a scalable service-oriented architecture with semantic Web Services." BT Technology Journal 23(3): 191-201.
- Ehrig, M. (2007). Ontology Alignment: Bridging the Semantic Gap. Heidelberg, Springer.
- Ehrig, M. and Y. Sure (2004). Ontology Mapping – An Integrated Approach. In: The Semantic Web: Research and Applications. C. Bussler, J. Davies, D. Fensel und R. Studer, Springer Berlin / Heidelberg. 3053: 76-91.
- Elenius, D., G. Denker, et al. (2005). The OWL-S Editor – A Development Tool for Semantic Web Services. In: The Semantic Web: Research and Applications. A. Gómez-Pérez und J. Euzenat, Springer Berlin / Heidelberg. 3532: 45-66.
- Elhadad, M., M. Balaban, et al. (2008). "Effective Business Process Outsourcing: The Prosero Approach." International Journal of Interoperability in Business Information Systems 3(1).
- Erl, T. (2005). Service-Oriented Architecture - Concepts, Technology, and Design. Upper Saddle River, NJ, Prentice Hall Professional Technical Reference.
- Euzenat, J. (2004). An API for Ontology Alignment. In: The Semantic Web – ISWC 2004. S. A. McIlraith, D. Plexousakis und F. van Harmelen, Springer Berlin / Heidelberg. 3298: 698-712.
- Euzenat, J. (2006). An API for ontology alignment (version 2.1), INRIA Rhône-Alpes, Montbonnot, Frankreich.
- Euzenat, J. (2008). Algebras of Ontology Alignment Relations. In: The Semantic Web – ISWC 2008, Springer Berlin / Heidelberg. 5318: 387-402.
- Euzenat, J. (2011). "Ontology Alignment Evaluation Initiative." 8.6.2011, from <http://oaci.ontologymatching.org/>.
- Euzenat, J., J. Barrasa, et al. (2004). Knowledge Web Deliverable D2.2.3: State of the art on ontology alignment. Technical Report.

- Euzenat, J., A. Ferrara, et al. (2010). "Results of the Ontology Alignment Evaluation Initiative 2010." Proceedings of the 5th International Workshop on Ontology Matching (OM-2010) **689**: 85-117.
- Euzenat, J. and P. Shvaiko (2007). Ontology matching. Heidelberg, Springer.
- Euzenat, J. and P. Valtchev (2004). "Similarity-based ontology alignment in OWL-lite." Proceedings of the 16th European Conference on Artificial Intelligence (ECAI): 333-337.
- Farrell, J. and H. Lausen. (2007). "Semantic Annotations for WSDL and XML Schema." Retrieved 01.08.2008, from <http://www.w3.org/TR/sawSDL/>.
- Fensel, D. and F. van Harmelen (2007). "Unifying Reasoning and Search to Web Scale." Internet Computing, IEEE **11**(2): 96-95.
- Ferstl, O. K. and E. J. Sinz (1990). "Objektmodellierung betrieblicher Informationssysteme im Semantischen Objektmodell (SOM)." Wirtschaftsinformatik **32**(6): 566-581.
- Frege, G. (1892). "Über Sinn und Bedeutung." Zeitschrift für Philosophie und philosophische Kritik **100**(1): 25-50.
- Furnas, G. W., T. K. Landauer, et al. (1987). "The vocabulary problem in human-system communication." Communications of the ACM **30**(11): 964-971.
- Gaitanides, M. (2004). Prozessorganisation. In: Handwörterbuch Unternehmensführung und Organisation. G. Schreyögg und A. v. Werder. Stuttgart, Schäffer-Poeschel: 1208-1218.
- Gangemi, A., N. Guarino, et al. (2003). "Sweetening WORDNET with DOLCE." AI Magazine **24**(3): 13-24.
- Gordijn, J. and H. Akkermans (2001). "e3-value: Design and Evaluation of e-Business Models." IEEE Intelligent Systems **16**(4): 11-17.
- Grimm, S. (2007). Discovery - Identifying Relevant Services. In: Semantic Web Services. R. Studer, S. Grimm und A. Abecker, Springer Berlin Heidelberg: 211-244.
- Grimm, S., P. Hitzler, et al. (2007). Knowledge Representation and Ontologies. In: Semantic Web Services: Concepts, Technologies and Applications. R. Studer, S. Grimm und A. Abecker. Heidelberg, Springer: 51-106.
- Gruber, T. R. (1993). "A translation approach to portable ontology specifications." Knowledge Acquisition **5**(2): 199-220.
- Haarslev, V. and R. Müller (2001). RACER System Description. In: Automated Reasoning. R. Goré, A. Leitsch und T. Nipkow, Springer Berlin / Heidelberg. **2083**: 701-705.

- Haller, A., E. Cimpian, et al. (2005). "WSMX - A Semantic Service-Oriented Architecture." Proceedings. 2005 IEEE International Conference on Web Services: 321-328.
- Hammer, M. and J. Champy (1994). Business Reengineering – Die Radikalkur für das Unternehmen. Frankfurt/Main, New York, Campus.
- Hamming, R. W. (1950). "Error detecting and error correcting codes." Bell System Technical Journal **29**(2): 147-160.
- Hendler, J. and D. L. McGuinness (2000). "The DARPA agent markup language." IEEE Intelligent Systems **15**(6): 67-73.
- Hepp, M., F. Leymann, et al. (2005). "Semantic business process management: a vision towards using semantic Web services for business process management." Proceedings of the IEEE International Conference on e-Business Engineering: 535-540.
- Hepp, M. and D. Roman (2007). "An ontology framework for semantic business process management." Proceedings of Wirtschaftsinformatik, Karlsruhe, 28. Februar - 2. März 2007.
- Heutschi, R. (2007). Serviceorientierte Architektur: Architekturprinzipien und Umsetzung in die Praxis. Berlin, Springer.
- Heutschi, R. and J. W. Schemm (2005). Fallstudie: Service-orientierte Architektur bei Deutsche Post Brief. St. Gallen, Institut für Wirtschaftsinformatik, Universität St. Gallen.
- Hevner, A. R., S. T. March, et al. (2004). "Design Science in Information Systems Research." MIS Quarterly **28**(1): 75-105.
- Hollingsworth, D. (1993). The workflow reference model. The WfMC Specification, Workflow Management Coalition.
- Holschke, O., J. Rake, et al. (2010). "Steigerung der Softwareflexibilität bei Geschäftsprozessänderungen." WIRTSCHAFTSINFORMATIK **52**(1): 3-15.
- Horn, A. (1951). "On sentences which are true of direct unions of algebras." The Journal of Symbolic Logic **16**(1): 14-21.
- Hu, W., J. Chen, et al. (2010). "ObjectCoref & Falcon-AO: results for OAEI 2010." Proceedings of the 5th International Workshop on Ontology Matching (OM-2010) **689**: 158-165.
- Hu, W., N. Jian, et al. (2005). "GMO: A graph matching for ontologies." Proceedings of the Integrating Ontologies Workshop at K-CAP 2005: 41-48.

- Hu, W. and Y. Qu (2008). "Falcon-AO: A practical ontology matching system." Web Semantics: Science, Services and Agents on the World Wide Web 6(3): 237-239.
- Hu, W., Y. Zhao, et al. (2007). "Falcon-AO: results for OAEI 2007." Proceedings of the 2nd International Workshop on Ontology Matching (OM-2007) 304: 170-178.
- Iqbal, K., M. L. Sbodio, et al. (2008). Semantic Service Discovery using SAWSDL and SPARQL. Proceeding of the Fourth International Conference on Semantics, Knowledge and Grid. Beijing, IEEE Computer Society: 205-212.
- Jaro, M. A. (1989). "Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida." Journal of the American Statistical Association 84(406): 414-420.
- Jean-Mary, Y. R., E. P. Shironoshita, et al. (2009). "Ontology matching with semantic verification." Web Semantics: Science, Services and Agents on the World Wide Web 7(3): 235-251.
- Jean-Mary, Y. R., E. P. Shironoshita, et al. (2010). "ASMOV: Results for OAEI 2010." Proceedings of the 5th International Workshop on Ontology Matching (OM-2010) 689: 126-133.
- Jędrzejczak, A., A. Filipowska, et al. (2008). Deliverable 4.5: sBPMN and sEPC to BPMN Translation. SUPER Deliverable, EU IST Project: Semantics Utilised for Process management within and between Enterprises.
- Jurafsky, D. and J. H. Martin (2009). Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition. Upper Saddle River, New Jersey, Prentice Hall.
- Kaufer, F. and M. Klusch (2006a). WSMO-MX: a logic programming based hybrid service matchmaker. In: Proceedings of the 4th IEEE European Conference on Web Services (ECOWS'06): 161-170.
- Kaufer, F. and M. Klusch (2006b). "WSMO-MX: a logic programming based hybrid service matchmaker." Proceedings of the 4th IEEE European Conference on Web Services (ECOWS2006), Zürich, Switzerland: 161-170.
- Keller, G., M. Nüttgens, et al. (1992). "Semantische Prozeßmodellierung auf der Grundlage „Ereignisgesteuerter Prozeßketten (EPK)“." Veröffentlichungen des Instituts für Wirtschaftsinformatik (IWi), Universität des Saarlandes(89).
- Keller, U., R. Lara, et al. (2007). Semantic Web Service Discovery in the WSMO Framework. In: Semantic Web Services: Theory, Tools and Applications. J. Cardoso, Springer: 281-316.

- Keller, U., R. Lara, et al. (2005). Automatic Location of Services. In: The Semantic Web: Research and Applications. A. Gómez-Pérez und J. Euzenat, Springer Berlin / Heidelberg, 3532: 1-16.
- Kelly, M. (2007). Enhanced Telecom Operations Map (eTOM) - The Business Process Framework, TeleManagement Forum.
- Kifer, M. and G. Lausen (1989). F-logic: a higher-order language for reasoning about objects, inheritance, and scheme. Proceedings of the 1989 ACM SIGMOD international conference on Management of data. Portland, Oregon, United States, ACM.
- Kifer, M., G. Lausen, et al. (1995). "Logical foundations of object-oriented and frame-based languages." Journal of the ACM (JACM) 42(4): 741-843.
- Kim, D. and Y. Lee (2010). "Semantic Service Discovery Exploiting Efficient Concept Match over Large Amount of Heterogeneous Ontologies." Proceedings of the 10th IEEE International Conference on Computer and Information Technology: 1536-1541.
- King, W. R., V. Grover, et al. (1989). "Using information and information technology for sustainable competitive advantage: some empirical evidence." Information & Management 17: 87-93.
- Klein, M. (2001). "Combining and relating ontologies: an analysis of problems and solutions." Proceedings of the IJCAI-01 Workshop on Ontologies and Information Sharing.
- Klusch, M. and B. Fries (2009). Hybrid OWL-S service retrieval with OWLS-MX: Benefits and pitfalls. Proceedings of the First International Joint Workshop on Service Matchmaking and Resource Retrieval in the Semantic Web. Busan, Republic of Korea, IEEE Press.
- Klusch, M., B. Fries, et al. (2005). Owls-mx: Hybrid owl-s service matchmaking. Proceedings of 1st Intl. AAAI Fall Symposium on Agents and the Semantic Web Arlington VA, USA.
- Klusch, M., B. Fries, et al. (2006). Automated semantic web service discovery with OWLS-MX. Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems: 915-922.
- Klusch, M., B. Fries, et al. (2009). "OWLS-MX: A hybrid Semantic Web service matchmaker for OWL-S services." Web Semantics Science Services and Agents on the World Wide Web 7(2): 121-133.
- Klusch, M. and P. Kapahnke (2008). Semantic web service selection with SAWSDL-MX. Proceedings of the 7th International Semantic Web Conference. Karlsruhe, Germany: 3-18.

- Klusch, M., P. Kapahnke, et al. (2008). "Evaluation of WSMML Service Retrieval with WSMO-MX." Proceedings of the IEEE International Conference on Web Services, Beijing, China: 401-408.
- Klusch, M. and F. Kaufer (2009). "WSMO-MX: A hybrid Semantic Web service matchmaker." Web Intelligence and Agent Systems 7(1): 23-42.
- Klyne, G. and J. J. Carroll. (2004). "Resource Description Framework (RDF): Concepts and Abstract Syntax." W3C Recommendation Retrieved 15.03.2011, from <http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/>
- Knublauch, H., R. W. Ferguson, et al. (2004). The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications. In: The Semantic Web – ISWC 2004, S. A. McIlraith, D. Plexousakis und F. van Harmelen, Springer Berlin / Heidelberg. 3298: 229-243.
- Kopecký, J., T. Vitvar, et al. (2007). "SAWSDL: Semantic Annotations for WSDL and XML Schema." IEEE Internet Computing 11(6): 60-67.
- Kourtesis, D. and I. Paraskakis (2008). Combining SAWSDL, OWL-DL and UDDI for Semantically Enhanced Web Service Discovery. In: The Semantic Web: Research and Applications, S. Bechhofer, M. Hauswirth, J. Hoffmann und M. Koubarakis, Springer Berlin / Heidelberg. 5021: 614-628.
- Krafzig, D., K. Banke, et al. (2004). Enterprise SOA: Service-Oriented Architecture Best Practices (The Coad Series), Prentice Hall PTR Upper Saddle River, NJ, USA.
- Krafzig, D., K. Banke, et al. (2005). Enterprise SOA - Service-Oriented Architecture Best Practices. Upper Saddle River, NJ, Prentice Hall.
- Krallmann, H., M. Schönherr, et al. (2007). Systemanalyse im Unternehmen – Prozessorientierte Methoden der Wirtschaftsinformatik. München, Wien, Oldenbourg.
- Krummenacher, R., D. Winkler, et al. (2010). WSMML2Reasoner - A Comprehensive Reasoning Framework for the Semantic Web. 9th International Semantic Web Conference, ISWC 2010. Shanghai, China, Springer.
- Kühne, T. (2006). "Matters of (meta-) modeling." Journal on Software and Systems Modeling 5(4): 369-385.
- Kupiec, J. (1992). "Robust part-of-speech tagging using a hidden Markov model." Computer Speech & Language 6(3): 225-242.
- Kurz, M. (1993). Ein ganzheitlich orientierter Ansatz zur wirtschaftlichen Neugestaltung von Bürosystemen - Unterstützungsmöglichkeiten einer spezifischen Gruppenentscheidung. Heidelberg, Physica-Verlag.
- Lara, R., W. Binder, et al. (2005). Semantics for Web Service Discovery and Composition, Technical report, Knowledge Web.

- Lausen, H., R. Lara, et al. (2007). Description: Semantic Annotation for Web Services. In: Semantic Web Services: Concepts, Technologies and Applications. R. Studer, S. Grimm und A. Abecker. Heidelberg, Springer: 179-209.
- Legner, C. (2006). Abschlussbericht Competence Center Business Networking 3 (CC BN3). St. Gallen, Universität St. Gallen - Hochschule für Wirtschafts-, Rechts und Sozialwissenschaften (HSG).
- Legner, C. and R. Heutschi (2007). SOA Adoption in Practice - Findings from Early SOA Implementations. European Conference on Information Systems (ECIS 2007). St. Gallen.
- Levenshtein, V. I. (1965). "Binary codes capable of correcting deletions, insertions, and reversals." Soviet Physics Doklady **10**(8): 707-710.
- Levin, B. (1993). English verb classes and alternations, Chicago.
- Leymann, F. (2001). Web Services Flow Language (WSFL 1.0), IBM Software Group.
- Li, K. (2005). Lumina: using wsdl-s for web service discovery. LSDIS Lab, Department of Computer Science, Athens, University of Georgia. **Master's Thesis**.
- Li, K., B. Xu, et al. (2007). "An Ontology Mapping Approach Using Web Search Engine" Journal of Southeast University **23**(3): 352-356.
- Li, L. and I. Horrocks (2004). "A software framework for matchmaking based on semantic web technology." International Journal of Electronic Commerce **8**(4): 39-60.
- Lim, E. P., J. Srivastava, et al. (1993). "Entity identification in database integration." Proceedings of the Ninth International Conference on Data Engineering: 294-301.
- Lin, D. (1998). "An information-theoretic definition of similarity." Proceedings of the Fifteenth International Conference on Machine Learning (ICML): 296-304.
- Lublinsky, B. (2007). "Defining SOA as an architectural style." Retrieved 10.03.2011, from <http://www.ibm.com/developerworks/architecture/library/ar-soastyle/>.
- Manola, F. and E. Miller. (2004). "RDF Primer." W3C Recommendation Retrieved 15.03.2011, from <http://www.w3.org/TR/2004/REC-rdf-primer-20040210/>.
- Markovic, I. and A. Pereira (2008). Towards a Formal Framework for Reuse in Business Process Modeling. In: Business Process Management Workshops. A. ter Hofstede, B. Benatallah und H.-Y. Paik, Springer Berlin / Heidelberg. **4928**: 484-495.
- Martin, D., M. Burstein, et al. (2004). "OWL-S: Semantic Markup for Web Services." W3C Member Submission Retrieved 16.03.2011, from <http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/>.

- Martin, D., M. Burstein, et al. (2008). "OWL-S: Semantic markup for Web services." Technical Overview Retrieved 16.03.2011, from <http://www.ai.sri.com/dam/services/owl-s/1.2/overview/>.
- Martin, D., M. Burstein, et al. (2007). "Bringing Semantics to Web Services with OWL-S." World Wide Web 10(3): 243-277.
- Masak, D. (2006). IT-Alignment, Springer.
- Maurer, G. (1996). Von der Prozessorientierung zum Workflow Management. Teil 1: Prozessorientierung-Grundgedanken, Kernelemente, Kritik. Arbeitspapiere WI. Mainz, Lehrstuhl für Allgemeine BWL und Wirtschaftsinformatik. 9/1996.
- Mayer, R. E. (1997). "Multimedia learning: Are we asking the right questions?" Educational Psychologist 32(1): 1-19.
- McGuinness, D. L. and F. v. Harmelen. (2004). "OWL Web Ontology Language Overview." Retrieved 28.08.08, from <http://www.w3.org/TR/owl-features/>.
- McGuinness, D. L. and F. van Harmelen. (2004). "OWL Web Ontology Language - Overview" W3C Recommendation Retrieved 15.03.2011, from <http://www.w3.org/TR/2004/REC-owl-features-20040210/>.
- McIlraith, S. A., D. Plexousakis, et al. (2004). An API for Ontology Alignment. In: The Semantic Web – ISWC 2004, Springer Berlin / Heidelberg. 3298: 698-712.
- McIlraith, S. A., T. C. Son, et al. (2001). "Semantic Web services." IEEE Intelligent Systems 16(2): 46-53.
- Melnik, S., H. Garcia-Molina, et al. (2002). "Similarity flooding: a versatile graph matching algorithm and its application to schema matching." Proceedings of the 18th International Conference on Data Engineering (ICDE): 117-128.
- Mendling, J. (2007). Detection and prediction of errors in epc business process models. Institute of Information Systems and New Media, Vienna University of Economics and Business Administration. **Doctoral Thesis**.
- Mendling, J., H. A. Reijers, et al. (2009). "Activity labeling in process modeling: Empirical insights and recommendations." Information Systems 35(4): 467-482.
- Mendling, J., H. A. Reijers, et al. (2010). "Seven process modeling guidelines (7PMG)." Information and Software Technology 52(2): 127-136.
- Miles, A. and J. R. Perez-Aguera (2007). "SKOS: Simple knowledge organisation for the web." 43(3-4): 69-83.
- Miller, G. A. (1995). "WordNet: a lexical database for English." Communications of the ACM 38(11): 39-41.

- Miller, G. A., R. Beckwith, et al. (1990). "Introduction to WordNet: An On-line Lexical Database." International Journal of Lexicography 3(4): 235-244.
- Moitra, D. and J. Ganesh (2005). "Web services and flexible business processes: towards the adaptive enterprise." Information and Management 42(7): 921-933.
- Muñoz-Avila, H., F. Ricci, et al. (2005). Language Games: Solving the Vocabulary Problem in Multi-Case-Base Reasoning. In: Case-Based Reasoning Research and Development, Springer Berlin / Heidelberg. 3620: 35-49.
- Nigam, A. and N. S. Caswell (2003). "Business artifacts: An approach to operational specification." IBM Systems Journal 42(3): 428-445.
- Niles, I. and A. Pease (2001). "Towards a standard upper ontology." Proceedings of the international conference on Formal Ontology in Information Systems, Ogunquit, Maine, USA: 2-9.
- Nitzsche, J., T. van Lessen, et al. (2007a). BPEL for Semantic Web Services (BPEL4SWS). In: On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops. R. Meersman, Z. Tari und P. Herrero, Springer Berlin / Heidelberg. 4805: 179-188.
- Nitzsche, J., T. van Lessen, et al. (2007b). BPELLight. In: Business Process Management. G. Alonso, P. Dadam und M. Rosemann, Springer Berlin / Heidelberg. 4714: 214-229.
- Nitzsche, J., D. Wutke, et al. (2007c). "An Ontology for Executable Business Processes." Proceedings of Workshops on Semantic Business Process and Product Lifecycle Management (SBPM) at the 4th European Semantic Web Conference (ESWC) 251: 52-60.
- Noy, N. F. and M. A. Musen (2001). "Anchor-PROMPT: Using non-local context for semantic matching." Proceedings of the workshop on Ontologies and Information Sharing at the International Joint Conference on Artificial Intelligence: 63-70.
- Nüttgens, M. and V. Zimmermann (1998). Geschäftsprozeßmodellierung mit der objektorientierten Ereignisgesteuerten Prozeßkette (oEPK). In: Informationsmodellierung - Branchen , Software- und Vorgehensreferenzmodelle, Maicher, M., Scheruhn, H.J.: 23-36.
- Oey, K. J., H. Wagner, et al. (2005). Mehr als alter Wein in neuen Schläuchen: Eine einführende Darstellung des Konzepts der serviceorientierten Architekturen. In: Unternehmensarchitekturen und Systemintegration. S. Aier und M. Schönherr. Berlin, GITO: 197-220.
- Offermann, P. (2009). Eine Methode zur Konzeption betrieblicher Software mit einer serviceorientierten Architektur, GITO.

- Offermann, P., C. Schröpfer, et al. (2007). SOA: The IT-Architecture behind Service-Orientation. In: MDD, SOA und IT-Management. U. Steffens, J. S. Addicks und N. Streekmann. Berlin, GITO: 1-11.
- Ogden, C. K. and I. A. Richards (1960). The Meaning of Meaning: a study of the influence of language upon thought and of the science of symbolis. London, Routledge & Kegan Paul.
- Ohne Autor (1996). OMG Common Facilities RFP-4 - Common Business Objects and Business Object Facility. OMG TC Document CF/96-01-04, Object Management Group.
- OMG (2011). Business Process Model and Notation (BPMN) - Version 2.0, Object Management Group.
- Ortega, R. E., J. W. Avery, et al. (2003). Search Query Autocompletion. U. S. P. Office. USA, Amazon.com Inc., Seattle, WA (US).
- Oundhakar, S., K. Sivashanmugam, et al. (2005). "Discovery of web services in a multi-ontology and federated registry environment." International Journal of Web Services Research 2(3): 1-32.
- Paolucci, M., T. Kawamura, et al. (2002). Semantic Matching of Web Services Capabilities. In: The Semantic Web — ISWC 2002. I. Horrocks und J. Hendler, Springer Berlin / Heidelberg. 2342: 333-347.
- Papazoglou, M. P., P. Traverso, et al. (2006). Service-Oriented Computing: A Research Roadmap. Service Oriented Computing (SOC), Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany.
- Papazoglou, M. P. and W.-J. van den Heuvel (2006). "Service-Oriented Design and Development Methodology." International Journal of Web Engineering and Technology 2(4): 412-442.
- Pedrinaci, C. and J. Domingue (2007). "Towards an Ontology for Process Monitoring and Mining." Proceedings of Workshops on Semantic Business Process and Product Lifecycle Management (SBPM) at the 4th European Semantic Web Conference (ESWC) 251: 76-87.
- Pedrinaci, C., J. Domingue, et al. (2008). "Semantic Business Process Management: Scaling Up the Management of Business Processes." Proceedings of the IEEE International Conference on Semantic Computing: 546-553.
- Petri, C. A. (1962). Kommunikation mit Automaten. Institut für Instrumentelle Mathematik, Technische Hochschule Bonn. **Dissertation**.
- Polleres, A., H. Lausen, et al. (2006). Semantische Beschreibung von Web Services. In: Semantic Web. T. Pellegrini und A. Blumauer, Springer Berlin Heidelberg: 505-524.

- Porter, M. F. (1993). "An algorithm for suffix stripping." Program – Automated Library and Information Systems 14(3): 130-137.
- Prud'hommeaux, E. and A. Seaborne. (2008). "SPARQL Query Language for RDF." W3C Recommendation Retrieved 22.03.2011, from <http://www.w3.org/TR/rdf-sparql-query/>.
- Rahm, E. and P. A. Bernstein (2001). "A survey of approaches to automatic schema matching." The VLDB Journal 10(4): 334-350.
- Rahm, E., H.-H. Do, et al. (2004). "Matching large XML schemas." ACM SIGMOD Record 33(4): 26-31.
- Rake, J., O. Holschke, et al. (2009). Enhancing Semantic Service Discovery in Heterogeneous Environments
- Business Information Systems. In: Lecture Notes in Business Information Processing. W. Aalst, J. Mylopoulos, M. Rosemann, M. J. Shaw und C. Szyperski, Springer Berlin Heidelberg. 21: 205-216.
- Regev, G., I. Bider, et al. (2007). "Defining Business Process Flexibility with the Help of Invariants." Software Process Improvement and Practice 12: 65-79.
- Resnik, P. (1995). "Using information content to evaluate semantic similarity in a taxonomy." Proceedings of the 14th International Conference Artificial Intelligence (IJCAI), Montréal: 448-453.
- Roman, D., U. Keller, et al. (2005). "Web service modeling ontology." Applied Ontology 1(1): 77-106.
- Scaife, M. and Y. Rogers (1996). "External cognition: how do graphical representations work?" International Journal of Human-Computer Studies 45(2): 185-213.
- Scheer, A.-W. (1990). Wirtschaftsinformatik – Informationssysteme im Industriebetrieb. Berlin et al., Springer.
- Schmidt, A., K. Hüner, et al. (2010). Fallstudie Deutsche Telekom AG – Einheitliche Datenarchitektur als Grundlage für unternehmensweites Datenqualitätsmanagement. St. Gallen, Universität St. Gallen, Institut für Wirtschaftsinformatik.
- Schröpfer, C. (2009). Das SOA-Management-Framework. Lehrstuhl Systemanalyse und EDV. Berlin, Berlin Institute of Technology. **Ph.D. Thesis**.
- Schröpfer, C., M. Binshtok, et al. (2009). Introducing Preferences over NFPs into Service Selection in SOA. In: Service-Oriented Computing - ICSOC 2007 Workshops. E. Di Nitto und M. Ripeanu, Springer Berlin / Heidelberg. 4907: 68-79.
- Seidlmeier, H. (2010). Prozessmodellierung mit ARIS: Eine beispielorientierte Einführung für Studium und Praxis, Vieweg+Teubner Verlag.

- Shangguan, Z., Z. Gao, et al. (2008). Ontology-Based Process Modeling Using eTOM and ITIL. In: Research and Practical Issues of Enterprise Information Systems II. L. Xu, A. Tjoa und S. Chaudhry, Springer Boston. 255: 1001-1010.
- Shvaiko, P. and J. Euzenat (2008). Ten Challenges for Ontology Matching. In: On the Move to Meaningful Internet Systems: OTM 2008, Springer Berlin / Heidelberg. 5332: 1164-1182.
- Si, Z. and S. Zhefeng (2009). "Using WSMO to Enable Mediation of Heterogeneous Services and Semi-automation of Service Discovery and Execution in Semantic SOA." Proceedings of the International Conference on Web Information Systems and Mining 2009: 638-642.
- Sinkkila, R., E. Makela, et al. (2008). "Combining Context Navigation with Semantic Autocompletion to Solve Problems in Concept Selection." First International Workshop on Semantic Metadata Management and Applications, SeMMA 2008, Located at the Fifth European Semantic Web Conference (ESWC 2008) 346: 61-68.
- Sinkkilä, R., E. Mäkelä, et al. (2008). "Combining context navigation with semantic autocompletion to solve problems in concept selection." Proceedings of the 1st International Workshop on the Semantic Metadata Management and Applications (SEMMA 2008), 5th European Semantic Web Conference (ESWC2008), Tenerife, Spain, June 1-5, 2008.: 1-5.
- Sirin, E., B. Parsia, et al. (2007). "Pellet: A practical OWL-DL reasoner." Web Semantics: Science, Services and Agents on the World Wide Web 5(2): 51-53.
- Sivashanmugam, K., A. Sheth, et al. (2003). "Adding Semantics to Web Services Standards." Proceedings of the International Conference on Web Services: 395-401.
- Stachowiak, H. (1973). Allgemeine Modelltheorie, Springer-Verlag.
- Stoilos, G., G. Stamou, et al. (2005). A String Metric for Ontology Alignment. In: The Semantic Web – ISWC 2005. Y. Gil, E. Motta, V. Benjamins und M. Musen, Springer Berlin / Heidelberg. 3729: 624-637.
- Sycara, K., M. Paolucci, et al. (2003). "Automated discovery, interaction and composition of semantic web services." Web Semantics: Science, Services and Agents on the World Wide Web 1(1): 27-46.
- Thatte, S. (2001). XLANG - Web Services for Business Process Design, Microsoft.
- Thomas, O. (2006). Management von Referenzmodellen: Entwurf und Realisierung eines Informationssystems zur Entwicklung und Anwendung von Referenzmodellen. Berlin, Logos Verlag.
- Thomas, O. and M. Fellmann (2009). "Semantische Prozessmodellierung – Konzeption und informationstechnische Unterstützung einer ontologiebasierten Reprä-

- sentation von Geschäftsprozessen." WIRTSCHAFTSINFORMATIK 51(6): 506-518.
- Tsarkov, D. and I. Horrocks (2006). FaCT++ Description Logic Reasoner: System Description. In: Automated Reasoning. U. Furbach und N. Shankar, Springer Berlin / Heidelberg, 4130: 292-297.
- Tversky, A. (1977). "Features of similarity." Psychological Review 84(4): 327-352.
- Usanavasin, S., S. Takada, et al. (2005). Semantic Web Services Discovery in Multi-ontology Environment. In: On the Move to Meaningful Internet Systems 2005: OTM 2005 Workshops. R. Meersman, Z. Tari und P. Herrero, Springer Berlin / Heidelberg, 3762: 59-68.
- Uschold, M. and M. Gruninger (2004). "Ontologies and semantics for seamless connectivity." ACM SIGMOD Record 33(4): 58-64.
- Valtchev, P. (1999). Construction automatique de taxonomies pour l'aide à la représentation de connaissances par objets - Automatic taxonomy design for object-based knowledge representations. Grenoble, France, J. Fourier University.
- van der Aalst, W., A. ter Hofstede, et al. (2003). Business Process Management: A Survey. In: Business Process Management. A. ter Hofstede und M. Weske, Springer Berlin / Heidelberg, 2678: 1-12.
- van der Aalst, W. M. P. (1999). "Formalization and verification of event-driven process chains." Information and Software Technology 41(10): 639-650.
- van der Aalst, W. M. P. and A. H. M. ter Hofstede (2005). "YAWL: yet another workflow language." Information Systems 30(4): 245-275.
- Van der Aalst, W. M. P. and K. M. Van Hee (1996). "Business process redesign: a Petri-net-based approach." Computers in Industry 29(1-2): 15-26.
- van Dongen, B. F. and W. M. P. van der Aalst (2005). "A meta model for process mining data." Proceeding of the CAISE 2005 Workshops 161: 309-320.
- van Lessen, T., J. Nitzsche, et al. (2009). An Execution Engine for Semantic Business Processes. In: Service-Oriented Computing - ICSOC 2007 Workshops. E. Di Nitto und M. Ripeanu, Springer Berlin / Heidelberg, 4907: 200-211.
- Verma, K., K. Sivashanmugam, et al. (2005). "METEOR-S WSDI: A Scalable P2P Infrastructure of Registries for Semantic Publication and Discovery of Web Services." Information Technology and Management 6(1): 17-39.
- Vinoski, S. (2005). "Old measures for new services." IEEE Internet Computing 9(6): 72-74.

- Voß, M., A. Hess, et al. (2006). "Towards a framework for large scale quality architecture." Perspectives in Software Quality - Short Papers of the 2nd International Conference on the Quality of Software Architectures (QoSA): 53-59.
- W3C OWL Working Group. (2009). "OWL 2 Web Ontology Language " W3C Recommendation Retrieved 15.03.2011, from <http://www.w3.org/TR/2009/REC-owl2-overview-20091027/>.
- Wang, P. and B. Xu (2008a). "Debugging Ontology Mapping: A Static Approach." Computing and Informatics 27(1): 21-36.
- Wang, P. and B. Xu (2008b). "Lily: ontology alignment results for OAEI 2008." Proceedings of the Ontology Matching Workshop at the International Semantic Web Conference (ISWC 2008): 167-175.
- Wang, P. and B. Xu (2009). "Lily: ontology alignment results for OAEI 2009." Proceedings of the 4th International Workshop on Ontology Matching (OM-2009) 551: 186-192.
- Wang, P., B. Xu, et al. (2010). "Extracting Semantic Subgraphs to Capture the Real Meanings of Ontology Elements." Tsinghua Science & Technology 15(6): 724-733.
- Wang, P., Y. Zhou, et al. (2011). Matching Large Ontologies Based on Reduction Anchors. The Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI 2011). Barcelona, Catalonia, Spain.
- Wang, X., T. Vitvar, et al. (2006). A QoS-Aware Selection Model for Semantic Web Services. In: Service-Oriented Computing – ICSOC 2006, A. Dan und W. Lamersdorf, Springer Berlin / Heidelberg, 4294: 390-401.
- Weber, I., I. Markovic, et al. (2007). A Conceptual Framework for Composition in Business Process Management. In: Business Information Systems. W. Abramowicz, Springer Berlin / Heidelberg, 4439: 54-66.
- Weske, M. (1999). "Business-Objekte: Konzepte, Architekturen, Standards." Wirtschaftsinformatik 41(1): 4-11.
- Wetzstein, B., Z. Ma, et al. (2007). "Semantic business process management: A lifecycle based requirements analysis." Proceedings of Workshops on Semantic Business Process and Product Lifecycle Management (SBPM) at the 4th European Semantic Web Conference (ESWC) 251: 1-12.
- Winkler, W. E. (1999). The state of record linkage and current research problems, Statistical Research Division, US Census Bureau.
- Workflow Management Coalition (1999). Terminology & Glossary. Winchester, Workflow Management Coalition.
- World Wide Web Consortium. (2004). "Web Services Architecture." Retrieved 6/12/2006, from <http://www.w3.org/TR/ws-arch/>.

- Wu, Z. and M. Palmer (1994). "Verbs semantics and lexical selection." Proceedings of the 32nd Annual Meetings of the Association for Computational Linguistics (ACL), Las Cruces, USA: 133-138.
- Wynn, M. T., H. M. W. Verbeek, et al. (2009). "Business process verification - finally a reality!" Business Process Management Journal 15(1): 74-92.
- Yuzhong, Q., H. Wei, et al. (2006). Constructing virtual documents for ontology matching. Proceedings of the 15th international conference on World Wide Web. Edinburgh, Scotland, ACM: 23-31.
- Zaremba, M., M. Moran, et al. (2009). Instance-based Service Discovery with WSMO/WSML and WSMX. In: Semantic Web Services Challenge. C. Petrie, T. Margaria, H. Lausen und M. Zaremba, Springer US. 8: 169-183.
- Ziemann, J. and J. Mendling (2005). EPC-based modelling of BPEL processes: a pragmatic transformation approach. Proceedings of the 7th International Conference "Modern Information Technology in the Innovation Processes of the Industrial Enterprises". Genova, Italy.
- Zimmermann, A., M. Krötzsch, et al. (2006). "Formalizing ontology alignment and its operations with category theory." Proceeding of the 2006 conference on Formal Ontology in Information Systems: 277-288.
- zur Mühlen, M. (2004). Workflow-based Process Controlling - Foundation, Design, and Implementation of Workflow-driven Process Information Systems, Logos Verlag Berlin.

Anhang A

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
  <!ENTITY example "http://example.org/ontologien/example.owl#" >
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>
<rdf:RDF xmlns="http://example.org/ontologien/example.owl#"
  xml:base="http://example.org/ontologien/example.owl"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:example="http://example.org/ontologien/example.owl#"
  <owl:Ontology rdf:about="http://example.org/ontologien/example.owl"/>

  <!-- http://example.org/ontologien/example.owl#hasStatus -->
  <owl:ObjectProperty rdf:about="&example;hasStatus">
    <rdfs:domain rdf:resource="&example;Geschaeftsobjekt"/>
    <rdfs:range rdf:resource="&example;Status"/>
  </owl:ObjectProperty>

  <!-- http://example.org/ontologien/example.owl#Auftrag -->
  <owl:Class rdf:about="&example;Auftrag">
    <rdfs:subClassOf rdf:resource="&example;Geschaeftsobjekt"/>
  </owl:Class>

  <!-- http://example.org/ontologien/example.owl#Fertigungsauftrag -->
  <owl:Class rdf:about="&example;Fertigungsauftrag">
    <rdfs:subClassOf rdf:resource="&example;Auftrag"/>
  </owl:Class>

  <!-- http://example.org/ontologien/example.owl#Geschaeftsobjekt -->
  <owl:Class rdf:about="&example;Geschaeftsobjekt"/>

  <!-- http://example.org/ontologien/example.owl#Kundenrechnung -->
  <owl:Class rdf:about="&example;Kundenrechnung">
    <rdfs:subClassOf rdf:resource="&example;Rechnung"/>
  </owl:Class>

  <!-- http://example.org/ontologien/example.owl#Pruefung -->
  <owl:Class rdf:about="&example;Pruefung">
    <rdfs:subClassOf rdf:resource="&example;Geschaeftsobjekt"/>
  </owl:Class>

  <!-- http://example.org/ontologien/example.owl#Rechnung -->
  <owl:Class rdf:about="&example;Rechnung">
    <rdfs:subClassOf rdf:resource="&example;Geschaeftsobjekt"/>
  </owl:Class>

  <!-- http://example.org/ontologien/example.owl#Serviceauftrag -->
  <owl:Class rdf:about="&example;Serviceauftrag">
```

```

        <rdfs:subClassOf rdf:resource="&example;Auftrag"/>
    </owl:Class>

    <!-- http://example.org/ontologien/example.owl#Status -->
    <owl:Class rdf:about="&example;Status"/>

    <!-- http://example.org/ontologien/example.owl#Zahlungspruefung -->
    <owl:Class rdf:about="&example;Zahlungspruefung">
        <rdfs:subClassOf rdf:resource="&example;Pruefung"/>
    </owl:Class>

    <!-- http://example.org/ontologien/example.owl#abgeschlossen -->
    <owl:Class rdf:about="&example;abgeschlossen">
        <rdfs:subClassOf rdf:resource="&example;Status"/>
    </owl:Class>

    <!-- http://example.org/ontologien/example.owl#angesetzt -->
    <owl:Class rdf:about="&example;angesetzt">
        <rdfs:subClassOf rdf:resource="&example;Status"/>
    </owl:Class>

    <!-- http://example.org/ontologien/example.owl#gesendet -->
    <owl:Class rdf:about="&example;gesendet">
        <rdfs:subClassOf rdf:resource="&example;Status"/>
    </owl:Class>

    <!-- http://example.org/ontologien/example.owl#ueberprueft -->
    <owl:Class rdf:about="&example;ueberprueft">
        <rdfs:subClassOf rdf:resource="&example;Status"/>
    </owl:Class>
</rdf:RDF>

```

Abbildung 90: Beispielcode für eine Ontologie zur Darstellung des Abrechnungsservice als OWL Ontology