Engineering ontologischer Modelle in der Automatisierungstechnik

Der Fakultät für Maschinenbau
der Helmut-Schmidt-Universität / Universität der Bundeswehr Hamburg
zur Erlangung des akademischen Grades eines Doktor-Ingenieurs (Dr.-Ing.)
genehmigte

DISSERTATION

von Constantin Frank Hildebrandt, M. Sc.

aus Wernigerode

Hamburg 2021

für

Clara Maria Hildebrandt

- sapere aude -

Vorwort des Autors

Die vorliegende Arbeit entstand im Rahmen meiner Tätigkeit als wissenschaftlicher Mitarbeiter an der Professur für Automatisierungstechnik der Helmut-Schmidt-Universität / Universität der Bundeswehr Hamburg in der Zeit von Mai 2016 bis Mai 2020. Nachfolgend möchte ich all den Menschen meinen Dank aussprechen, die maßgeblich zum Gelingen dieser Arbeit beigetragen haben.

Mein besonderer Dank gilt Herrn Prof. Dr.-Ing. Alexander Fay, der diese Arbeit durch zahlreiche Diskussionen, Anregungen, und insbesondere durch das entgegengebrachte Vertrauen in meine Arbeit und meine Fähigkeiten erst möglich gemacht hat. Ohne die gewährte Freiheit, meine Ideen in eine Vielzahl von Projekten einfließen lassen zu können, wäre diese Arbeit nicht möglich gewesen. Ebenso danke ich Prof. Dr.-Ing. Christian Diedrich für das Interesse an der Arbeit und die Erstellung des Zweitgutachtens. Prof. Dr.-Ing. Jens P. Wulfsberg danke ich für den Vorsitz des Promotionsverfahrens.

Ich bedanke mich bei den ehemaligen Partnern der Forschungsprojekte CrESt und SemAnz40 für die vielen konstruktiven Diskussionen. Des Weiteren möchte ich den Kollegen in den drei Unternehmen herzlich danken, in welchen ich meine entwickelten Methoden im industriellen Kontext anwenden konnte, welche ich aus Gründen der Geheimhaltung jedoch nicht namentlich nennen darf. Ohne die detailreichen Diskussionen und das kritische Feedback wäre die Qualität der vorliegenden Arbeit nicht erreichbar gewesen.

Gern blicke ich auch auf die Zeit am Institut zurück. Die Zusammenarbeit mit derart innovativen, ideenreichen, offenen und inspirierenden Menschen hat mir sehr viel Freude bereitet und mich sehr geprägt. Insbesondere möchte ich mich bei Herrn Aljosha Köcher, Herrn Dr.-Ing. André Scholz, Frau Birte Caesar, Herrn Felix Gehlhoff, Herrn Alexander Hayward und Herrn Sebastian Törsleff für die vielen gemeinsamen Gespräche bedanken, welche maßgebliche zum Gelingen dieser Arbeit beigetragen haben. Auch allen Studenten, welche durch ihre studentischen Arbeiten zu dieser Dissertation beigetragen haben, sei an dieser Stelle gedankt.

Mein besonderer Dank gilt meinen Freunden und vor allem meiner Familie, die während meiner Forschungstätigkeit oft auf mich verzichten musste. Ganz besonders möchte ich mich bei meiner Frau Rebecca bedanken. Ohne dein Verständnis, deinen Rückhalt und deine liebevolle Unterstützung, wäre die Erstellung dieser Arbeit nicht möglich gewesen.

Norderstedt, 19.04.2021

Constantin Frank Hildebrandt

Inhaltsverzeichnis

Begriffsverzeichnis **IX**

Verzeichnis der Abkürzungen **X**

Verzeichnis verwendeter Formelzeichen **XI**

Kurzfassung **XII**

Abstract **XIII**

I. Grundlagen und Zielbestimmung **1**

I-1. Einleitung . . . 1
I-1.1. Motivation und Problemstellung . . . 1
I-1.2. Forschungsmethodik . . . 3
I-1.3. Forschungsziel und Struktur der Arbeit . . . 5
I-2. Grundlagen und Begriffsbestimmung . . . 6
I-2.1. Grundlagen zum Engineering automatisierter Produktionssysteme . . . 6
I-2.1.1. Terminologische Grundlagen zum Engineering automatisierter Produktionssysteme . . . 7
I-2.1.2. Daten, Informationen und Wissen im Kontext des Engineerings 9
I-2.1.3. Cyber-Physische Systeme . . . 13
I-2.2. Ontologische Grundlagen . . . 18
I-2.2.1. Terminologische Grundlagen zu Ontologien . . . 18
I-2.2.2. Nutzen und Anwendungsfälle von Ontologien . . . 20
I-2.2.3. Ontologisches Spektrum und Arten von Ontologien . . . 22
I-3. Stand der Wissenschaft zur Nutzung von Ontologien im Engineering automatisierter Produktionssysteme . . . 24
I-3.1. Kriterien zur Analyse des Stands der Wissenschaft . . . 24
I-3.2. Anwendungsfälle von Ontologien im Engineering automatisierter Produktionssysteme . . . 27
I-3.3. Analyse und Diskussion des Stands der Wissenschaft zu Ontologien im Engineering automatisierter Produktionssysteme . . . 32
I-3.3.1. Analyse des Stands der Wissenschaft . . . 32
I-3.3.2. Diskussion der Analyseergebnisse . . . 34
I-4. Lösungsansatz, Forschungsteilziele und Anforderungen . . . 36
I-4.1. Methodischer Überblick . . . 36
I-4.2. Forschungsteilziel I: Ontologie-Anforderungserhebung . . . 37
I-4.3. Forschungsteilziel II: Ontologie-Entwicklung TBox . . . 39
I-4.4. Forschungsteilziel III: Ontologie-Entwicklung ABox . . . 40

II. Ontologie-Anforderungserhebung 43
II-1. Grundlagen zur modellbasierten Anforderungserhebung . . . 43
II-1.1. System und Systemkontext . . . 44
II-1.2. Perspektiven des operationellen Kontextes . . . 44
II-1.3. Modellierungssprachen im Kontext der modellbasierten Entwicklung . . 46
II-1.3.1. Unified Modeling Language . . . 46
II-1.3.2. System Modeling Language (SysML) . . . 47
II-1.3.3. Profile im Kontext von UML . . . 47
II-1.4. Metamodelle mit Relevanz für die Anforderungserhebung . . . 48
II-1.4.1. Funktionsmetamodell nach Ludewig et al. . . . 48
II-1.4.2. Funktionsmetamodell nach Lamm und Weilkiens . . . 49
II-1.4.3. Funktionsmetamodell nach Scholz . . . 49
II-1.4.4. Strukturmetamodell nach Regnat et al. . . . 49
II-1.4.5. Kontextmetamodell nach Daun et al. . . . 50
II-2. Stand der Wissenschaft zur Anforderungserhebung an Ontologien . . . 50
II-2.1. Ontologie-Anforderungserhebung nach Uschold und Gruninger . . . 51
II-2.2. Ontologie-Anforderungserhebung nach Suárez-Figueroa et al. . . . 51
II-2.3. Ontologie-Anforderungserhebung nach Nicola et al. . . . 52
II-2.4. Ontologie-Anforderungserhebung nach Peroni . . . 53
II-2.5. Bewertung und Auswahl der Ontologie-Anforderungserhebungsmethoden 54
II-3. Metamodell und UML-Profile zur modellbasierten Anforderungserhebung . . . 56
II-3.1. Begriffliche Definitionen und Metamodell . . . 57
II-3.1.1. Beschreibung von System und Kontext . . . 57
II-3.1.2. Funktionale Beschreibungen . . . 59
II-3.1.3. Beschreibung ontologischer Anforderungen . . . 61
II-3.2. Profildiagramme . . . 63
II-3.2.1. Profil für ontologische Anforderungen . . . 63
II-3.2.2. Funktionsstrukturprofil . . . 64
II-3.2.3. Funktionsverhaltensprofil . . . 64
II-3.2.4. Systemgruppenstrukturprofil . . . 65
II-4. Modellierungsmethodik . . . 65
II-4.1. Methodischer Überblick . . . 65
II-4.2. Definition ontologischer Ziele . . . 67
II-4.3. Definition von Systemanforderungen . . . 69
II-4.3.1. Modellierung von Anwendungsfällen . . . 69
II-4.3.2. Modellierung von Funktionsstrukturen . . . 70
II-4.3.3. Modellierung des Funktionsverhaltens . . . 70
II-4.3.4. Modellierung der Systemgruppenstruktur . . . 71
II-4.4. Definition ontologischer Anforderungen . . . 72
II-4.4.1. Erhebung funktionaler und nicht-funktionaler ontologischer Anforderungen . . . 73
II-4.4.2. Strukturierung, Priorisierung und Verfolgung ontologischer Anforderungen . . . 75

III. Ontologie-Entwicklung TBox **77**
III-1. Grundlagen zu Semantic Web Technologien . . . 77
III-1.1. RDF - Resource Description Framework . . . 77
III-1.2. RDFS - Resource Description Framework Schema . . . 79
III-1.3. OWL - Web Ontology Language . . . 81
III-1.4. SPARQL - SPARQL Protocol And RDF Query Language . . . 83
III-2. Stand der Wissenschaft zur Ontologie-Entwicklung (TBox) . . . 84
III-2.1. Methodik nach Presutti et al. . . . 85
III-2.2. Methodik nach Pinto et al. . . . 85
III-2.3. Methodik nach Peroni . . . 85
III-2.4. Methodik nach de Nicola et al. . . . 86
III-2.5. Methodik nach Braun et al. . . . 87
III-2.6. Methodik nach Halilaj . . . 87
III-2.7. Methodik nach de Almeida Falbo . . . 88
III-2.8. Bewertung und Auswahl der Methoden zur Ontologie-Entwicklung (TBox) . . . 89
III-3. Methodik zur Domänenexperten-zentrischen Entwicklung von Ontologien . . . 91
III-3.1. Ontologie-Design-Muster . . . 91
III-3.1.1. Charakteristika von Ontologie-Design-Mustern und Übertragung auf die Automatisierungstechnik . . . 91
III-3.1.2. Methodischer Überblick zur TBox-Entwicklung basierend auf Ontologie-Design-Mustern . . . 95
III-3.2. Ontologie-Erstellung (Lightweight-TBox) . . . 97
III-3.2.1. Suche und Bewertung relevanter Informationsressourcen . . . 98
III-3.2.2. Modellierung von Lightweight-Ontologie-Design-Mustern . . . 103
III-3.3. Ontologie-Erstellung (Heavyweight-TBox) . . . 111
III-3.3.1. Modellierung der Heavyweight-Ontologie-Design-Muster und der Alignment-Ontologie . . . 111
III-3.3.2. Verifikation der erstellten TBox . . . 123
III-3.3.3. Dissemination der Ontologie-Design-Muster . . . 124
III-4. Wiederverwendbare Ontologie-Design-Muster . . . 126
III-4.1. Modellierung von Systemfunktionen basierend auf der VDI 3682 . . . 127
III-4.2. Modellierung des Systemverhaltens basierend auf der ISA 88 . . . 127
III-4.3. Modellierung von Systemstrukturen basierend auf der VDI 2206 . . . 129
III-4.4. Modellierung von Datenelementen basierend auf der DIN EN 61360 . . . 130
III-4.5. Alignment-Ontologie . . . 132

IV. Ontologie-Entwicklung ABox **135**
IV-1. Grundlagen zur ABox-Modellierung . . . 135
IV-1.1. Verfahren zur Erstellung der ABox . . . 136
IV-1.1.1. Manuelle Modellierung der ABox . . . 136
IV-1.1.2. Automatisierte Erstellung der ABox . . . 136
IV-1.2. Artefakte und deren Eignung für automatisierte Verfahren . . . 139
IV-1.2.1. Engineering- und Laufzeit-Artefakte . . . 139
IV-1.2.2. Qualität, Offenheit und Reifegrade . . . 140

IV-2. Stand der Wissenschaft zur ABox-Modellierung . . . 142
IV-2.1. Kriterien zur Analyse des Stands der Wissenschaft . . . 142
IV-2.2. Verfahren für die automatisierte ABox-Erstellung . . . 143
IV-2.2.1. Verfahren für Daten in relationalen Formaten . . . 143
IV-2.2.2. CAEX-basierte Verfahren . . . 144
IV-2.2.3. Formatübergreifende Verfahren . . . 144
IV-2.3. Zusammenfassung der Analyse des Stands der Wissenschaft . . . 145
IV-3. Methodik zum werkzeugunterstützten ABox-Engineering . . . 146
IV-3.1. Methodischer Überblick . . . 146
IV-3.1.1. Charakterisierung des ABox-Entwicklungskonzptes . . . 146
IV-3.1.2. Methodischer Überblick zur ABox-Entwicklungsmethodik . . . 148
IV-3.2. Herleitung der Modellierungsstrategie . . . 149
IV-3.2.1. Analyse relevanter RDF-Strukturen . . . 150
IV-3.2.2. Analyse existierender Artefakte . . . 153
IV-3.2.3. Definition der Modellierungsstrategie . . . 161
IV-3.3. Werkzeug-Entwicklung zur semi-automatischen ABox-Entwicklung . . . 162
IV-3.3.1. Architektur des Werkzeugs . . . 162
IV-3.3.2. Methodik der Werkzeugentwicklung . . . 164
IV-3.3.3. Modellierung von Entitäten . . . 166
IV-3.3.4. Modellierung assertionaler Muster . . . 167
IV-3.3.5. Implementierung im Werkzeug LIOnS . . . 171
IV-3.4. Modellierungsmethodik im Kontext des Engineerings fertigungstechnischer Systeme . . . 173
IV-3.4.1. Bestimmung des Workflows . . . 175
IV-3.4.2. Modellverknüpfung . . . 175
IV-3.4.3. Modellerstellung und finale Verifikation . . . 177

V. Evaluation **178**
V-1. Methodischer Überblick . . . 178
V-1.1. Evaluationsmethodik . . . 178
V-1.2. Iterationen und zeitlicher Verlauf der Evaluation . . . 180
V-2. Darstellung der Fallstudien . . . 181
V-2.1. Industrielle Fallstudie „Wartungsassistenz“ . . . 181
V-2.1.1. Beschreibung des Fallbeispiels . . . 181
V-2.1.2. Durchführung und Ergebnisse der Fallstudien . . . 182
V-2.2. Industrielle Fallstudie „automatisierte Fähigkeitsabfrage“ . . . 191
V-2.2.1. Beschreibung des Fallbeispiels . . . 191
V-2.2.2. Durchführung der Fallstudie . . . 191
V-2.3. Industrielle Fallstudie „Prozessanalyse“ . . . 196
V-2.3.1. Beschreibung des Fallbeispiels . . . 197
V-2.3.2. Durchführung der Fallstudien . . . 198
V-3. Übergreifende Bewertung der Ergebnisse . . . 212
V-3.1. Forschungsteilzielbezogene Bewertung . . . 212
V-3.1.1. Forschungsteilziel I . . . 212
V-3.1.2. Forschungsteilziel II . . . 213
V-3.1.3. Forschungsteilziel III . . . 215

V-3.2. Forschungszielbezogene Bewertung 216

VI. Zusammenfassung und Ausblick **219**
VI-1. Zusammenfassung 219
VI-2. Ausblick 221

Anhang **223**
A. Tabellen 223
B. Abbildungen 227
C. Listings 232

Literaturverzeichnis **249**
Wissenschaftliche Publikationen 249
Normen und Richtlinien 262
Publikationen des Autors 263
Studentische Arbeiten 265
Internetquellen und Software 265

Begriffsverzeichnis

Daten	Objektive Fakten, die ohne Zusammenhang und weitere Hintergründe nicht deutbar und als „Rohmaterial“ zu verstehen sind. [VDI 5610-2]
Engineering	„Das Engineering ist die systematische Anwendung von Kenntnissen über physikalische Gesetzmäßigkeiten zur Konzeption, Erschaffung und Verbesserung von Anlagen. Das Engineering erfordert das Zusammenwirken verschiedener Gewerke, z. B. der Prozesstechnik, des Maschinenbaus, der Elektrotechnik und der Automatisierungstechnik.“ [VDI 3695-1]
Engineering-Werkzeug	Engineering-Werkzeuge repräsentieren alle bei der Erfüllung der Engineering-Aufgabe notwendigen technischen Arbeitsmittel, sowohl auf Seiten der Software als auch auf Seiten der Hardware. Software-Werkzeuge unterstützen den Mitarbeiter beim Engineering-Prozess durch optimierte Prozessabläufe, vordefinierte Vorlagen, einfache Bedienung und Verwendung standardisierter Beschreibungsmittel. [VDI 3695-1]
Engineering-Artefakt	Alle materiellen und immateriellen Projektergebnisse. [VDI 3695-1]
Information	Strukturierte Daten mit Relevanz und Zweck, die semantisch in einen Kontext gebracht, kategorisiert, kalkuliert und korrigiert werden können. [VDI 5610-2]
Laufzeit-Artefakt	Als Laufzeit-Artefakte werden Daten eines Systems aus dem Betrieb bezeichnet, wobei hierzu das entsprechende Schema (z.B. OPC UA Informationsmodell als XML, oder SQL Schema) und die dazugehörigen Daten der Datenquelle zu einem spezifischen Zeitpunkt gehören.
Materialisierung	Im Rahmen einer Materialisierung wird die assertionale Box einer Ontologie automatisiert aus einem Artefakt erzeugt und dauerhaft verfügbar gemacht.
Virtualisierung	Im Rahmen einer Virtualisierung wird die assertionale Box einer Ontologie automatisiert aus einem Artefakt erzeugt und nur für die Dauer einer Anfrage verfügbar gemacht.
Wissen	Vernetzte Information, die in die Lage versetzt, Vergleiche anzustellen, Verknüpfungen herzustellen und Entscheidungen zu treffen. [VDI 5610-2]

Verzeichnis der Abkürzungen

ABox	Assertionale Box
API	Application Programming Interface
CSV	Comma-separated values
CQ	Competency Question
DIN	Deutsches Institut für Normung e. V.
DIW	Daten, Information und Wissen
ISA	Instrumentations, Systems, and Automation Society
ISO	International Organization for Standardization
JSON	Java Script Object Notation
MES	Manufacturing Execution System
ODM	Ontologie-Design-Muster
OWL	Ontology Web Language
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
SysML	Systems Modeling Language
SPARQL	SPARQL Protocol And RDF Query Language
SPS	Speicherprogrammierbare Steuerung
SQL	Structured Query Language
SUD	System under Development
TBox	Terminologische Box
UML	Unified Modelling Language
VDE	Verband der Elektrotechnik Elektronik Informationstechnik e. V.
VDI	Verein Deutscher Ingenieure e. V.
W3C	World Wide Web Consortium
XLS	Dateiformat von Microsoft Excel Tabellen
XML	Extensible Markup Language

Verzeichnis verwendeter Formelzeichen

α	Die Dynamik eines Daten-, Informations- oder Wissensobjektes
β	Rollenbesetzungen pro zeitlicher Bezugseinheit
B	Menge der Begriffe
B_A	Begriffliche Abdeckung
B_P	Begriffliche Präzision
B_{IR}	Menge der Begriffe der Informationsressource
B_G	Menge der Begriffe des Anforderungsglossars
B_O	Overhead an nicht relevanten Begriffen
γ	Vergleichsfaktor für Artefaktbewertung
Δ	Vollständigkeit der Anforderungserfüllung
E	Menge der Entitäten
F_{IR}	Formalismus einer Informationsressource
$\Im$	Semantische Interpretation
IRU	Unabhängigkeit einer Informationsressource von anderen Informationsressourcen
$IRAF$	Zielbezogene Anpassungsfähigkeit einer Informationsressource
IRK	Voraussichtliche Implementierungskomplexität einer Informationsressource
Λ	Menge der assertionalen Muster
$\Re$	Menge der Relationen
R	Menge der Kompetenzfragen und Antworten
SE_{IR}	Semantische Eignung einer Informationsressource
T	Menge der Tripel
T_{TBox}	Menge der Tripel der TBox
$T_{ABox}(t)$	Menge der Tripel der ABox in Abhängigkeit von der Zeit t
τ	Modellierungsbedarf
Q	Menge der Abfragen
Z	zeitlicher Aufwand

Kurzfassung

Mit einer steigenden Anzahl an Software-Funktionalitäten im Kontext eines Produktionssystems und einer zunehmenden Menge an Daten, Informationen und Wissen über Produktionssysteme steigt der Druck hin zu einer effizienten Nutzung dieser Daten, Informationen und Wissensbestandteile. Ontologien sind ein geeignetes Mittel, um unterschiedlichste Aspekte im Kontext eines Produktionssystems zu formalisieren und somit für entsprechende Systeme nutzbar zu machen. Die Entwicklung einer Ontologie ist jedoch eine nicht-triviale Aufgabe, welche einen hohen Aufwand unter Einbeziehung unterschiedlicher Stakeholder erfordert. Es ist daher von essentieller Bedeutung, dass diese hohen Aufwände über mehrere Projekte verteilt werden können, sodass sich langfristig eine gewisse Effizienz im Umgang mit den relevanten Technologien einstellen und deren Nutzung auf diese Weise einen Mehrwert im industriellen Umfeld schaffen kann. Die vorliegende Arbeit befasste sich daher mit dem Ziel, eine Engineering-Methode zu entwickeln, welche durch geeignete Automatisierung und Strukturierung von Modellierungstätigkeiten sowie ein entsprechendes Wiederverwendungskonzept die notwendige Effizienz für eine Anwendung im industriellen Umfeld schafft. Zur Erreichung dieses Ziels wurden zunächst bestehende Ansätze im Wissensgebiet der Automatisierungstechnik untersucht, welche eine Ontologie für einen spezifischen Anwendungsfall entwickeln. Basierend auf den darin erkannten methodischen Defiziten wurden drei Forschungsteilziele definiert: Die Entwicklung von je einer Methode zur Anforderungserhebung, zur Entwicklung der terminologischen Box und zur Entwicklung der assertionalen Box der Ontologie. Die entwickelte Methodik zur Anforderungserhebung verbindet die Anforderungserhebung an eine Ontologie mit der Anforderungserhebung der sie nutzenden Applikation. Auf diese Weise werden nicht nur sehr spezifische Anforderungen an die Ontologie erhoben, sondern auch Synergien im Entwicklungsprozess ermöglicht. Die Methodik zur Entwicklung der terminologischen Box der Ontologie zielt auf eine möglichst hohe Wiederverwendung der entwickelten Ontologien ab und basiert daher auf der Erstellung und Nutzung von Ontologie-Design-Mustern, welche das Wissen einer Domäne, bspw. in Form eines Standards, formalisieren. Die Methodik zur Erstellung der assertionalen Box hingegen zielt auf eine möglichst effiziente Erstellung der Fakten, welche über ein Produktionssystem existieren, und berücksichtigt daher Verfahren zur automatisierten Überführung von Artefakten aus der Systementwicklung in die Ontologie, aber auch ein semi-automatisiertes Verfahren basierend auf Nutzereingaben, um die Unabhängigkeit von Artefakten der Systementwicklung zu gewährleisten. Für die Evaluation der entwickelten Methodiken wurden mehrere Fallstudien in Industrieunternehmen durchgeführt, in denen die Vorteilhaftigkeit der vorliegenden Methodiken sowie die Erreichung des Forschungsziels dargelegt werden konnten.

Abstract

Due to an increasing amount of software features as well as data, information and knowledge about an automated production system, there is an increasing need for an efficient use of data, information and knowledge about these systems. Ontologies provide a suitable technology to formalize different aspects of an automated production system and could potentially be the data, information and knowledge provider for different software functionalities. The development of an ontology, however, is a non-trivial task that potentially requires high efforts of different stakeholders. Therefore, an efficient ontology development procedure is crucial in order to have a positive cost-benefit trade off. The goal of this thesis is to develop an engineering method that is efficient enough to be used in industry, enabled by automation and structuring of relevant modelling activities. Furthermore, the developed approach should focus on a high reusability of developed ontological artefacts to decrease the development efforts with every further application of the method. In order to reach this goal, an analysis of existing approaches has taken place, that have developed an ontology for a use case in the automation domain. Based on the limitation of the used development procedures, three sub research goals were identified: the development of a Requirements Engineering method for ontologies, a method for building the terminological box of an ontology, and a method for building the assertional box of an ontology. The Requirements Engineering method combines the modelling of ontological requirements with requirement models of the application that uses the ontology. This results in solution-focused ontological requirements, as well as synergies in activities between yet distinct modelling areas. With respect to the method for building the terminological box of the ontology, a special emphasis was set on the reusability of developed ontological artefacts, enabled by Ontology Design Patterns that encapsulate the knowledge of a domain. The method for building the assertional box of the ontology allows a high efficiency in the building procedure due to a combination of automatic and semi-automatic ontology building procedures that can be set up specifically for a project. Concerning the evaluation of the proposed engineering method, three different case studies in industrial companies have taken place in which the achievement of the goal has been proven.

I. Grundlagen und Zielbestimmung

Der nachfolgende Teil I dieser Arbeit befasst sich mit der Einleitung in das Thema (siehe Abschnitt I-1), wobei das Thema dieser Dissertation motiviert und die Struktur dieser Arbeit erläutert wird. Der Abschnitt I-2 befasst sich dann mit einer Definition von Grundlagen, welche für alle Teile dieser Arbeit von Relevanz sind. Darüber hinaus wird die im Rahmen der Arbeit zu lösende Problemstellung anhand einer Analyse des Stands der Wissenschaft zur Nutzung von Ontologien in der Domäne der Automatisierungstechnik (siehe Abschnitt I-3) hergeleitet. Der Abschnitt I-4 beendet den Teil I mit einer Definition der in der vorliegenden Arbeit zu erreichenden Forschungsteilziele und Anforderungen.

I-1. Einleitung

Nachfolgend wird die Motivation der vorliegenden Arbeit in Abschnitt I-1.1 dargelegt. Der darauf folgende Abschnitt I-1.2 geht auf die im Kontext dieser Arbeit angewandte Forschungsmethodik ein und legt die Struktur der Arbeit dar. Anschließend stellt Abschnitt I-1.3 das auf der zuvor eingeführten Motivation beruhende Forschungsziel vor.

I-1.1. Motivation und Problemstellung

Software implementiert heutzutage nicht nur einen signifikanten Teil der Funktionalitäten eines automatisierten Produktionssystem, sondern realisiert meist auch die relevantesten Funktionalitäten hinsichtlich der Wettbewerbsfähigkeit des Systems [Thr10]. Darüber hinaus nimmt die Anzahl von Software-Funktionalitäten im Kontext automatisierter Produktionssysteme zu, wie an den Anwendungsszenarien der „Industrie 4.0“ [Pla16@] zu erkennen ist. Dies spiegelt sich auch im zunehmenden Einsatz von Cyber-Physischen Systemen (CPS) im Rahmen der Produktion wider [Mon15], welche auf eine zunehmende Konvergenz von physischer und virtueller Welt abzielen. Ein CPS verbindet die physische und virtuelle Welt durch Verknüpfung von Objekten beider Welten, indem bspw. Daten über physische Prozesse mit Hilfe von Sensorik gesammelt und diese mit in der virtuellen Welt vorliegenden Auswertungslogiken [Sab+19] verknüpft werden, um bspw. die Regelparameter für den physischen Prozess zu optimieren. Die Architekturen zur Nutzung dieser CPS sind dabei aufgrund unterschiedlicher marktwirtschaftlicher Trends [Wie+07] stetig in Richtung zu stärkerer Flexibilisierung gerichtet. Während heutzutage viele automatisierte Produktionssysteme entsprechend der klassischen Automatisierungspyramide strukturiert sind [Vog+09; Rüß+15@], welche eine monolithische Verknüpfung der einzelnen Systeme vorsieht, werden zukünftige Architektu-

ren wesentlich offener gestaltet sein müssen. Dies insbesondere hinsichtlich der Möglichkeit zur Integration neuer Software-Funktionalitäten [Jir+18] und einer dynamischeren Adaption bestehender Software-Funktionalitäten [Vog+14a], um eine höhere Automatisierung bei steigender Flexibilität zu gewährleisten.

Um in derart offenen und dynamischen Architekturen den Integrations-/Adaptionssaufwand möglichst gering zu halten, kommt der Beschreibung der zwischen den CPS ausgetauschten Informationen eine besondere Rolle zu, sodass die Software-Funktionalitäten die Bedeutung der ausgetauschten Informationen (gemeinhin auch als Semantik bezeichnet) möglichst automatisiert interpretieren können [Fay+15]. Da die Anzahl unterschiedlicher Informationsmodelle jedoch insbesondere im Produktionsumfeld aufgrund der Vielzahl unterschiedlicher Unternehmen, Entwickler aber auch Technologien zur Modellierung der Informationen sehr hoch ist [DPW18], ist ein wesentliches Problem die sogenannte semantische Heterogenität der von den CPS benötigten Informationen. Semantische Heterogenität bezeichnet dabei Unterschiede in der Modellierung von unterschiedlichen bzw. gleichen Konzepten (bspw. eines Sensorwertes) und deren Repräsentation (bspw. als Wert in einer relationalen Datenbank) [JOM17]. Liegt in den Informationen, welche zwischen zwei CPS ausgetauscht werden sollen, semantische Heterogenität vor, so muss diese vorher mitigiert werden, bevor die beiden CPS interagieren können [Epp11]. Auch erschwert die semantische Heterogenität der Informationsmodelle die Suche nach bestimmten Informationen, da man deren Benennung in einem Informationsmodell erst kennt, wenn man die Information gefunden hat [Pet+17]. In der Domäne der Automatisierungstechnik existieren derzeit eine Vielzahl an Industriestandards, die den Inhalt derartiger Informationsmodelle beschreiben, wie z.B. die DIN EN 62264 [DIN EN 62264 -1]. Problematisch ist jedoch, dass die meisten Standards sich lediglich auf eine textuelle oder visuelle Beschreibung der Informationsmodelle beschränken und keine formale, maschineninterpretierbare Beschreibung der Informationsmodelle bereitstellen [Fay+15].

Eine Möglichkeit zur Lösung dieses Problems ist die Definition eines „globalen“ Informationsmodells, in welches alle Informationen der jeweiligen CPS integriert werden, sodass die semantische Heterogenität durch das globale Informationsmodell mitigiert werden kann [JOM17]. CPS, die Informationen eines anderen CPS benötigen, müssen dann lediglich Kenntnis über das globale Informationsmodell besitzen und nicht über das Informationsmodell aller anderen Software-Funktionalitäten. Dies wird auch vom Forschungsbeirat der Plattform Industrie 4.0 als Forschungsbedarf identifiziert [Hir+19@]. Während bestehende Standards für die Informationsmodellierung der Domäne der Automatisierungstechnik, z.B. SysML oder AutomationML, hierfür aus technologischen Gründen umstritten sind [Sab+19], zeigt eine Vielzahl an Studien auf, dass Ontologien, oder spezifischer: die „Web Ontology Language“ (OWL), eine geeignete Sprache für ein globales Informationsmodell darstellen kann. Mit Hilfe von OWL können die Daten, Informationen und sogar das Wissen aus den vielfältigen Anwendungsfälle der Automatisierungstechnik repräsentiert werden [BiSa16]. Ein auf OWL basierendes Modell kann somit die Interoperabilität zwischen verschiedenen Software-Funktionalitäten gewährleisten, aber auch weitere Funktionalitäten ermöglichen, wie zum Beispiel automatisiertes Schlussfolgern [Gla+15]. Wie in [Wal+19@] gezeigt, werden Ontologien auch von der Plattform Industrie 4.0 als geeignet erachtet, um die Verwaltungsschalen von Industrie 4.0 Komponenten zu modellieren und somit die Interoperabilität zwischen diesen Komponenten zu gewährleisten.

Beim Entwurf von Ontologien wird jedoch oftmals auf heuristische Vorgehensweisen anstatt auf wohldefinierte Engineering-Methoden gesetzt [YCY19b]. Dies ist insofern problematisch, als dass die Entwicklung einer Ontologie mit einer Vielzahl an Herausforderungen konfrontiert ist, da vergleichsweise viele Wissensbereiche unterschiedlicher Stakeholder von Relevanz sind [Blf+17]. Diese Wissensbereiche umfassen insbesondere Wissen über die Inhalte der zu entwickelnden Ontologie, Wissen über Technologien zur Implementierung von Ontologien, sowie geeignete Methoden zu deren produktiver Nutzung in Software [JaUs99].

Aufgrund der Komplexität im Entwurf von Ontologien ergeben sich einige zentrale Problemfälle, die den Einsatz von Ontologien in der Praxis der Automatisierungstechnik bisher verhindern. Ein oftmals angeführtes Problem ist beispielsweise der hohe Aufwand der Erstellung der Ontologien [Dot+18; DiLi15]. Darüber hinaus argumentieren die Autoren in [DiLi15], dass der Umgang mit Ontologien in der Automatisierungstechnik noch nicht strukturiert genug ist, um im industriellen Umfeld eingesetzt zu werden. Dieser Sachverhalt wurde auch in [Sab+19] betont, wobei insbesondere der Mangel an adäquaten Software-Werkzeugen zur Modellierung von Ontologien hervorgehoben wird. Wie in [DPW18] beschrieben, besteht darüber hinaus auch ein Problem hinsichtlich der Wiederverwendung von Ontologien, da viele wissenschaftliche Veröffentlichungen für jeden Anwendungsfall neue Ontologien entwerfen, anstatt bestehende Ontologien wiederzuverwenden. Dieses Problem wurde auch in [Dot+18] angeführt, wobei die Autoren argumentieren, dass ohne eine anerkannte Sammlung von Ontologien die Akzeptanz der Technologie in der Domäne der Automatisierungstechnik enorm erschwert wird.

Das Potential, aber auch die genannten Problemfälle von Ontologien in der Automatisierungstechnik, motivieren das Forschungsthema dieser Arbeit: Das Engineering von ontologischen Modellen in der Domäne der Automatisierungstechnik.

I-1.2. Forschungsmethodik

Bevor Forschungsziel und Struktur dieser Arbeit vorgestellt werden, wird nachfolgend die im Rahmen dieser Arbeit angewandte Forschungsmethodik erläutert, welche die Grundlage für die Definition der Struktur dieser Arbeit liefert. Die vorliegende Arbeit ist im Bereich des „Design Science Research“ entsprechend [Hev+04] eingeordnet und zielt daher grundsätzlich auf den Entwurf von Artefakten ab, welche durch andere Wissenschaftler oder Praktiker zur Lösung wiederkehrender ingenieurtechnischer Probleme eingesetzt werden können. Als Artefakt wird in diesem Zusammenhang grundsätzlich ein Beschreibungsmittel, eine Methode oder ein Software-Werkzeug (siehe [Sch99]) bezeichnet, welches im Rahmen des Engineerings eines automatisierten Systems zum Einsatz kommt.

Grundlage der Forschungsmethodik im Rahmen dieser Arbeit ist eine hierarchische Unterteilung von Forschungszielen (siehe auch Abbildung I.1), wobei ein Forschungsziel in Anlehnung an [Wie14] folgende Fragen beantworten soll:

1. „Worum handelt es sich?“ - Benennung des zu entwickelnden Artefakts

2. „Was soll das zu entwerfende Artefakt leisten?“ - Problemkontext des Forschungsziels

3. „Wie soll es dies leisten?“ - Anforderungen an das Artefakt

4. „Warum soll das Artefakt dies leisten?“ - Motivation für die Entwicklung des Artefakts

Im nachfolgenden Abschnitt I-1.3 wird das Forschungsziel dieser Arbeit definiert, sowie das Vorgehen zur Erreichung des Forschungsziels. Da eine Beantwortung der dritten Frage obiger Definition ohne eine Definition relevanter Grundlagen und eine Untersuchung der Problemstellungen in der Ontologie-Entwicklung der Domäne der Automatisierungstechnik nicht möglich ist, werden Forschungsteilziele eingeführt. Forschungsteilziele besitzen dieselbe Gestalt wie Forschungsziele, verfügen jedoch über eine Menge von Anforderungen. Anforderungen an ein Artefakt werden entsprechend Wieringa [Wie14] aufgeteilt in funktionale Anforderungen und nicht-funktionale Anforderungen. Funktionale Anforderungen beziehen sich direkt auf den Problemkontext eines Forschungsteilziels, während nicht-funktionale Anforderungen definiert werden, um die vierte Frage eines Forschungsteilziels oder des Forschungsziels anzusprechen. Die Formulierung von Anforderungen erfolgt dabei stets nach dem Schema „(Artefakt-Anforderungen) in Kombination mit (Annahme oder Beleg) trägt bei zur Erreichung von (Antwort auf Frage 4)“. Demnach wird eine Anforderungen stets im Kontext einer Annahme oder eines Belegs definiert, welche auf einen Beitrag zur Erreichung der Zielgröße des Punkt 4 hindeuten.

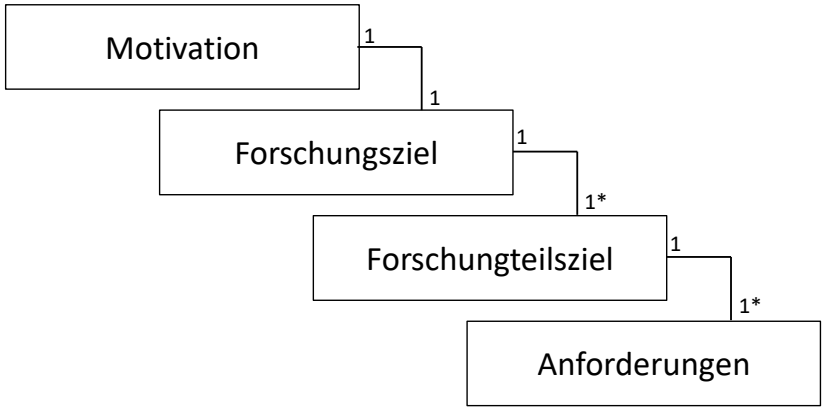

Abbildung I.1.: Zielstruktur in Anlehnung an [Wie14]

Zur Strukturierung der Untersuchung relevanter Grundlagen oder eines Wissensbereichs im Allgemeinen werden im Rahmen der vorliegenden Arbeit Wissensfragen genutzt. Wissensfragen werden zu Beginn des relevanten Abschnitts eingeführt und strukturieren die Untersuchung existierender Literatur dahingehend, dass die referenzierte Literatur hinsichtlich der Beantwortung der Wissensfragen analysiert und zusammengefasst wird [Wie14], wobei Wissensfragen in diesem Zusammenhang geschlossen oder offen formuliert werden können. Nachdem Forschungsteilziele inklusive Anforderungen definiert wurden, erfolgt eine Untersuchung der existierenden Literatur hinsichtlich eines Beitrags zum Forschungsteilziel im Sinne einer Erfüllung von Anforderungen eines Forschungsteilziels.

I-1.3. Forschungsziel und Struktur der Arbeit

Die vorliegende Arbeit widmet sich der Entwicklung von Ontologien in der Domäne der diskreten Fertigung bzw. der Automatisierungstechnik letzterer, als Teilgebiet der Produktionstechnik. Das Forschungsziel, welches entsprechend der Beschreibung in Abschnitt I-1.2 definiert wird, ist in Definition I.1 zu finden.

Definition I.1 (Forschungsziel): *Das Forschungsziel dieser Arbeit ist der Entwurf einer Engineering-Methodik zur Entwicklung von Ontologien in der Domäne der Automatisierungstechnik, welche den Entwicklungsprozess einer Ontologie soweit strukturiert, dass eine Ontologie-Entwicklung im industriellen Umfeld ermöglicht wird. Dabei soll einerseits durch Wiederverwendung entwickelter ontologischer Artefakte und andererseits durch geeignete Automatisierung von Modellierungstätigkeiten eine Effizienzsteigerung bei mehrmaliger Anwendung der Methodik erkennbar sein.*

Um dieses Forschungsziel systematisch zu erreichen, wird der Design-Zyklus entsprechend [Wie14] genutzt, welcher angewandt auf die vorliegende Arbeit in Abbildung I.2 (helle Bestandteile) zu finden ist und nachfolgend erläutert wird. Abschnitt I-2 beginnt mit einer Definition relevanter Grundlagen, bevor eine Untersuchung der Probleme in der Entwicklung von Ontologien in der Domäne der Automatisierungstechnik in Abschnitt I-3 erfolgt. Basierend auf den identifizierten Problemen werden Forschungsteilziele und Anforderungen in Abschnitt I-4 definiert.

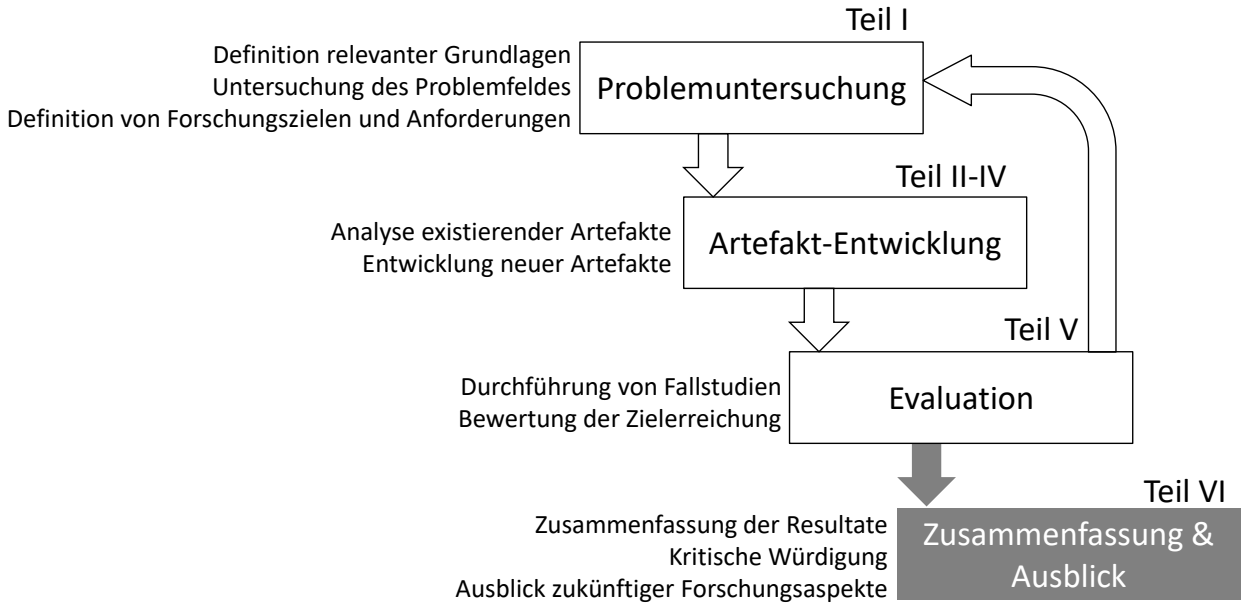

Abbildung I.2.: Struktur der Arbeit auf Basis des Design-Zyklus nach [Wie14]

Die Kapitel II bis IV befassen sich mit der Entwicklung von Artefakten in Bezug auf die in Abschnitt I-4 definierten Forschungsteilziele. Hierbei erfolgt zunächst eine Einführung relevanter Grundlagen für den jeweiligen Wissensbereich. Die Grundlagenabschnitte dieser Arbeit bauen aufeinander auf. Darüber hinaus erfolgt eine Untersuchung existierender Ansätze,

die bei der Erreichung der Forschungsteilziele als hilfreich erachtet werden. Im Rahmen der Evaluation der entwickelten Artefakte dieser Arbeit erfolgt eine Überprüfung der Erreichung des Forschungsziels. Dabei wird so verfahren, dass stets eine Überprüfung der Erfüllung der Anforderungen erfolgt. Sind die Anforderungen erfüllt, wird die Erreichung der Forschungsteilziele und des Forschungsziels diskutiert (siehe auch Kapitel V).

Forschung im Rahmen des Design-Science wird stets iterativ durchgeführt [JoPe14], was auch auf die vorliegende Arbeit zutrifft. Die hellen Bestandteile der Abbildung I.2 wurden im Rahmen der Entwicklung der Artefakte dieser Arbeit aus diesem Grunde mehrmals durchgeführt. Allerdings sind in den Kapiteln dieser Arbeit lediglich die Resultate nach Erreichung des Forschungsziels und somit mehrmaliger Iteration dargestellt und nicht die vielfältigen Zwischenergebnisse. Kapitel V geht, nach einer Einführung in die im Rahmen der Arbeit genutzte Evaluationsmethodik, auf die über mehrere Iterationen veränderte Gestalt der Artefakte ein und stellt dar, wie die Artefakte über mehrere Iteration an Reife gewannen.

Abschließend stellt Kapitel VI, welches nicht Bestandteil des Design-Zyklus ist, eine Zusammenfassung der Resultate dieser Arbeit sowie einen Ausblick für die zukünftige Forschung dar.

I-2. Grundlagen und Begriffsbestimmung

Dieser Abschnitt befasst sich mit einer Definition relevanter Grundlagen, welche für das Verständnis der Forschungsteilziele von Bedeutung sind. Hierzu stellt Abschnitt I-2.1 zunächst Grundlagen aus der Domäne des Engineerings automatisierter Systeme vor, bevor Abschnitt I-2.2 auf ontologische Grundlagen eingeht, welche aus der Domäne des Semantic Web stammen.

I-2.1. Grundlagen zum Engineering automatisierter Produktionssysteme

Der vorliegende Abschnitt definiert Grundlagen im Kontext des Engineerings, welche für die vorliegende Arbeit von Bedeutung sind. Zur Strukturierung der vorliegenden Abschnitte werden folgende Wissensfragen definiert:

1. Was ist Engineering und wie läuft das Engineering ab?
2. Was ist Systems Engineering, bzw. modellbasiertes Engineering?
3. Was ist ein automatisiertes Produktionssystem und woraus besteht es generell?
4. Was ist der Unterschied zwischen Daten, Informationen und Wissen?
5. In welchem Zusammenhang stehen Daten, Informationen und Wissen zum Engineering?
6. Was sind Cyber-Physische Systeme (CPS), und welche Relevanz haben Informations-

und Wissenmodelle für CPS?

7. Was ist semantische Heterogenität bzw. Interoperabilität?

Zur Beantwortung der Fragen eins bis drei geht Abschnitt I-2.1.1 zunächst auf terminologische Grundlagen ein. Die Modellierung von Daten, Informationen und Wissen ist im Rahmen dieser Arbeit von besonderer Relevanz. Daher wird sich Abschnitt I-2.1.2 mit der Beantwortung der Fragen vier und fünf befassen. In Abschnitt I-2.1.3 werden anschließend Cyber-Physische Systeme vorgestellt, wobei Frage sechs und sieben beantwortet werden.

I-2.1.1. Terminologische Grundlagen zum Engineering automatisierter Produktionssysteme

Entsprechend der [VDI 3695-1] ist das Engineering „die systematische Anwendung von Kenntnissen über physikalische Gesetzmäßigkeiten zur Konzeption, Erschaffung und Verbesserung von Anlagen. Das Engineering erfordert das Zusammenwirken verschiedener Gewerke, z. B. der Prozesstechnik, des Maschinenbaus, der Elektrotechnik und der Automatisierungstechnik.“ In Bezug auf Anlagen der Domäne der Fertigungstechnik definieren [FFV13]: „Das Engineering im fertigungstechnischen Maschinen- und Anlagenbau beinhaltet die Entwicklung und die Erstellung von Dokumenten durch Personen verschiedenster Domänen bzw. Disziplinen, d. h. Disziplin-/Gewerke-übergreifendes Engineering.“ Letzt genannte Gewerke sind entsprechend [VDI 2206] die an der Planung und Realisierung eines fertigungstechnischen Produktionssystems beteiligten Gewerke: Maschinenbau, Elektrotechnik und die Informationstechnik.

Ein Produktionssystem, oder System im Allgemeinen, ist entsprechend [Pat82] definiert als eine Menge von Elementen, welche Eigenschaften besitzen und welche durch Relationen miteinander verknüpft sind. Darüber hinaus verfügt ein System über eine System-Grenze, die das System umhüllt bzw. von der Umwelt trennt. Ein automatisiertes Produktionssystem besteht aus einem technischen System (auch Anlage genannt), in dem ein technischer Prozess abläuft, einem Rechner- und Kommunikationssystem zur Erfassung, Verarbeitung und Darstellung von Informationen, und dem Prozessbedienpersonal (siehe auch Abbildung I.3) [VDB13]. Entsprechend [Sch18] kann die von [Pat82] definierte „Menge von Elementen“ in die drei Kategorien Funktion, Struktur und Verhalten eingeteilt werden, welche im Rahmen der Entwicklung des Systems definiert werden müssen.

Im Allgemeinen lässt sich der Entwicklungsprozess eines Systems entsprechend [Sch99] in vier Phasen unterteilen. Hierbei wird aus der Anforderungsdefinition eine Idee bezüglich der Realisierung des Systems entwickelt, aus welcher dann in der Entwurfsphase ein Modell des Systems umgesetzt erstellt wird. In der Realisierungsphase entsteht dann entsprechend dem vorher definierten Modell des Systems ein Produkt, in Form eines Produktionssystems oder eines Produktes. Bei der Inbetriebnahme des Systems entsteht dann letztlich die in der Anforderungsphase definierte Leistung. [Sch99]

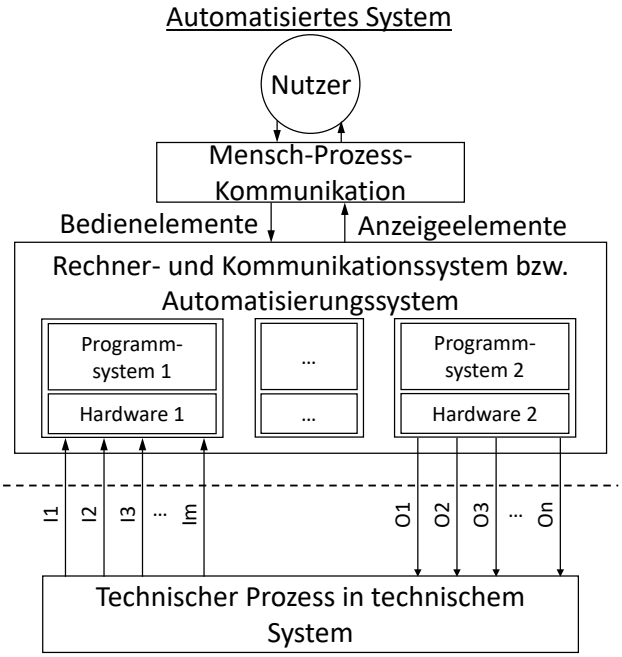

Abbildung I.3.: Prinzipieller Aufbau eines automatisierten Systems [VDB13]

Im Zuge einer zunehmenden Menge an Software-Funktionalitäten im Kontext des Engineerings automatisierter Systeme kommt Ansätzen aus dem Systems Engineering eine steigende Bedeutung zu [Vog+14a]. Das Systems Engineering ist ein interdisziplinärer Ansatz, der alle technischen und verwaltungsspezifischen Aufwände berücksichtigt, um eine Menge von Erwartungen, Anforderungen und Einschränkungen einer Gruppe von Stakeholdern in eine Lösung zu verwandeln und über den Lebenszyklus dieser Lösung zu betreuen [ISO 15288]. Im Kontext des Systems Engineering werden Ansätze aus dem Software Engineering auf das Engineering technischer Systeme übertragen, wobei unter anderem eine Nutzung von Modellierungssprachen des Software Engineerings erfolgt [Vog+14b]. Dies erfolgt mit dem Ziel, die durch die zusätzlichen Software-Funktionalitäten eingebrachte Komplexität beherrschbar zu machen. Dabei kann ein modellbasierter Ansatz genutzt werden, der von der Anforderung bis zur Inbetriebnahme Entwicklungsergebnisse in Modellen abbildet, wobei die erstellten Modelle einerseits Transparenz hinsichtlich der Entwicklungsergebnisse schaffen, aber auch partiell zur Verifikation genutzt werden können [Dum+15]. Im Zuge eines modellbasierten Ansatzes kommen oftmals Modelle der Unified Modelling Language (UML) oder Systems Modelling Language (SysML), welche durch die Object Management Group (OMG) standardisiert werden, zum Einsatz [WeSo14]. Dabei ist die Nutzung von UML- und SysML-Modellen im Engineering vergleichsweise flexibel möglich, da einerseits eine große Menge von Modellen bereits durch die OMG standardisiert sind (siehe [Obj17@] und [Obj18@]), aber auch die Erstellung daraus abgeleiteter Modellierungssprachen mit Hilfe des Profilmechanismus möglich ist (siehe [Obj17@]).

I-2.1.2. Daten, Informationen und Wissen im Kontext des Engineerings

Die nachfolgenden Abschnitte führen die relevanten Begriffe „Daten, Information und Wissen“ ein (siehe Abschnitt I-2.1.2.1) und zeigen deren Einsatz und Relevanz im Engineering auf (siehe Abschnitt I-2.1.2.2 bis I-2.1.2.4).

I-2.1.2.1. Daten, Information und Wissen

In Anlehnung an [Cho96] kann folgende Trennung der Begriffe Signale, Daten, Informationen und Wissen vorgenommen werden. Ein System, welches äußere Signale (bspw. durch Sensorik) wahrnimmt, selektiert diese und überführt sie mit Hilfe einer Zeichencodierung in eine physische Struktur. Es entstehen aus selektierten Umgebungssignalen Daten. Dies entspricht der Definition der [VDI 5610-2], wonach Daten „objektive Fakten [sind], die ohne Zusammenhang und weitere Hintergründe nicht deutbar und als Rohmaterial zu verstehen sind“. Wird diesen Daten dann eine Bedeutung zugemessen, bzw. ihre Signifikanz für eine bestimmte Aufgabe erkannt, dann werden die Daten zu Informationen, wobei eine kognitive Strukturierung zum Einsatz kommt. Dies entspricht der Definition der [VDI 5610-2], wonach Informationen „strukturierte Daten mit Relevanz und Zweck [sind], die semantisch in einen Kontext gebracht, kategorisiert, kalkuliert und korrigiert werden können“. Existieren über Informationen nun bestimmte Auffassungen, bzw. kann die Konsistenz einer Information begründet befürwortet oder widerlegt werden, oder aus der Verknüpfung mehrerer Informationen begründet eine Schlussfolgerung gezogen werden, dann werden Informationen zu Wissen. Wissen ist in der [VDI 5610-2] definiert als „vernetzte Information, die in die Lage versetzt, Vergleiche anzustellen, Verknüpfungen herzustellen und Entscheidungen zu treffen“. Abbildung I.4 stellt den vorgestellten Zusammenhang dar.

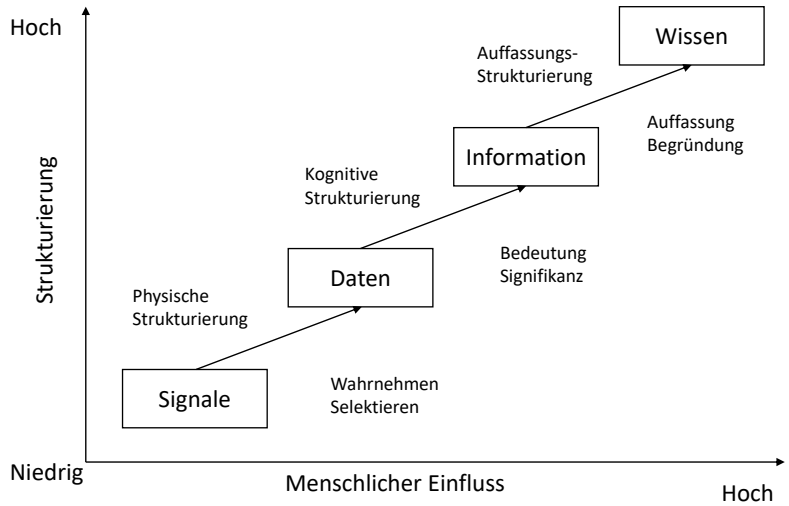

Abbildung I.4.: Von Signalen zu Wissen, nach [Cho96], Abbildung aus [Row07]

I-2.1.2.2. Datenaustauschformate im Engineering

Zur Speicherung und zum Austausch der im Rahmen des Engineerings erzeugten Daten ist ein geeignetes Datenformat nötig [Dra10]. Dabei kann entsprechend der [VDI 4499-3] in vier verschiedene Kategorien von Datenformaten unterschieden werden, welche in Tabelle I.1[1] zu finden sind und nachfolgend erläutert werden.

Ein herstellerspezifisches (proprietäres) Format ist von einem Hersteller für ein bestimmtes Softwarewerkzeug mit dem Ziel definiert worden, die Daten für ein Softwarewerkzeug bestmöglich zu speichern. Ein de-facto-Standardformat bildet sich demgegenüber meist aus proprietären Formaten heraus und erhält aufgrund einer hohen Marktdurchdringung einen standardähnlichen Status. De-facto-Standardformate können sich zu standardisierten Formaten entwickeln, sofern eine entsprechende Offenlegung des Formats durch die Hersteller gewährleistet ist. Herstellerunabhängige, nicht standardisierte Datenformate sind meist das Resultat einer herstellerübergreifenden Zusammenarbeit, mit dem Ziel, einen Austausch zwischen den eigenen Software-Werkzeugen zu ermöglichen. Spezifikationen der herstellerunabhängigen, standardisierten Formate sind hingegen in Normen. Diese Formate zeichnen sich durch eine gute Dokumentation aus, da das Gremium auf die möglichst universelle Einsetzbarkeit des Formats achtet und Entscheidungen über Änderungen und Weiterentwicklungen dokumentiert. Die Implementierung der Formatverwendung in Softwarewerkzeugen ist allerdings nicht immer identisch hinsichtlich der softwaretechnischen Realisierung, weshalb ein Im- oder Export in verschiedenen Software-Werkzeugen unterschiedliche Ergebnisse hervorrufen kann. Die Verwendung eines Datenformats für eine spezifische Anwendung ist im Hinblick auf Wiederverwendung, Konvertierungsaufwände und Datenverluste abzuwägen. [VDI 4499-3]

Tabelle I.1.: Klassifikation von Datenaustauschformaten entsprechend [VDI 4499-3]

Klassifikation der Datenaustauschformate	Beispiele
Herstellerspezifische (proprietäre) Formate	DXF, 3DS, 3D XML
De-facto-Standards	Format Wavefront Object (.OBJ); PLM XML
Herstellerunabhängige, nicht standardisierte Datenformate	IGES
Herstellerunabhängige, standardisierte Datenformate	STEP, VRML, CAEX, JT, AutomationML

I-2.1.2.3. Informationsmodelle und Informationsaustausch im Engineering

Bezogen auf den klassischen Entwicklungsablauf und den Informationsaustausch zwischen den Gewerken werden, beginnend mit dem Maschinenbau als führendem Gewerk, entsprechend [HWH14] zwischen den Gewerken Maschinenbau und Elektrotechnik im einfachsten Fall Stücklisten elektrisch relevanter Bauteile ausgetauscht. Die in der Stückliste definierten Bauteile werden dann in einem Elektro-CAD-Programm elektrisch und informationstechnisch miteinander verbunden, wobei gegebenenfalls durch den Elektroplaner Steuerungskomponenten ausgewählt und in der Stückliste ergänzt werden. Das elektrotechnische Gewerk definiert Signale von Sensoren und Aktoren in einer Signalliste oder auch Adressliste und übergibt diese an das informationstechnische Gewerk [HWH14]. Aus letztgenannter Liste und einer detaillierten Ablaufbeschreibung des Gewerks Maschinenbau erstellt das Gewerk Informati-

[1] Tabelle wurde um AutomationML ergänzt, basierend auf der Veröffentlichung der [DIN EN 62714-1]

onstechnik das Steuerungsprogramm für das zu entwickelnde System.

Dem Ansatz des funktionalen Engineerings entsprechend [Sch18] folgend, lassen sich die durch die einzelnen Gewerke erzeugten Informationen im Engineering funktionsorientiert in einem übergeordneten Modell, dem Funktionsmodell, strukturieren. Dieses Modell fungiert als durchgängiges, die Teilmodelle verbindendes Gesamtmodell. Abbildung I.5 stellt die zwischen den Gewerken ausgetauschten Informationen entsprechend [Sch18] dar. In dieser Sichtweise lassen sich die Informationen, die über ein System existieren in drei Teilbereiche gliedern, welche auch der Definition von [Pat82] folgen: Funktion, Struktur und Verhalten. Während die funktionale Sicht den Zweck des Systems beschreibt, bspw. durch eine Beschreibung des technischen Prozesses, beinhalten die anderen beiden Sichten die realisierungsrelevanten Informationen (Baustrukturen und Abläufe) [Sch18].

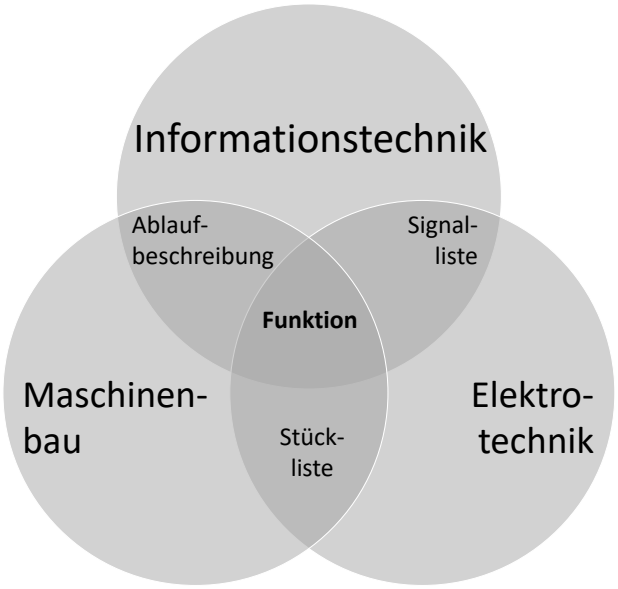

Abbildung I.5.: Erweitertes 3-Kreise Modell nach [Sch18]

Als Beispiel für die in Informationsmodellen der Digitalen Fabrik enthaltenen Informationen können Elemente der [VDI 4499-3] herangezogen werden. Um die Übersicht zu erhöhen werden diese Beispiele nachfolgend entsprechend den Kategorien Produkt, Prozess und Ressource unterteilt:

- Produkt:
  - Ein Katalog, welcher alle Bauteile und Baugruppen, die im Unternehmen verwendet werden, beinhaltet.
  - Die Produktstrukturen und Stücklisten der vom Unternehmen hergestellten Erzeugnisse einschließlich der Verbindungselemente (wie Klebenähten und Schweißpunkten).
- Prozess: Eine Beschreibung der zur Herstellung der Erzeugnisse vollzogenen Fertigungsprozesse. Ein solcher Fertigungsprozess besteht aus hierarchisch strukturierten Opera-

tionen, die untereinander durch Vorgänger-Nachfolger-Beziehungen verknüpft sind. Jede Operation betrifft ein oder mehrere Teile oder Baugruppen, erfordert den Einsatz von Ressourcen und benötigt eine gewisse Zeit zur Ausführung. Die Operationsstruktur verbindet die Produkt- mit der Ressourcenstruktur.

- Ressource:
  - Ein Katalog, welcher alle relevanten Werkzeuge und Betriebsmittel beinhaltet.
  - Die Fabrikstruktur, die die Gliederung des Werks in Hallen, Linien, Anlagen, Stationen, Logistikflächen und das verbindende Wegenetz sowie die Fördertechnik beschreibt.
  - Ressourcenstrukturen, die alle in einer Station oder Anlage physisch vorhandenen Ressourcen enthalten.

Die beispielhaft genannten Elemente sind jedoch i.d.R. nicht in einem gemeinsamen Informationsmodell verfügbar. Stattdessen verfügt jedes Gewerk für seine spezifischen Aufgaben über seine eigenen Werkzeuge und Informationsmodelle [Sch+14]. Wie in [Ang+10] beschrieben, sind die von den einzelnen Gewerken erzeugten Informationsmodelle jedoch aufgrund ihrer Komplexität und ihres hohen Detaillierungsgrades durch die anderen Gewerke nur bedingt lesbar, weshalb als Übergabedokumente oftmals einfache Listen genutzt werden. Diese Listen bilden einen Ausschnitt des eigentlichen Modells und bedürfen aufwendiger Pflege [Ang+10], was zu einem Bruch des Informationsflusses im Engineering führen kann [Sch18].

I-2.1.2.4. Wissensmodelle im Engineering

Grundsätzlich kann Wissen in implizites und explizites Wissen unterschieden werden. Während implizites Wissen an Personen gebunden, schwer kommunizierbar und kaum formalisierbar ist, ist explizites Wissen auf verschiedenen Ebenen formalisierbar bzw. formalisiert (z. B. Sprache, Schrift, Wissensmodelle), damit kommunizierbar und in verschiedenen Medien speicherbar [VDI 5610-1].

Im Verlauf der Entwicklung eines automatisierten Produktionssystems nimmt die Menge an Informationen über dieses System zu, wobei von den Anforderungen bis zur Inbetriebnahme unterschiedliche Gewerke und Personen ihr fachspezifisches Wissen einbringen. Neben der Anreicherung oder Verarbeitung von Systeminformationen mit implizitem Wissen können hierbei auch Modelle, welche eine explizite Darstellung eines bestimmten Wissensbereichs darstellen, Einsatz finden. Hierzu ist es notwendig, das entsprechende Wissen in einem Modell darzustellen, mit dessen Hilfe eine automatische Anwendung dieses Wissens und somit eine Verringerung von Fehlern sowie evtl. eine Verringerung der Aufwände erfolgen kann. [GlFa16]

Zur Nutzung derartiger Wissensmodelle können Wissensbasierte Systeme herangezogen werden. Bei Systemen dieser Art liegt eine klare Trennung des modellierten Wissens in fallunabhängiges (Klassenwissen) und fallbezogenes (Faktenwissen) Wissen vor, sowie Regelwissen und eine separate Wissensverarbeitungskomponente [Str+11]. Mittels eines Wissensbasierten Systems können Analysen von Systemen (bspw. Betriebsdatenerfassung), Dia-

gnosen (bspw. Fehlerursachen in der Produktion) sowie Gestaltungsassistenz (bspw. Gestaltungsvorschläge) implementiert werden [Run+11].

Abseits der softwareseitigen Nutzung der Wissensmodelle ist deren Inhalt stark davon abhängig, welches Wissen zur Bewältigung einer bestimmten Aufgabe in dem Modell formalisiert werden muss. So muss bspw. für die Durchführung von Security-Analysen das Wissen über komplexe Zusammenhänge zwischen Anlagenteilen, Bedrohungen und Schutzmaßnahmen formalisiert werden [GlFa16]. Für die Durchführung von Diagnosen müssen hingegen Anlagenstrukturen und Kausalketten zwischen diesen Strukturen betrachtet werden [Leg12; AVH16]. Im Rahmen der Rekonfiguration von Systemen aufgrund veränderlicher Anforderungen sind hingegen insbesondere die Fähigkeiten des Systems und die Eigenschaften der Umgebung von Interesse [Jär12]. Allen Ansätzen ist gemein, dass sie bestimmte Informationen miteinander verknüpfen und somit Kausalitäten ableiten können, Schlussfolgerungen ermöglichen, oder Konsistenz prüfen [Leg12].

Für die Repräsentation von Wissensmodellen sind im Rahmen der vorliegenden Arbeit insbesondere Ontologien aus dem Bereich der Semantic Web Technologien von Relevanz, welche in Abschnitt I-2.2 vorgestellt werden.

I-2.1.3. Cyber-Physische Systeme

Cyber-Physische Systeme (CPS) verbinden physische und digitale Prozesse, wobei meist auf software-intensive eingebettete Systeme zurückgegriffen wird, welche die physischen Prozesse überwachen, diese Informationen an einen digitalen Prozess übermitteln und mit dessen Feedback eine Steuerung des physischen Prozesses vornehmen können [Lee08].

Entsprechend [VDB13] kann es sich bei CPS um Teile von Geräten, Gebäuden, Verkehrsmitteln, Verkehrswegen, Produktionsanlagen, Logistik- und Managementprozessen handeln, welche:

- mittels Sensoren und Aktoren unmittelbar physikalische Daten erfassen und auf physikalische Vorgänge einwirken,
- weltweit verfügbare Daten und Dienste nutzen
- und über Mensch-Maschine-Schnittstellen (dediziert in Geräten, unspezifisch etwa über Browser, etc.) verfügen.

I-2.1.3.1. Informations- und Wissensmodelle im Kontext von CPS

Da im industriellen Einsatz meist nicht ein einzelnes CPS eingesetzt wird, sondern mehrere CPS miteinander interagieren, entsteht ein Netzwerk von miteinander kommunizierenden CPS, die Daten, Informationen und Wissen austauschen [KJW17]. Im Rahmen dessen übernehmen CPS eine Vielzahl von Aufgaben, welche vormals in der klassischen Automatisierungspyramide hierarschisch strukturiert waren [Vog+09] (siehe auch Abbildung

I.6). Diese verteilte Architektur schafft zwar die Möglichkeit einer flexibleren Anpassung an veränderliche Umgebungen [Mon15], ein wesentliches ingenieurtechnisches Problem hierbei ist jedoch die Organisation der dabei entstehenden großen Informationsflüsse im Rahmen der Interaktion der einzelnen CPS [Iar+16]. Das Kernproblem ist dabei nicht die Informationen zu verarbeiten, sondern die relevanten Daten für Analysen, Verfolgung, Evaluation und Verständnis laufender technischer Prozesse in der Produktion zu extrahieren [Iar+16]. Dieses Problem ist kritisch, da die auf Basis der notwendigen Daten getroffenen Entscheidungen unter Umständen laufende Prozesse beeinflussen können [Iar+16].

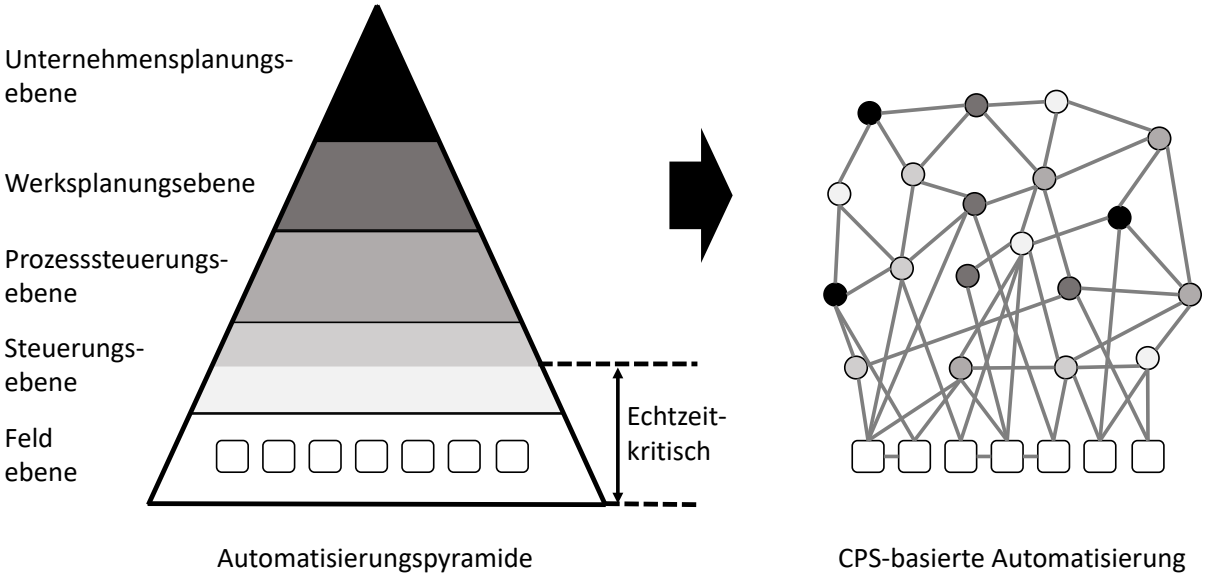

Abbildung I.6.: Dekomposition der klassischen Automatisierungspyramide in ein Netzwerk von CPS [Mon15]

Die Lösung dieses Problems wird dadurch erschwert, dass das Engineering von CPS viel komplexes Wissen erfordert und eine große Menge an Daten und Informationen erzeugt, welche durch, unter Umständen unterschiedliche, CPS in Entwicklung und Betrieb zu verarbeiten sind [Sab+19]. Im Rahmen dessen werden die unterschiedlichen Daten und Informationen über ein CPS in der Regel von unterschiedlichen Gewerken erzeugt, welche später miteinander in Verbindung gesetzt (integriert) werden müssen [Gra+18]. Dies kann zum Zwecke der Entwicklung eines einzelnen CPS sein, wobei beispielsweise geometrische Informationen und deren datentechnische Repräsentation zwischen Gewerken ausgetauscht werden müssen (siehe Abschnitt I-2.1.2.3). Auch können zum Zwecke der Integration zur Laufzeit Daten und Informationen unterschiedlicher CPS miteinander harmonisiert werden [Jir+18]. Hierbei können Probleme hinsichtlich der semantischen Interoperabilität der zu integrierenden Daten und Informationen entstehen [Gra+18; Sab+19; JOM17]. Dies ist in der Tatsache begründet, dass die genutzten Modelle bspw. Sensorwerte unterschiedlich darstellen, sowohl hinsichtlich der genutzten Syntax als auch hinsichtlich der genutzten Semantik [Sab+19]. Die Komplexität der Daten, Informationen und des Wissens, aber auch die verteilte Erstellung und Verarbeitung führen somit zu Problemen der syntaktischen und semantischen Interoperabilität und erschweren daher die Extraktion der Daten, Informationen und des Wissens.

Von besonderem Wert kann die Erstellung eines Wissensmodells sein, welches alle not-

wendigen Daten und Informationen verknüpft, sodass aus Daten und Informationen automatisiert geschlussfolgert werden kann [Iar+16]. So können mit Hilfe eines Wissensmodells bspw. Web-Services von CPS modelliert werden, sodass andere CPS durch Interpretation der Informationen und deren Verknüpfungen die notwendige Reihenfolge von Web-Services zur Erreichung ihrer Ziele bestimmen können [PLM13].

I-2.1.3.2. Semantische Heterogenität

Wie in [VDI 4499-1] beschrieben, ist der Austausch von Informationen über ein technisches System eine junge Disziplin, in der noch nicht ausreichend Erfahrung vorliegt, weshalb diverse Kommunikationsprobleme auf semantischer Ebene existieren, sodass Sender bspw. Attributwerte verwenden, die für den Empfänger keine oder eine andere Bedeutung haben. Dies wird insbesondere dann zum Problem, wenn die Vielzahl der unterschiedlichen Datenformate und Informationsmodelle für einen bestimmten Engineering-Schritt relevant sind und analysiert werden müssen, da diese semantischen Probleme dann dazu führen, dass eine aufwendige manuelle Integration der Informationen durchgeführt werden muss [Hal05].

Entsprechend [Stu11], kann bei Vorliegen von Unterschieden in der Interpretation von Informationen einer Informationsquelle durch zwei Systeme ein Konflikt entstehen, der auf sogenannte semantische Heterogenität zurückzuführen ist. Dabei kann semantische Heterogenität entsprechend [JOM17] definiert werden, als Unterschiede in der Modellierung von unterschiedlichen/äquivalenten Begriffen. Entsprechend Schnieder und Schnieder [ScSc08] ist es zur unmissverständlichen Beschreibung der Bedeutung eines Begriffes nötig, die in Abbildung I.7 dargestellten Konstituenten eines Begriffs zu definieren. Diese sind:

- Begriffsbezeichnung: Die Repräsentation, welche mit sprachlichen Mitteln erfolgen kann
- Begriffsdefinition: Beschreibt den Begriff innerhalb eines Begriffssystems durch Bezugnahme auf andere Begriffe
- Begriffsumfang: Stellt die Gesamtheit der einem Begriff auf derselben Hierarchiestufe untergeordneten Begriffe dar
- Begriffsinhalt: Stellt die Attribute dar, welche einem Begriff zugeordnet werden
- Begriffsbeziehung: Definiert die Beziehungen eines Begriffs zu anderen Begriffen, sodass kausale Zusammenhänge beschrieben werden können

Die wesentliche Ursache für semantische Heterogenität ist, dass die Informationsmodelle im Engineering zunehmend verteilt entwickelt werden und sich nicht zwingend in ein übergeordnetes Modell integrieren [Gra+18], unterschiedliche Entwickler erzeugen dementsprechend unterschiedliche Modelle über die gleichen bzw. sich überschneidenden Sachverhalte.

Mögliche Ursachen für semantische Heterogenität werden in [Sau98] klassifiziert, wobei hier in die zwei Hauptkategorien „Begriffsungenauigkeit“ und „Unschärfe komplexer Gebilde“ unterschieden wird, siehe auch Abbildung I.8. Begriffsungenauigkeiten werden weiter in

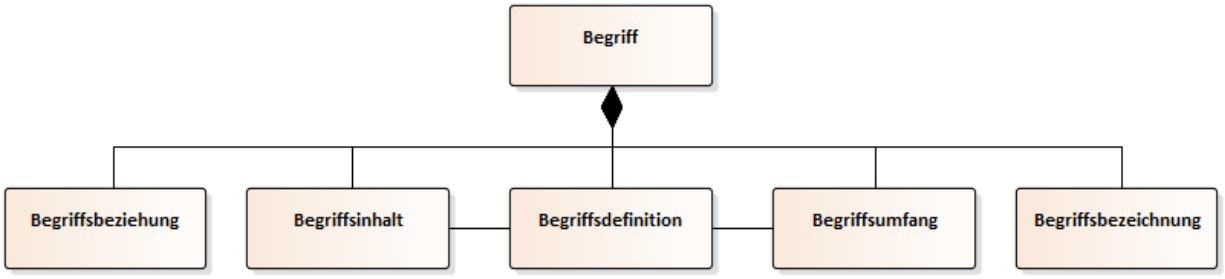

Abbildung I.7.: Konstituenten eines Begriffs nach [ScSc08]

ausdrucksstarke und ausdrucksschwache Bezeichnungen unterteilt werden, wobei ausdrucksstarke Bezeichnungen mehrere Interpretationen zulassen und somit nicht eindeutig sind und ausdrucksschwache Bezeichnungen gar keine Interpretation zulassen, aufgrund mangelnder Spezifikation. Treten Begriffsungenauigkeiten im Kontext ansonsten klarer Bezeichnungen auf, so können diese oftmals behoben werden. Anders ist es bei komplexen Gebilden (wie z. B. einem Datenbank-Schema), welche aufgrund von ungenauem Verständnis einzelner zentraler Begriffe (siehe Begriffsungenauigkeit), fehlendem Zusammenhang zwischen Elementen oder der Undurchschaubarkeit des Gebildes an sich nicht exakt verstanden werden können. [Sau98]

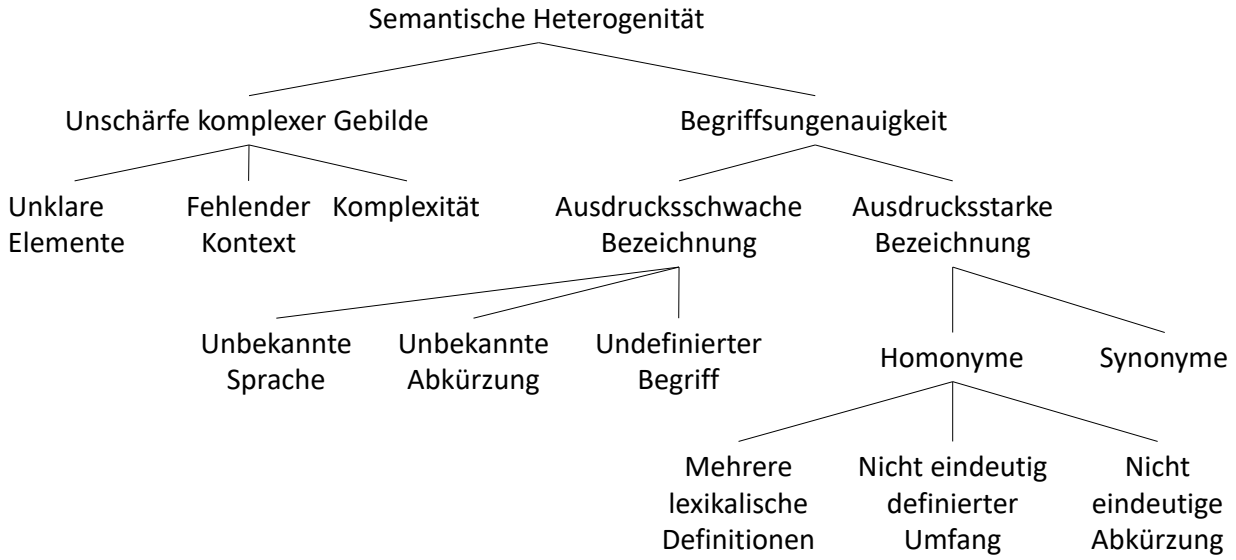

Abbildung I.8.: Klassifikation semantischer Heterogenität nach [Sau98]

I-2.1.3.3. Interoperabilität technischer Systeme

Entsprechend [Bar+12] beschreibt Interoperabilität die Fähigkeit von Software-[Werkzeugen] miteinander zu kollaborieren. In [Epp11] wird folgende Definition für Interoperabilität eingeführt: „Interoperabilität beschreibt die Fähigkeit von eigenständig und unabhängig operierenden, heterogenen Systemen ohne Hilfe von externen Einheiten zielgerichtet zusammenarbeiten zu können. Für die Interoperabilität spielt der Informationsaustausch zwi-

schen den Systemen eine wesentliche Rolle. Für die Interoperabilität der technischen Systeme im Umfeld der industriellen Produktion müssen im Allgemeinen neben explizit festgelegten Zustandsinformationen auch Modellinformationen ausgetauscht werden.“ Zur weiteren Unterscheidung hat sich im Rahmen verteilter Simulationen [Tol06] ein vielfach zitierte Abgrenzung ergeben, welche in Abbildung I.9 gezeigt ist. Demnach kann Interoperabilität in 7 Ebenen unterschieden werden:

- Keine Interoperabilität: Eigenständige Systeme, die nicht kollaborieren können.
- Technische Interoperabilität: Es existiert ein Kommunikationsprotokoll für den Austausch von Daten zwischen den beteiligten Systemen.
- Syntaktische Interoperabilität: Ein gemeinsames Datenformat ist definiert und wird angewendet.
- Semantische Interoperabilität: Innerhalb des gemeinsamen Datenformats wird ein gemeinsames Referenzmodell genutzt, um die Bedeutung der Daten zu ergänzen.
- Pragmatische Interoperabilität: Interoperierende Systeme sind sich der Methoden und des Verhaltens der beteiligten Systeme bewusst, die Verwendung übermittelter Daten ist somit bekannt.
- Dynamische Interoperabilität: Die interoperierenden Systeme kennen die Zustände der beteiligten Systeme und können entsprechend agieren.
- Konzeptuelle Interoperabilität: Konzeptuelle Modelle sind dokumentiert basierend auf definierten Engineering-Methoden, welche die Interpretation und Evaluation durch andere Ingenieure zulassen, wobei ein vollständig spezifiziertes, aber implementierungsunabhängiges Modell entsteht.

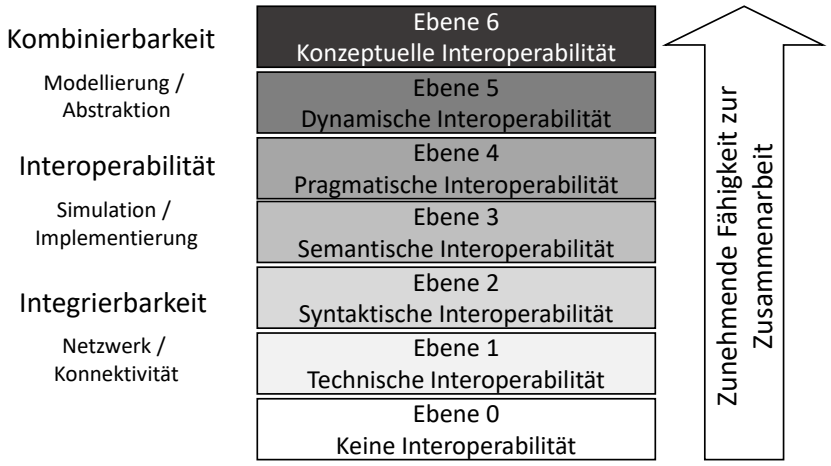

Abbildung I.9.: Ebenen der Interoperabilität zwischen technischen Systemen nach [Tol06]

I-2.2. Ontologische Grundlagen

Der vorliegende Abschnitt definiert relevante Grundlagen zu Ontologien. Zur Strukturierung der vorliegenden Abschnitte werden folgende Wissensfragen definiert:

1. Was ist eine Ontologie und woraus besteht sie?
2. Wie sieht ein vereinfachter Entwicklungsprozess einer Ontologie aus?
3. Was können relevante Stakeholder bzw. Akteure im Rahmen der Entwicklung einer Ontologie sein?
4. Was ist ein Wissensgraph?
5. Was sind mögliche Anwendungsfälle von Ontologien?
6. Welche Arten von Ontologien existieren?

Zur Beantwortung der Wissensfragen 1 bis 4 stellt Abschnitt I-2.2.1 zunächst einige terminologische Grundlagen dar. Abschnitt I-2.2.2 stellt anschließend Anwendungsfälle von Ontologien zur Beantwortung von Frage 5 vor, bevor Abschnitt I-2.2.3 den Begriff der Ontologie auf unterschiedliche Technologien abbildet und Frage 6 beantwortet.

I-2.2.1. Terminologische Grundlagen zu Ontologien

Ontologien sind ein Teilgebiet der Wissensrepräsentation und besitzen entsprechende Eignung zur maschineninterpretierbaren Beschreibung von Wissen [UsGr04]. Entsprechend [JaUs99] kann eine Ontologie eine Vielzahl an Formen einnehmen, wird aber notwendigerweise ein Vokabular von Begriffen und eine Spezifikation ihrer Bedeutung beinhalten. Dies inkludiert eine Definition der Begriffe, sowie eine Beschreibung, wie die Begriffe miteinander in Beziehung stehen. Zusammengenommen entsteht daraus die Beschreibung der entsprechenden Domäne und eine Einschränkung der Bedeutung von Begriffen in diesem Zusammenhang. Weiterhin können entsprechend [Baa+05] in einer Ontologie zwei Arten von Wissen modelliert werden. Intensionales Wissen, welches eine generelle Beschreibung von Begriffen und deren Merkmalen darstellt, und extensionales Wissen, welches Begriffe für die Beschreibung eines spezifischen Problems nutzt. Diese zwei Kategorien können entsprechend [Baa+05] als die terminologische Box (TBox) und assertionale Box (ABox) bezeichnet werden. Die TBox beinhaltet intensionales, terminologisches Wissen, welches die Eigenschaften von Begriffen einer Domäne allgemeingültig mit Hilfe von logischen Deklarationen beschreibt. Die ABox enthält extensionales, assertionales Wissen, das spezifisch für einen individuellen Fall der Domäne ist. Die Beschreibungen der ABox folgen dabei den allgemeingültigen Regeln der TBox. Intensionales Wissen (die TBox) wird gemeinhin als zeitlos betrachtet, in dem Sinne, dass das darin modellierte Wissen keinen zeitlichen Änderungen unterworfen ist. Das extensionale Wissen (die ABox) hingegen, welche sich nur auf einen bestimmten Fall bezieht, kann durchaus zeitlichen Änderungen unterworfen sein, entsprechend den Änderungen des Falls selbst. In diesem Zusammenhang ist auch der Begriff der Konsistenz von Interesse. Ent-

sprechend [GFC04] beschreibt Konsistenz, dass bei validen Eingaben keine widersprüchlichen Schlüsse gezogen werden können. Eine Definition in TBox oder ABox ist damit konsistent, wenn und nur wenn die individuelle Definition konsistent ist und nicht im Widerspruch zu anderen existierenden Definitionen und Schlussfolgerungen steht.

Der Beschreibung von [StSt09] nach, ist der Entwicklungsprozess einer Ontologie vereinfacht wie folgt. Im Rahmen der Entwicklung einer Ontologie wird die Realität einer Domäne (z.B. des Gewerks Maschinenbau) in Form von Wahrnehmungsmustern aufgefasst. Diese Wahrnehmungsmuster werden in eine Menge von Begriffen B und Relationen $\Re$ überführt. Diese Menge an Begriffen und Relationen wird dann in eine Ontologie O_K überführt, wobei eine Sprache L genutzt wird, in welche die Begriffe und Relationen überführt werden, indem B sowie $\Re$ an Elemente der Sprache L gebunden werden. Die Überführung von B sowie $\Re$ an Elemente der Sprache L wird daher auch ontologische Bindung K genannt. Dieser Vorgang wird in Abbildung I.10 zusammengefasst. [StSt09]

Zur Unterscheidung der für einen Anwendungsfall relevanten und irrelevanten Begriffe und Relationen müssen Anforderungen an eine Ontologie erhoben werden. Dies kann mit Hilfe von Kompetenzfragen und Antworten erfolgen. Eine Kompetenzfrage stellt dabei eine informelle, textuelle Frage dar, welche durch die zu entwickelnde Ontologie beantwortet werden soll [UsGr96].

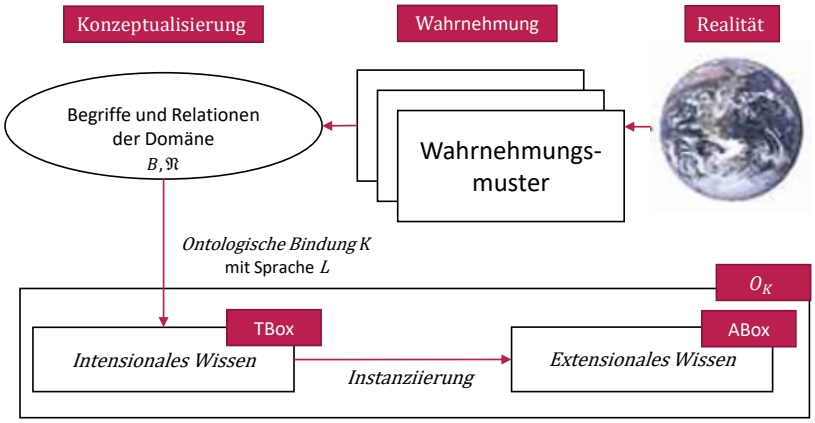

Abbildung I.10.: Vereinfachter Entwicklungsprozess einer Ontologie, in Anlehnung an [StSt09]

Die Ontologie O_K, bestehend aus einer TBox, sowie einer ABox die den Definitionen der TBox folgt, kann die Wissensbasis eines Systems definieren [UsGr04]. Dabei sind folgende Rollen von Akteuren an der Ontologie-Entwicklung generell beteiligt, wobei die individuelle Zusammensetzung je nach Projekt variieren kann [JaUs99]:

- Ontologie-Autor: Diese Rolle erstellt die Ontologie (TBox im engeren Sinne) und wird i.d.R. durch eine Person eingenommen.
- (Operationaler) Datenautor: Die Rolle des Erstellers operationaler Daten (ABox im engeren Sinne), die in der Ontologie modelliert sind. Diese Rolle kann durch eine Applikation oder Person eingenommen werden.

- Applikations-Entwickler: Die Rolle, welche die Applikation entwickelt, die die Ontologie nutzt. Diese Rolle wird i.d.R. durch eine Person eingenommen.
- Applikations-Nutzer: Die Rolle, welche die Applikation nutzt, welche vom Applikations-Entwickler erstellt wird. Diese Rolle kann durch eine Applikation oder Person eingenommen werden.
- Wissensarbeiter: Die Rolle einer Person, welche das Wissen aus der Ontologie nutzt.

Eine Ontologie ist somit die Möglichkeit, das Wissen einer Domäne zu formalisieren. Eine besondere Form von Ontologien sind die sogenannten Wissensgraphen (englisch: „knowledge graph“), welche zwar wie alle Ontologien über eine TBox und eine ABox verfügen, jedoch hinsichtlich der Größe einen starken Fokus auf die ABox legen [Pan+17]. Im Rahmen der Erstellung von Wissensgraphen werden diverse Datenquellen in einen Wissensgraphen überführt und im Rahmen dieser Überführung (auch Mapping genannt) mit zusätzlicher Semantik (z.B. Typisierungen) angereichert [Pan+17]. Technologisch gesehen können Wissensgraphen zwar mit Hilfe von RDF, RDFS, OWL (siehe auch Abschnitt III-1) implementiert werden, jedoch kann grundsätzlich jeder zusammenhängende Graph, dessen Teile typisiert werden können, als Wissensgraph betrachtet werden [Pau16].

I-2.2.2. Nutzen und Anwendungsfälle von Ontologien

Entsprechend [UsGr04] ist eine explizite, formale Semantik von Begriffen, die eine Applikation interpretieren und verarbeiten soll, insbesondere dann notwendig, je weniger die folgenden Kriterien zutreffen:

- Es gibt ein domänenweit einheitliches Verständnis über die Bedeutung von Begriffen sowie deren Syntax.
- Software-Entwickler betten diese einheitlichen Begriffe auf einheitliche Weise in Software-Applikationen ein.
- Alle Datenbanken und Software-Applikationen nutzen diese Begriffe auf eine einheitliche, abgestimmte Weise.

Daraus ergibt sich im Wesentlichen ein Nutzen der Ontologie in Form der Reduktion von Heterogenität von Begriffen und deren Interpretation. Entsprechend [Stu11] kann dabei unterschieden werden in strukturelle und semantische Heterogenität. Während strukturelle Heterogenität auf eine unterschiedliche Beschreibung desselben Begriffes hinausläuft, beschreibt semantische Heterogenität eine unterschiedliche Interpretation desselben Begriffes. Strukturelle Heterogenität tritt auf, wenn in unterschiedlichen Informationsmodellen derselbe Begriff an unterschiedlichen Stellen zu finden ist bzw. die Informationsmodelle grundsätzlich eine unterschiedliche Objektstruktur aufweisen. Semantische Heterogenität hingegen tritt auf, wenn in unterschiedlichen Informationsmodellen derselbe Begriff eine unterschiedliche Bedeutung hat, also bspw. der Begriff „Typ“ einmal zur Modellierung eines Datentyps (z.B. „integer“) und einmal zur Modellierung eines Domänentyps (z.B. „Produkt“) verwendet wird. Eine On-

tologie kann sowohl strukturelle als auch semantische Heterogenität mitigieren, allerdings nur, wenn alle beteiligten Systeme über eine geteilte ontologische Bindung verfügen, also genutzte Sprache und darin abgebildete Begriffe und Relationen allen beteiligten Systemen bekannt sind [StSt09]. Der Grund entsprechend [StSt09] ist, dass ohne ein geteiltes Verständnis der ontologischen Bindung das Problem der semantischen Heterogenität nur auf eine formale Ebene verlagert wird, da auf Ebene der Ontologie dieselben Begriffe unterschiedlich formalisiert sind.

Entsprechend [JaUs99] können vier wesentliche Anwendungsfälle von Ontologien angeführt werden: die Nutzung einer Ontologie zur neutralen Erstellung von Applikationen, zur Spezifikation von Applikationen, als gemeinsamer Zugriff auf Informationen und die ontologiebasierte Suche von Informationen. Die vier Anwendungsfälle entsprechend [JaUs99] werden nachfolgend erläutert.

Neutrale Erstellung von Applikationen: In diesem Anwendungsfall wird eine Ontologie in einer einzigen Sprache entwickelt (eine ontologische Bindung). Diese Ontologie wird dann mit Hilfe eines Mappings in verschiedene Zielartefakte übersetzt, welche von Applikationen genutzt werden. Die Vorteile dieses Ansatzes sind geringere Kosten bei der Entwicklung von Zielartefakten durch Wiederverwendung, applikationsübergreifende Übertragbarkeit von Wissen, verbesserte Wartbarkeit von Applikationen durch ein zentral wartbares Artefakt (die Ontologie). [JaUs99]

Spezifikation von Applikationen: Im Rahmen dieses Anwendungsfalls wird eine Ontologie entwickelt, wobei die in der Ontologie definierten Begriffe eine oder mehrere Applikationen spezifizieren. Die Applikationen basieren auf dieser Spezifikation, weshalb andere Applikationen oder Entwickler diese Spezifikation auslesen und nutzen können. Die Vorteile dieses Ansatzes sind die Verfügbarkeit einer semantisch homogenen Dokumentation, was wiederum die Wartbarkeit positiv beeinflusst aber auch die Wiederverwendbarkeit von Applikationslösungen erleichtert. [JaUs99]

Gemeinsamer Zugriff auf Informationen: In diesem Anwendungsfall wird eine Ontologie entwickelt, welche unterschiedlichen Applikationen (oder Menschen) einen gemeinsamen Zugriffspunkt auf heterogene Informationsquellen ermöglicht. Dabei werden die heterogenen Informationsquellen in die Ontologie überführt, sodass die Applikationen lediglich die Semantik der Ontologie kennen müssen und nicht die Semantiken der heterogenen Informationsquellen. Die Vorteile dieses Ansatzes sind insbesondere Interoperabilität, aber auch eine Steigerung der Wiederverwendbarkeit der heterogenen Informationsquellen, da diese semantisch aufbereitet werden und somit einfacher verständlich sind. [JaUs99]

Ontologiebasierte Suche von Informationen: Zentrale Idee hinter diesem Anwendungsfall ist die Beschreibung von Informationsressourcen (bspw. Dokumenten, Webseiten, Experten) mit Hilfe einer Ontologie, sodass die Suche nach Informationen mit Hilfe einer Ontologie durchgeführt werden kann. Dabei sollen die Vorteile sich insbesondere auf eine Senkung der notwendigen Zeit zur Informationsaquise sowie eine Steigerung der Präzision bei Suchen beziehen. [JaUs99]

Eine detailliertere Unterteilung ist in der Literaturanalyse von [YCY19b] zu finden, worin

11 Anwendungsfälle von Ontologien genannt werden (siehe Tabelle I.2), welche im Rahmen des Systems Engineering von Interesse sein können. Dabei verweisen die Autoren der Studie darauf, dass diese Anwendungsfälle nicht überschneidungsfrei sind.

Tabelle I.2.: Anwendungsfälle von Ontologien im Systems Engineering nach [YCY19b]

Nr.	Anwendungsfälle von Ontologien im Systems Engineering
1	Ermöglichung von Interoperabilität und Kommunikation zwischen mehreren Gewerken oder Stakeholdern
2	Integration, Mapping, Austausch und Wiederverwendung von Wissen
3	Explizite Beschreibung von Begriffen, deren Relationen zur Vermeidung von Unvollständigkeit und Mehrdeutigkeit
4	Entwicklung einer Wissensrepräsentation für eine Domäne
5	Vereinheitlichung von deklarativem Wissen durch ein kontrolliertes Vokabular
6	Bereitstellung von Kern- und Basisbegriffen zur Beschreibung anderer Begriffe
7	Definition einer homogenen Terminologie zur Eliminierung von Inkonsistenzen
8	Ermöglichung eines geteilten Verständnisses über eine Domäne
9	Formalisierung von Wissen
10	Modellierung und Inferenz von maschinenlesbaren, programmierbaren, komplexen logischen Axiomen
11	Visualisierung und Navigation in einer Wissensdatenbank

Eine noch detailliertere Darstellung von Anwendungsfällen ist in [Sab+19] in Bezug auf CPS zu finden. Demnach können Semantic Web Technologien (siehe Kapitel III), bzw. Ontologien im engeren Sinne, im Rahmen der Entwicklung insbesondere eingesetzt werden, um verschiedene Informationsmodelle miteinander zu verknüpfen, die Konsistenz zwischen den Informationsmodellen zu prüfen und die Wiederverwendung der Entwicklungsartefakte durch erleichterte Auffindbarkeit zu erhöhen. Im Rahmen des Betriebs eines CPS können Ontologien hingegen eingesetzt werden, um ein Monitoring sowie eine Anomaliedetektion des Verhaltens der CPS zu ermöglichen, im Rahmen des Wartungsmanagement defekte Bauteile zu identifizieren und neue vorzuschlagen, sowie im Rahmen der Rekonfiguration Vorschläge für Anpassungen an veränderliche Anforderungen zu erstellen. [Sab+19]

I-2.2.3. Ontologisches Spektrum und Arten von Ontologien

Wie in Abschnitt I-2.2.1 angeführt, können unterschiedliche Sprachen L für die Definition einer Ontologie O_K genutzt werden. Mögliche Sprachen von Ontologien können nach [UsGr04] entsprechend ihres Formalisierungsgrades unterschieden werden. Dabei ist der Formalisierungsgrad definiert durch die Möglichkeit zur formalen (maschineninterpretierbaren) Definition von Semantik, also der Bedeutung von Begriffen. Abbildung I.11 zeigt das ontologische Spektrum von Uschold und Gruninger. Wesentlich in der Unterscheidung ist nicht nur der Formalisierungsgrad, sondern auch die Klassifikation. Hierbei unterscheiden die Autoren in vier Kategorien (siehe Abbildung I.11 für Beispiele): Glossare und Datenkataloge, Thesauri und Taxonomien, Datenmodelle bzw. XML Schemata, Metadaten und logische Sprachen.

Während alle in Abbildung I.11 gezeigten Sprachen der Definition einer Ontologie entsprechend Abschnitt I-2.2.1 genügen, weisen die unterschiedlichen Sprachen aufgrund ihres unterschiedlichen Formalisierungsgrades eine unterschiedliche semantische Ausdrucksstärke

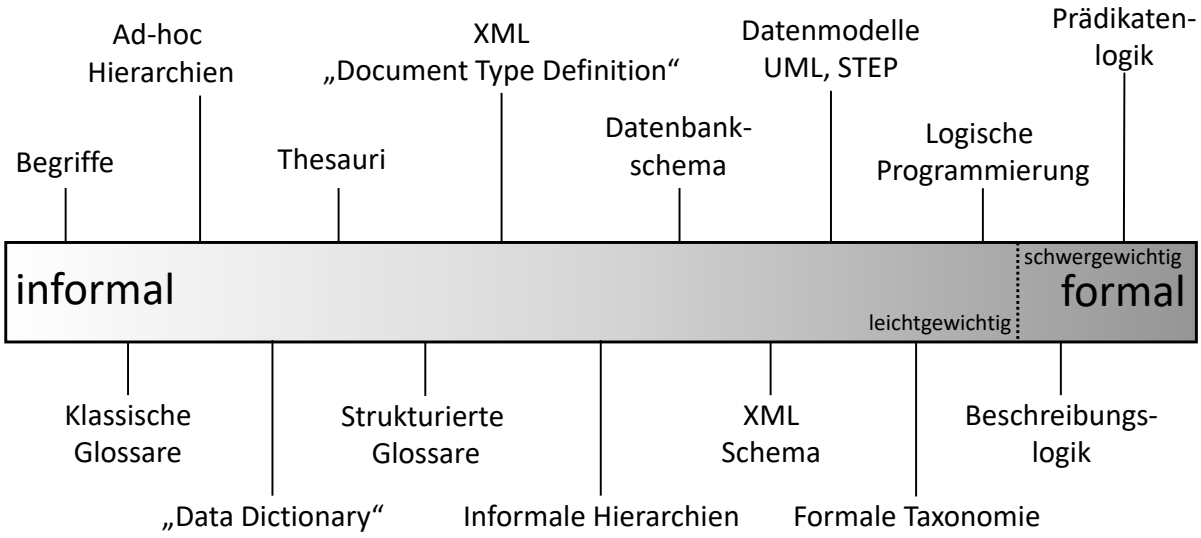

Abbildung I.11.: Ontologisches Spektrum in Anlehnung an [UsGr04]

auf [LaMc01]. In [GFC04] wird daher eine Unterscheidung von Ontologien in zwei Klassen eingeführt: leichtgewichtige (englisch: „lightweight“) und schwergewichtige (englisch: „heavyweight“) Ontologien[2]. Dabei sind alle Ontologien leichtgewichtig, welche auf einer Sprache beruhen, die keine Modellierung von logischen Axiomen und Beschränkungen (englisch: „constraints“) zulässt, wobei im Umkehrschluss alle Ontologien, die auf einer Sprache beruhen, die dies zulässt, sogenannte schwergewichtige Ontologien sind. Da die Modellierung von logischen Axiomen und Beschränkungen nur in logischen Sprachen wie z.B. Beschreibungslogik, oder Prädikatenlogik möglich sind [Baa+05], werden schwergewichtige Ontologien i.d.R. mit Ontologien gleichgesetzt, welche in einer logischen Sprache formuliert sind.

Des Weiteren können Ontologien unterschieden werden hinsichtlich des Abstraktionsgrades ihrer enthaltenen Begriffe und Relationen [GFC04], in Top-Level Ontologien, Domänenontologien und aufgabenspezifische Ontologien:

- Top-Level Ontologien beschreiben sehr generische Begriffe und stellen einen übergeordneten Begriffsrahmen bereit, unter welchen detailliertere Begriffe taxonomisch untergliedert werden. Die Wiederverwendbarkeit dieser Ontologien soll unabhängig von Domänen oder Aufgaben sein, wobei jedoch die Anwendbarkeit aufgrund der sehr abstrakten Begriffe beschränkt ist.

- Domänenontologien stellen Begriffe bereit, welche nur in einer bestimmten Domäne (Medizin, Biologie, Ingenieurwesen, Automotive, o.ä.) von Relevanz sind. Die Begriffe, die in einer Domänenontologie verwendet werden, lassen sich i.d.R. den Begriffen einer Top-Level-Ontologie unterordnen. Die Wiederverwendbarkeit ist zwar auf eine bestimmte Domäne begrenzt, bietet jedoch eine höhere Anwendbarkeit aufgrund der höheren Spezialisierung der Begriffe.

[2]Im Verlauf der Arbeit werden die englischen Begriffe für schwergewichtige („heavyweight“) und leichtgewichtige („lightweight“) Ontologien, sowie Beschränkungen („Constraints“) genutzt

- Aufgabenspezifische Ontologien beschreiben die Begriffe, die von Relevanz für die Bearbeitung/Durchführung einer bestimmten Aufgabe/Aktivität sind (bspw. Diagnose, Planung, etc.), wobei auch eine Spezialisierung von Begriffen einer Top-Level-Ontologie erfolgen kann. Dabei kann die aufgabenspezifische Ontologie Bezüge zu einer domänenspezifischen Ontologie haben.

I-3. Stand der Wissenschaft zur Nutzung von Ontologien im Engineering automatisierter Produktionssysteme

Während im vorangegangenen Kapitel Grundlagen zum Themenbereich Engineering und Ontologien dargestellt wurden, widmet sich dieses Kapitel der Verbindung beider Themenbereiche. Hierzu werden zunächst Wissensfragen zur Untersuchung der einschlägigen Literatur definiert, aus denen dann Analysekriterien abgeleitet werden. Im Anschluss werden Ansätze vorgestellt und analysiert, welche Ontologien im Rahmen des Engineerings automatisierter Produktionssysteme einsetzen. Daraufhin erfolgt eine Bewertung des Stands der Wissenschaft mit der Ableitung einer wissenschaftlichen Lücke, welche im Rahmen dieser Arbeit geschlossen werden soll.

I-3.1. Kriterien zur Analyse des Stands der Wissenschaft

Zur Analyse des Stands der Wissenschaft werden nachfolgend Wissensfragen definiert, die für jede einzelne Veröffentlichung getrennt beantwortet werden sollen. Auf diese Weise soll ein einheitliches Bild des Stands der Wissenschaft zur Nutzung von Ontologien im Engineering geschaffen werden.

1. Welchen generellen Einsatzzweck einer Ontologie verfolgt der Ansatz?

Im ersten Schritt ist die Frage zu klären, welcher generelle Einsatzzweck einer Ontologie in einem bestimmten Ansatz verfolgt wird. Die Beantwortung dieser Frage soll Aufschluss darüber geben, warum die Autoren sich für eine Ontologie entschieden haben. Im Allgemeinen lässt sich der Einsatzzweck einer Ontologie entsprechend [JaUs99] in vier Kategorien aufteilen (Neutrale Erstellung, Ontologie als Spezifikation, Gemeinsamer Zugriff auf Informationen, ontologiebasierte Suche). Diese Unterteilung muss natürlich nicht trennscharf sein, da eine Ontologie auch für mehrere Zwecke eingesetzt werden kann. Dennoch bietet sich die Unterteilung an, um aus der Vielzahl verfügbarer Publikationen eine erste Gruppierung vorzunehmen.

2. Welchen spezifischen Engineering-Einsatzzweck einer Ontologie verfolgt der Ansatz?

Da es sich bei der untersuchten Literatur um Veröffentlichungen im Kontext des Engineerings handelt, wird jede Veröffentlichung neben den oben genannten, generellen Einsatzzwecken einer Ontologie auch einen spezifischen Einsatzzweck verfolgen. Der spezifische Einsatzzweck soll dabei eine Engineering-Tätigkeit darstellen, bei der die Ontologie genutzt wird.

3. Wie werden die Anforderungen an die Ontologie definiert?

Da in einem Wissensmodell, mit begrenzten Aufwand, nur eine bestimmte Menge an Wissen formalisiert werden kann, müssen im Rahmen eines Projektes Anforderungen definiert werden, um die relevante Wissensbestandteile zu identifizieren und zu charakterisieren [VDI 5610-2]. Die Beantwortung dieser Frage soll Aufschluss darüber geben, wie die Autoren die Anforderungen an die Ontologie bestimmt haben.

4. Welche Kategorien von Wissen werden in der Ontologie formalisiert?

Wie in vorangegangenen Abschnitten (siehe insb. Abschnitt I-2.1.2.3) gezeigt, hat sich die Unterteilung von Modellen in Struktur-, Funktions- und Verhaltensmodelle bewährt. Die Beantwortung dieser Frage soll Aufschluss darüber geben, für welche dieser drei Perspektiven Ontologien entwickelt werden und was in der jeweiligen Perspektive dargestellt wird.

5. Welche Wissensgrundlage wird zur Ontologie-Erstellung genutzt?

In einer Ontologie wird Wissen expliziert und formalisiert. Für diesen Zweck ist eine entsprechende Wissensquelle notwendig, welche grundsätzlich zweierlei sein kann. Sie kann implizites Wissen von Projektmitarbeiter explizieren oder vorhandenes explizites Wissen formalisieren. Im ersten Fall ist der Mensch die Hauptquelle des Wissens, während im zweiten Fall existierende Artefakte die Hauptquelle des Wissens darstellen. Die Beantwortung dieser Frage soll Aufschluss darüber geben, welche Wissensquellen durch die jeweiligen Autoren genutzt werden.

6. Welche Ontologien werden zur Erstellung wiederverwendet?

Wie in vorangegangenen Abschnitten beschrieben, trägt Wiederverwendung von Modellen im Engineering maßgeblich zur Optimierung von Kosten, Zeitaufwand, Qualität der Lösung sowie Wartungsaufwand. Die Beantwortung dieser Frage soll Aufschluss darüber geben, welche Ontologien wiederverwendet werden, bzw. ob überhaupt Ontologien wiederverwendet werden.

7. Wie werden die Ontologien zur Wiederverwendung bereitgestellt?

Da es sich bei den untersuchten Veröffentlichungen um wissenschaftliche Literatur handelt, ist anzunehmen, dass die erstellten Ontologien der jeweiligen Autoren der wissenschaftlichen Community zur Wiederverwendung bereitgestellt werden. Die Beantwortung dieser Frage soll Aufschluss darüber geben, auf welchem Wege Wissen, das in Forschungsprojekten formalisiert wurde, für industrielle Projekte bereitgestellt wird.

8. Welche Methodik wird zur Erstellung der TBox genutzt?

Die Beantwortung dieser Frage zielt darauf ab, die von den jeweiligen Autoren genutzten Methodiken zu identifizieren, welche für diese Engineering-Tätigkeit genutzt wurden.

9. Welche Werkzeuge werden zur Erstellung der TBox genutzt?

Da anzunehmen ist, dass allein aus Gründen der Effizienz eine entsprechende werkzeugtechnische Unterstützung zur Modellierung der TBox genutzt wird, soll die Beantwortung dieser Frage Aufschluss darüber geben, welche Werkzeuge in den jeweiligen Projekten eingesetzt wurden.

10. Welche Methodik wird zur Erstellung der ABox genutzt?

Die ABox enthält extensionales, assertionales Wissen, das spezifisch für einen individuellen Fall der Domäne ist. Nimmt man also bspw. an, dass in einem Projekt eine TBox für Funktionen entworfen wurde und in der ABox die spezifischen Funktionen einer Anlage modelliert werden, so soll die Beantwortung dieser Frage Ausschluss darüber geben, welche Methodik zur Erstellung der hierfür notwendigen ABox genutzt wurde.

11. Welche Werkzeuge werden zur Erstellung der ABox genutzt?

Da wie bei der Erstellung der TBox anzunehmen ist, dass eine entsprechende werkzeugtechnische Unterstützung vorliegen sollte, soll auch hier die Beantwortung der Frage Aufschluss über genutzte Werkzeuge liefern.

Tabelle I.3 fasst die erläuterten Kriterien zusammen und verweist pro Kriterium auf eine entsprechende Einordnung, welche entweder geschlossen vorgenommen wird (siehe z.B. „Allgemeine Kategorie des Einsatzzweck“), oder ohne entsprechende Klassifikation schlicht genannt wird.

Tabelle I.3.: Merkmale zur Analyse des Stands der Wissenschaft

Merkmal	**Einordnung**
Allgemeine Kategorie des Einsatzzweck	Neutrale Erstellung / Ontologie als Spezifikation / Gemeinsamer Zugriff auf Information / ontologiebasierte Suche
Spezifischer Engineering-Einsatzzweck	Nennung
Methodik der Anforderungserhebung	Nennung
Kategorien von Wissen und wesentliche Inhalte	Struktur / Funktion / Verhalten und Nennung wesentlicher Inhalte
Wissensgrundlage der Ontologie	Implizites Wissen mit Nennung / Explizites Wissen mit Nennung
Wiederverwendung existierender Ontologien	Nennung
Verfügbarkeit der ontologischen Artefakte	Nennung der Art der Verfügbarkeit entsprechend Veröffentlichung
Genutzte Methodik zur TBox-Modellierung	Nennung
Genutzte Tools zur TBox-Modellierung	Nennung
Genutzte Methodik zur ABox-Modellierung	Nennung
Genutzte Tools zur ABox-Modellierung	Nennung

I-3.2. Anwendungsfälle von Ontologien im Engineering automatisierter Produktionssysteme

In den nachfolgenden Abschnitten werden Ansätze aus der Domäne der Automatisierungstechnik vorgestellt, welche eine Ontologie zur Lösung eines bestimmten Problems entwerfen. Herangezogen wurden diese Ansätze, da sie neben der Nutzung einer Ontologie zur Lösung eines bestimmten Problems zusätzlich auch Erkenntnisse zu den in Abschnitt I-3.1 definierten Fragen liefern. Darüber hinaus wurde bei der Auswahl darauf geachtet, möglichst alle in Abschnitt I-2.2.2 genannten Anwendungsfälle abzudecken (siehe auch erste Zeile in Tabelle I.3), sowie eine Konzentration des spezifischen Einsatzzwecks auf einzelne Anwendungsfälle zu vermeiden (siehe auch zweite Zeile in Tabelle I.3). Die Darstellung der einzelnen Ansätze erfolgt unter Berücksichtigung der in Abschnitt I-3.1 definierten Fragen.

Kontextbasierte Orchestrierung semantischer Feldgerätefunktionalitäten - nach Loskyll

In [Los+12] wird ein Ansatz beschrieben, durch welchen eine Ad-hoc Orchestrierung von Feldgerätefunktionalitäten, basierend auf aktuellen Produktionsanforderungen durchgeführt werden kann. Hierzu sind die Feldgerätefunktionalitäten sowie die Produktionsanforderungen mit Hilfe einer Ontologie spezifiziert. Zur Spezifikation der Anforderungen an die Ontologie wird eine textuelle Spezifikation von Funktionalitäten durchgeführt, die die Ontologie nutzen. Die TBox ist hierbei mit Hilfe der Methodik aus [FGJ97] aus diversen Standards aufgebaut und besitzt eine Formalisierung funktionaler (Feldgerätefunktionalitäten und deren Parameter), struktureller (Anlagenstruktur und Equipment) und verhaltensorientierter (Spezifikation der Web-Services zu deren Aufruf) Wissensbereiche entsprechend der berücksichtigten Standards. Für die Modellierung von TBox und ABox wurde das Werkzeug Protégé [Sta19@] genutzt, wobei für die Erstellung der ABox eine selbst entworfenes Vorgehen basierend auf manuell editierten Vorlagen genutzt wurde. Wiederverwendung fanden eine Ontologie für Web-Services (OWL-S entsprechend [Mar+07]), sowie eine Upper-Ontologie (AVILUS-Ontologie entsprechend [Bre+11]). Die Ontologie dient im Rahmen des Ansatzes einerseits als Spezifikation der Feldgerätefunktionalitäten, aber auch als gemeinsamer Zugriffspunkt für alle beteiligten Systeme. Des Weiteren wird die Ontologie genutzt, um die Suche nach passenden Feldgerätefunktionalitäten bei Vorliegen einer Produktionsanfrage durchzuführen. [Los+12]

Adaptivität und semantische Interoperabilität von Manufacturing Execution Systemen - nach Schleipen

Der in [Sch13] vorgestellte Ansatz verfolgt das Ziel semantischer Interoperabilität zwischen Manufacturing Execution Systems (MES), wobei eine Ontologie als Verständigungs-, Koordinations- und Mappinghilfe entwickelt wurde. Demnach dient die Ontologie dem Zweck, eine Spezifikation der zwischen MES ausgetauschten Informationen bereitzustellen. Zur Herleitung der Anforderungen wurden die Aufgaben (im Sinne von Funktionen) von MES aufgegführt und deren Informationsbedarf bestimmt. Die TBox wurde mit Hilfe einer auf

[Leu06@] basierenden Methode entwickelt und formalisiert strukturelle (Struktur der Anlage) und verhaltensorientierte (Datenpunkte und Services) Wissensbereiche. Die entwickelten Ontologien sind kostenpflichtig im Rahmen des Erwerbs der Richtlinie [VDI 5600-3] erhältlich. Die Wissensgrundlagen liefern neben diversen Standards auch implizites Wissens aus einer VDI-Arbeitsgruppe. Für die Modellierung der TBox wurde das Werkzeug Protégé [Sta19@] genutzt, sowie eine Eigenentwicklung (OWLTreePrint), welche den Fachexperten die teilweise Formalisierung der TBox ermöglichen sollte. [Sch13]

Manufacturing Resource Capability Ontology - nach Järvenpää et al.

In [Jär12] wird ein Ansatz zur fähigkeitsbasierten Anpassung von Produktionssystemen in veränderlichen Umgebungen vorgestellt. Dabei erfolgt eine Spezifikation von Fähigkeiten produktionstechnischer Systeme zum automatischen Abgleich von Anforderungen und aktuellen Fähigkeiten einer Anlage/Teilanlage und zur Generierung von Vorschlägen zur Adaption der Anlage an neue Anforderungen, sofern aktuelle Fähigkeiten die Anforderungen nicht erfüllen können. Alle beteiligten Systeme nutzen hierbei eine Ontologie als gemeinsamen Zugriffspunkt für Informationen und können nach aktuellen Produktanforderungen oder existierenden Fähigkeiten suchen. Der in [Jär+19] vorgestellte Entwicklungsprozess nutzt Kompetenzfragen und Antworten für die Modellierung der Anforderungen an die Ontologie, wobei im Rahmen der TBox-Entwicklung auf die Methodik nach [SSS09] zurückgegriffen wird. Im Rahmen der Entwicklung werden einerseits diverse Standards formalisiert, allerdings auch implizites Wissen der Autoren. Die TBox formalisiert funktionales (Fähigkeiten und deren Parameter) und strukturelles (Anlagenstrukturen) Wissen, wobei das Werkzeug Protégé [Sta19@] zur Modellierung von TBox und ABox genutzt wird. Die erstellte Ontologie ist entsprechend [Jär+19] online verfügbar.

Adaptive Production Management Ontology - nach Harcuba et al.

In [HaVr15; Kaz+13] wird eine Ontologie vorgestellt, welche zur Optimierung von Produktionsstarts von Produkten in der Luftfahrt entwickelt wurde. In dem vorgestellten Ansatz erfolgt eine eventbasierte Fertigungsplanung in Anlaufphasen der Produktion durch ein Agentensystem, wobei die Ontologie unternehmensübergeifend die relevanten Daten für die Fertigungsplanung repräsentiert und somit als gemeinsamer Informationszugriffspunkt für die Agenten, aber auch für eine Suche nach Informationen, bspw. aktuellen Aufträgen, dient. Für die Anforderungserhebung an die Ontologie wurde eine UML-basierte Dokumentation der Interaktionen zwischen den Agenten mit Hilfe eines Sequenzdiagramms durchgeführt. Die TBox basiert zwar auf Standards, weist jedoch, wie in [Kaz+13] erkennbar, auch implizites Wissen der Autoren auf, wobei strukturelles (Anlagenstruktur) und verhaltensorientiertes Wissen (Spezifikation von Services für den Aufruf von Aktionen) formalisiert wird. Für die Modellierung der ABox wird ein nicht veröffentlichtes, von den Autoren in [Kaz+13] definiertes „Ontology Management Tool“ genutzt, um die ABox zu generieren.

Context Modeling with Situation Rules for Industrial Maintenance - nach Aarnio et al.

Der Beitrag in [AVH16] beschreibt ein kontextsensitives System, das Wartungspersonal mit wichtigen Informationen über die Wartungsaufgabe versorgen soll. Dabei wird eine Ontologie genutzt, welche mit Informationen über den aktuellen Kontext des Wartungstechnikers (Aufgabe, ERP-, MES-, Businformationen, etc.) gefüllt wird. Für die Anforderungserhebung an die Ontologie wird eine UML-basierte Dokumentation der Kategorien von relevanten ontologischen Begriffen und deren Attributen in Form eines UML-Klassendiagramms genutzt. Die Ontologie dient somit als gemeinsamer Zugriffspunkt für Applikationen der Wartungstechniker und erlaubt auch eine ontologiebasierte Suche nach wartungsrelevanten Informationen, wobei die Ontologie funktionales (Wartungsaufgaben), strukturelles (Anlagenstrukturen und Orte), sowie verhaltensorientiertes (Zustände der Anlagen) Wissen enthält. Für die Erstellung der TBox wurde eine von den Autoren entwickelte Methodik genutzt, wobei explizites Wissen aus Standards und Veröffentlichungen der entsprechenden Domäne (Wartung), sowie implizites Wissen der Autoren formalisiert wurde. Für die Erstellung von TBox und ABox wurde das Werkzeug TopBraid Composer [Top19[@]] genutzt.

RDF-based Information for Manufacturing Companies - nach Petersen et al.

Die Autoren in [Pet+17] stellen einen Ansatz vor, durch den ein maschinenübergreifendes Werkzeug- und Energiemanagement ermöglicht wird. Hierzu wird mittels einer Ontologie ein gemeinsamer Zugriffspunkt geschaffen, der die semantisch heterogenen Datenquellen der Maschinen in einem übergeordneten Modell vereint. Anfragende Applikationen können dann über die Ontologie auf die Quelldaten zugreifen, ohne dass sie die Semantik der Quelldaten kennen müssen. Die Autoren erheben die ontologischen Anforderungen mit Hilfe von Kompetenzfragen und Antworten. Bei der Erstellung der TBox wird auf die in [Hal+16] definierte Methodik zurückgegriffen, wobei explizites Wissen aus diversen Standards, aber auch implizites Wissen der beteiligten Autoren wiederverwendet wird. Darüber hinaus werden auch einige Domänenontologien wiederverwendet. Die entwickelten ontologischen Artefakte sind online veröffentlicht. Für die Erstellung der TBox werden die Werkzeuge TopBraid Composer [Top19[@]] und Protégé [Sta19[@]] verwendet. Für die Erstellung der ABox werden einerseits ein durch [Hal+16] entwickeltes Werkzeug und andererseits Mappings verwendet. [Pet+17]

IEEE Core Ontology for Robotics and Automation - nach Schlenoff et al.

In [Sch+12] werden die Resultate der IEEE-RAS (Robotics and Autonomous Systems) Arbeitsgruppe ORA (Ontologies for Robotics and Automation) vorgestellt. Das Ziel der Arbeitsgruppe war die Entwicklung von einer Standard-Ontologie und dazugehörigen Methoden für die Wissensrepräsentation und Schlussfolgerungen auf deren Basis im Kontext der Robotik und Automation. Der Standard [IEEE 1872-2015] ist das Resultat dieser Arbeitsgruppe. Die IEEE Norm ist grundsätzlich nicht begrenzt auf einen spezifischen Anwendungszweck in der Robotik, allerdings werden zwei Beispiele herangezogen. Das erste Beispiel bezieht sich auf autonome Unterwasserfahrzeuge, welche ihre Daten bzw. Informationen entsprechend der Onto-

logie mit anderen Fahrzeugen und Software-Plattformen austauschen. Hierbei wird mit Hilfe der Ontologie ein Austausch von Daten und Informationen zwischen verschiedenen Agenten realisiert. Der zweite Anwendungsfall bezieht sich auf die Verpackung von Produkten, wobei die Ontologie hierbei die aktuelle Situation repräsentiert und Schlussfolgerungen bei Problemen zulassen soll. Die TBox formalisiert im wesentlichen strukturelles (Umgebungsinformationen, sowie Systemstrukturen) Wissen, wobei entsprechend [Fio+15] an Erweiterungen um funktionale und verhaltensorientierte Aspekte gearbeitet wird. Die Ontologien des Standards sind einzeln verfügbar und können individuell verwendet werden.

Manufacturing's Semantics Ontology - nach Lemaignan et al.

In [Lem+06] wird die „MAnufacturing's Semantics ONtology" (MASON) vorgestellt. Die Ontologie wird mit dem Zweck entworfen, wesentliche fallunabhängige Konzepte der fertigenden Industrie abzubilden, sodass MASON als eine Upper-Ontologie dienen kann, mit deren Hilfe diverse Domänenontologie integriert werden können. Der wesentliche Fokus der Ontologie liegt auf der Beschreibung von Entitäten, Operationen und Ressourcen im Umfeld der Produktion, womit die Ontologie einerseits strukturelle, aber auch funktionale Aspekte beinhaltet. Auf diese Weise soll ein effektiver Austausch von Daten und Informationen in heterogenen Umgebungen ermöglicht werden. Die Autoren entwerfen die Ontologie auf Basis ihrer Erfahrungen aus vorherigen Projekten, welche sich unter anderem mit Agentensystemen beschäftigten, in denen eine gemeinsame Sprachbasis zwischen den Agenten geschaffen werden muss, aber auch mit automatischer Kostenabschätzung in der Produktion, welche eine Vielzahl von historischen Daten aus heterogenen Datenquellen erfordert. Die Anforderungserhebung an die Ontologie wird mit Hilfe von vergleichsweise abstrakten textuellen Definition der die Ontologie nutzenden Funktionen durchgeführt. Die Ontologie ist online verfügbar. [Lem+06]

Automatic generation of field control strategies for supporting (re-)engineering of manufacturing systems - nach Legat et al.

Der in [LSV14] vorgestellte Ansatz erlaubt im Falle von notwendigen Änderungen der Steuerungsstrategie eines Produktionssystems, bspw. durch einen Produktwechsel, ein Re-Engineering der Anlage. Bei diesem Vorgang werden die Fähigkeiten der Anlage mit den neuen Anforderungen verglichen und dann eine neue Steuerungsstrategie abgeleitet. Für die Ableitung aller möglichen Alternativen (Zustandsraum) wird eine Ontologie als formale Spezifikation der notwendigen Informationen genutzt. Darüber hinaus werden durch Schlussfolgerung weitere Informationen ergänzt. Für die Anforderungserhebung nutzen die Autoren eine vergleichsweise abstrakte textuelle Spezifikation der notwendigen TBox-Inhalte. Die TBox formalisiert strukturelles (Ressourcen und deren Strukturen), funktionales (fertigungstechnische Operationen) und verhaltensorientiertes (Zustände sowie deren Vor- und Nachbedingungen von Operationen) Wissen, welches im Wesentlichen dem impliziten Wissen der Autoren entstammt. Für die Modellierung der TBox und ABox wird im Wesentlichen die OWL-API entsprechend [Man19[@]] genutzt. [LSV14]

Plug-&-Play Cyber-Physical System Components - nach Jirkovsky et al.

Der Ansatz der Autoren [Jir+18] zielt auf die Realisierung einer Plug-&-Play Funktionalität für Feldgeräte ab, wobei eine Ontologie hierbei die semantische Interoperabilität sicherstellen soll. Die Ontologie beinhaltet Feldgerätebeschreibungen, welche bei Verbindung des Feldgerätes mit dem Netzwerk an ein Steuerungssystem übermittelt und auf Konsistenz geprüft werden. Bei entsprechender Konsistenz der Spezifikation wird die Feldgerätebeschreibung für andere Feldgeräte verfügbar gemacht, sodass die bei Suchen berücksichtigt werden kann. Darüber hinaus werden die von den Feldgeräten generierten Informationen in der Ontologie einbezogen, sodass neben einer statischen Spezifikation auch dynamische Informationen durch das Steuerungssystem abgebildet werden können. Die von den Autoren entwickelte Ontologie, welche im Wesentlichen auf der wiederverwendeten „Semantic Sensor Network Ontology“ entsprechend [Hal+17@] beruht, berücksichtigt strukturelle (Systemstrukturen) und funktionale (Feldgerätefunktionen sowie Messdaten und Parameter) Aspekte. Für die Anforderungserhebung nutzen die Autoren textuelle Applikationsanforderungen aus einem Fallbeispiel, in welchem die Feldgeräte eines Wasserkraftwerks beschrieben werden. [Jir+18]

Intermediate Ontological Layer - nach Ocker et al.

Der Ansatz der Autoren in [OPV19] verfolgt das Ziel, unterschiedliche Informations- bzw. Wissensquellen mit einer Ontologie zu verknüpfen, sodass das Wissen über eine Anlage im Rahmen von Engineering-Aufgaben abgefragt werden kann. Dabei kann es sich beispielsweise um funktionale Fragestellungen handeln (bspw. ob ein Produkt herstellbar ist). Die entwickelte Ontologie soll dabei ermöglichen, dass unterschiedlichste Domänenontologien mit der entwickelten Ontologie vereint werden können. Für die Anforderungserhebung nutzen die Autoren Kompetenzfragen und Antworten und formalisieren auf diese Weise strukturelle (Produktionssysteme und deren Zusammensetzung) und funktionale (Prozesse, Fähigkeiten sowie deren In- und Outputs) Aspekte. Hierbei werden ausschließlich explizite Wissensquellen herangezogen, wobei neben wissenschaftlichen Veröffentlichungen auch Standards verwendet werden. Darüber hinaus findet auch ein Vergleich von existierenden Ontologien bei der Erstellung der TBox statt. [OPV19]

Virtuelle Techniken im industriellen Umfeld – nach Brecher et. al.

Im Ansatz der Autoren in [Bre+11] werden Informationen aus einzelnen Anwendersystemen in eine Ontologie integriert, wobei auch unterschiedliche Datenformate berücksichtigt werden. Dabei wird das Ziel verfolgt, alle Informationen, die während der Entwicklung über ein technisches System anfallen, in einer übergreifenden Semantik zur Verfügung zu stellen, sodass die bisher disjunkt vorliegenden Daten miteinander verknüpft werden. Auf diese Weise soll ein übergreifendes Informationsmodell geschaffen werden, welches eine einheitliche Semantik bietet. Für die Anforderungserhebung wurde eine textuelle Spezifikation von Informationsflüssen in Anwendungsszenarien vorgenommen. Die daraus entwickelte TBox berücksichtigt strukturelle (Informationen aus CAD-Daten des Maschinenbaus und der Elek-

trotechnik), funktionale (mechatronische Funktionen bzw. Systembeschreibungen aus dem PLM System), sowie verhaltensorientierte (Informationen aus automatiserungstechnischen Artefakten) Aspekte und formalisiert insbesondere implizites Wissen der Projektbeteiligten. Für die TBox-Modellierung werden Protégé [Sta19@] sowie TopBraid Composer [Top19@] genutzt. Für die ABox-Erstellung werden insbesondere automatisierte Mappings der jeweiligen Engineering-Artefakte genutzt, wobei ein nicht näher spezifiziertes Werkzeug zum Einsatz kommt. Neben der AVILUS-Upper Ontologie wurden diverse Domänenontologien für die jeweiligen Gewerke im Engineering entwickelt, jedoch nicht veröffentlicht. [Bre+11]

I-3.3. Analyse und Diskussion des Stands der Wissenschaft zu Ontologien im Engineering automatisierter Produktionssysteme

Der vorliegende Abschnitt fasst die Analyse der Ansätze in den beschriebenen Anwendungsfällen von Ontologien im Engineering automatisierter Produktionssysteme anhand der in Abschnitt I-3.1 eingeführten Kriterien zusammen und diskutiert die Ergebnisse.

I-3.3.1. Analyse des Stands der Wissenschaft

Dieser Abschnitt fasst nachfolgend die Analyse des Stands der Wissenschaft mit Hilfe von Tabelle I.4 zusammen. Zunächst sei darauf hingewiesen, dass die Heterogenität der spezifischen Einsatzzwecke zeigt, dass bei der Untersuchung der einschlägigen Veröffentlichungen nicht nur eine Art von spezifischem Einsatzzweck betrachtet wurde. Dies ist auch daran erkennbar, dass der von den jeweiligen Autoren erzielte allgemeine Einsatzzweck (Spezifikation (SP), gemeinsamer Zugriff auf Informationen (GZ), ontologiebbasierte Suche (OS)) nicht mit allen analysierten Veröffentlichungen deckungsgleich ist. Es sei darauf hingewiesen, dass der generelle Einsatzzweck der neutralen Erstellung (NE) in keiner Veröffentlichung erkennbar war. Auf eine Kategorisierung der spezfischen Einsatzzwecke wird aufgrund mangelnder Relevanz für die nachfolgende Analyse verzichtet.

Betrachtet man die von den Autoren angeführte Art der Anforderungserhebung an die entwickelte Ontologie, sofern diese genannt wurde oder erkennbar war, sind zweierlei Kategorien von Anforderungserhebungsmethoden ersichtlich. Die erste Kategorie betrachtet ausschließlich Anforderungen, die aus der Appliaktion resultieren. Diese, dann entweder textuell (TA) oder mit UML-Modellen (SA) spezifizierten Anforderungen an die Applikation, genügen den Autoren dann für die Ontologie-Entwicklung. Die zweite Kategorie nutzt Kompetenzfragen und Antworten (CQ), um aufzuzeigen, welche Fragen auf welche Weise durch die zu entwickelnde Ontologie (i.S.v. Abfragen) beantwortet werden sollen.

In Bezug auf die im Anwendungsfall relevanten Bestandteile der Ontologie werden stets Struktur-, Funktions- und Verhaltensmodelle genutzt. Allerdings variieren die spezifischen Inhalte dieser Modelle teilweise enorm, bspw. betrachten die Autoren von [Lem+06] auch Funktionen, die ausschließlich durch Menschen durchgeführt werden (z.B. Programmierung einer Steuerung), während die Autoren von [Jär+19] lediglich fertigungstechnische Funktionen betrachten. Darüber hinaus scheint es nicht für alle Einsatzzwecke notwendig, ein

Struktur-, Funktions- und Verhaltensmodell zu nutzen, sondern lediglich eines oder zwei der drei Modellarten.

Hinsichtlich der Wissensquellen für die Modellierung der Ontologie sind drei Arten erkennbar: Implizites Wissen der Autoren und beteiligter Projektmitarbeiter bzw. Fachexperten (IE), explizites Wissen einer Domäne (CW), das aus Veröffentlichungen (bspw. wissenschaftliche Veröffentlichungen) entnommen wird, und explizites Wissen, welches aus Veröffentlichungen von Standardisierungsgremien (S) entnommen wird. Es wird ersichtlich, dass implizites Wissen der Autoren meist in Kombination mit einem Standard genutzt wird, um das informale Wissen aus einem Standard zu formalisieren und mit Hilfe von implizitem Wissen an den Einsatzzweck der Autoren anzupassen. In einigen Fällen findet auch eine Literaturrecherche durch die Autoren statt, weshalb auch explizites, nicht standardisiertes Domänenwissen Verwendung findet.

Bezüglich der Wiederverwendung von existierenden Ontologien ist erkennbar, dass sowohl Upper-Ontologien (UO) als auch Domänenontologien (DO) wiederverwendet werden. Allerdings ist der Anteil der Autoren, die eine Domänenontologie wiederverwenden, recht gering, auch wenn ähnliche Teilmodelle (Struktur-, Funktions-, Verhaltensmodell) genutzt werden, oder der spezifische Einsatzzweck sehr ähnlich ist, wie z.B. bei [Jär+19] und [Los13][3] die Beschreibung von Fähigkeiten bzw. Feldgerätefunktionalitäten.

Betrachtet man die Vorgehensweise, wie die entwickelten Ontologien der wissenschaftlichen Gemeinschaft zur Verfügung gestellt werden, sind vier Fälle erkennbar. Im ersten Fall berichten die Autoren lediglich in der jeweiligen Veröffentlichung über die wesentlichen Konzepte der Ontologie. Reichen diese Berichte aufgrund mangelnder Ausführlichkeit, bspw. begründet in der begrenzten Länge eines Konferenzbeitrags, jedoch nicht aus, um die Ontologie als Leser nachzubilden, so wurde dies mit „nicht veröffentlicht“ (nV) bewertet. Liegt eine ausführliche Dokumentation vor, bspw. in Form einer Dissertationsschrift, so wurde dies mit einem „dokumentiert“ (D) bewertet. Im dritten Fall war eine Ontologie lediglich kostenpflichtig (KP) zu erwerben und im vierten Fall wurde die Ontologie online (O) kostenfrei zur Verfügung gestellt.

In Bezug auf die genutzten Methoden zur TBox-Entwicklung, wurden sowohl Eigenentwicklungen (E), als auch die Methoden entsprechend Fernández-López [FGJ97] (M1), Leuchter [Leu06@] (M2), Sure [SSS09] (M3) und Halilaj [Hal+16] (M4) zitiert.

Sofern im Rahmen der TBox-Erstellung Werkzeuge genutzt und benannt wurden, waren folgende Werkzeuge vertreten: Protégé [Sta19@] (W1), TopBraid Composer [Top19@] (W2), sowie durch die Autoren selbst entwickelte Werkzeuge (E). Des Weiteren wurde die OWL-API [Man19@] genannt, welche Java-basiert ist und insbesondere für eine rein softwaretechnische Nutzung geeignet ist. Das gleiche Bild ergibt sich bei der Betrachtung der Werkzeuge für die Erstellung der ABox.

Hinsichtlich der Methoden zur Erstellung der ABox nennen nur wenige Autoren ein Vorgehen, auf welche Weise die ABox in ihren jeweiligen Ansätzen erzeugt werden soll. Lediglich Loskyll [Los13] nennt eine eigens entwickelte Methodik (E), welche jedoch stark manuell ge-

[3]Eine englischsprachige Veröffentlichung ist mit [Los+12] vorhanden

prägt ist und laut Aussage des Autors sehr aufwendig ist. In einigen Ansätzen werden Mappings (MP) genutzt, um existierende Informationsquellen, wie z.B. Engineering-Artefakte oder Datenbanken, automatisiert in die ABox zu überführen.

Tabelle I.4.: Zusammenfassung der Analyse des Stands der Wissenschaft

Referenz	Analysekriterium									
	Allgemeine Kategorie des Einsatzzwecks	Methodik der Anforderungserhebung	Kategorien von Wissen	Wissensgrundlage der Ontologie	Wiederverwendung existierender Ontologien	Verfügbarkeit der ontologischen Artefakte	Genutzte Methodik zur TBox-Modellierung	Genutzte Tools zur TBox-Modellierung	Genutzte Methodik zur ABox-Modellierung	Genutzte Tools zur ABox-Modellierung
[Los+12]	SP/ GZ/ OS	TA	SFV	IE/S	UO/DO	D	M1	W1	E	W1
[Sch13]	SP	TA	SV	IE/S	-	KP	M2	W1/E	-	-
[Jär+19]	SP/ GZ/ OS	CQ	FV	IE/S	-	O	M3	W1	-	W1
[HaVr15]	GZ/ OS	SA	SV	IE/S	-	nV	-	-	-	E
[AVH16]	GZ/ OS	SA	SFV	IE/S/ CW	-	nV	E	W2	-	W2
[Pet+17]	GZ	CQ	SF	IE/S	DO	O	M4	W1/W2	MP	W1/W2
[Sch+12]	SP/ GZ	-	S	IE/S	UO	O	M1	-	-	-
[Lem+06]	SP/ GZ	TA	SF	IE	-	O	-	-	-	-
[LSV14]	SP	TA	SFV	IE	-	nV	-	W3	-	W3
[Jir+18]	SP/ GZ/ OS	-	SF	IE/S	DO	nV	-	-	-	W3
[OPV19]	SP/ GZ/ OS	CQ	SF	CW	UO/DO	nV	E	-	-	-
[Bre+11]	GZ	TA	SFV	IE	UO	nV	-	W1/W2	MP	E

„CQ“ - Kompetenzfragen; „CW“ - explizites Wissen einer Domäne; „D“ - ausführlich dokumentiert; „DO“ - Domänenontologien; „E“ - Eigententwicklung; „GZ“ - gemeinsamer Zugriff auf Informationen; „IE“ - Implizites Wissen; „KP“ - kostenpflichtig; „MP“ - automatisiert mit Mappings; „M1“ - Methode nach [FGJ97]; „M2“ - Methode nach [Leu06@]; „M3“ - Methode nach [SSS09]; „M4“ - Methode nach [Hal+16]; „nV“ - nicht veröffentlicht; „OS“ - ontologiebasierte Suche; „O“ - online verfügbar; „S“ - Veröffentlichungen von Standardisierungsgremien; „SP“ - Spezifikation; „SA“ - Anforderungsmodelle auf UML-Basis; „S/F/V“ - Struktur / Funktion / Verhalten; „TA“ - textuelle Anforderungen; „UO“ - Upper-Ontologie; „W1“ - Werkzeug [Sta19@]; „W2“ - Werkzeug [Top19@]; „W3“ - OWL-API;

I-3.3.2. Diskussion der Analyseergebnisse

In diesem Abschnitt werden nachfolgend die Ergebnisse der Analyse aus dem vorherigen Abschnitt diskutiert. Dabei werden drei Teilbereiche diskutiert, welche sich auf die Anforderungserhebung an die Ontologie, die Entwicklung der TBox und die Entwicklung der ABox beziehen.

Im Rahmen der Entwicklung technischer Systeme gilt: Um ein Entwicklungsprojekt zum Erfolg führen zu können, muss zunächst bekannt sein, was die Anforderungen an das System sind, und diese müssen geeignet dokumentiert sein [PoRu11]. Eine ausführliche Dokumentation der Anforderungen ist unerlässlich für einen effizienten, zielorientierten Entwicklungsprozess, sowie die Möglichkeit zur Überprüfung der Frage, ob die Ziele des Entwicklungsprojektes

erreicht sind, im Sinne einer Verifikation [PoRu11]. Dies gilt nicht nur für die Systementwicklung, sondern auch für die Entwicklung von Ontologien [SGV09]. Wie der Tabelle I.4 zu entnehmen ist, führt keiner der Autoren eine integrative Betrachtung von Anforderungen an die Applikation und Anforderungen an die Ontologie durch, wobei darauf hingewiesen sei, dass auch durch die jeweils genutzten Methoden zur TBox-Entwicklung dies nicht vorgeschlagen wird (siehe Abschnitt III-2). Dies ist insofern problematisch, da die Anforderungen an eine Ontologie sich stets aus einer der sie nutzenden Applikation ergibt. Wie Abschnitt I-2.1.2.4 und I-3.2 gezeigt haben, werden Ontologien nicht als Selbstzweck entworfen, sondern stets entworfen, um die Daten, Informationen und das Wissen für eine bestimmte Applikation zu formalisieren, weshalb eine reine Ermittlung von Kompetenzfragen und Antworten ohne Applikationsbezug keinen sinnvollen Startpunkt besitzen kann. Eine reine Definition der Anforderungen an die Applikation hingegen ist ebenso wenig ausreichend, da ohne eine entsprechende Dokumentation der Anforderungen an die Ontologie das Entwicklungsziel der Ontologie-Entwicklung nicht bzw. nur indirekt definiert ist. Dieser Lücke im aktuellen Stand der Wissenschaft soll sich der erste Methodenteil, Kapitel II dieser Arbeit, widmen.

Wie in [DPW18] beschrieben, besteht ein praktisches Problem hinsichtlich der Wiederverwendung von Ontologien, da viele Autoren für ihren jeweiligen Anwendungsfall neue Ontologien entwerfen, anstatt bestehende Ontologien wiederzuverwenden. Dies ist auch in den betrachteten Veröffentlichungen in Tabelle I.4 erkennbar. Sofern es eine Deckung der drei Teilaspekte Struktur, Funktion und Verhalten zwischen zwei Anwendungsfällen gibt, sollte eine Wiederverwendung in Betracht gezogen werden, da die Annahme besteht, das ähnliche Sachverhalte betrachtet werden. Eine Wiederverwendung, abseits der Referenzierung von Top-Level Ontologien, findet jedoch nur selten statt (siehe Eintrag von „DO“ für Domänenontologie). Dies steht im Widerspruch zum häufig angeführten Problem des hohen Aufwands bei der Erstellung einer Ontologie [Dot+18; DiLi15], da anzunehmen ist, dass die Wiederverwendung den Aufwand der Erstellung senkt. Ein Grund hierfür kann die Veröffentlichungspraxis der jeweiligen Autoren sein, welche den Zugang zur Ontologie erschweren, indem sie die Ontologie nur ausschnittsweise textuell bzw. grafisch spezifiziert veröffentlichen. Dies wurde auch von [YCY19b] als wesentlicher Hinderungsgrund für die Wiederverwendung von Ontologien benannt, da der Detailgrad einer textuellen bzw. grafischen Spezifikation naturgemäß nicht dem der eigentlichen Ontologie entsprechen kann. Ein weiterer Grund kann die Komplexität der resultieren Ontologien sowie aus implizitem Wissen resultierende Design-Entscheidungen sein [GaPr09]. Alle der in Tabelle I.4 untersuchten Ansätze stellten die entwickelten Ontologien stets als ein gesamtes Entwicklungsresultat dar, wobei die Ontologien umso größer waren, je höher der notwendige Detailgrad und der notwendige Modellumfang (Struktur, Funktion, Verhalten).

Dies führt auf das in [GaPr09] erwähnte Komplexitätsproblem. Wenn ein Projekt einen ähnlichen, aber nicht identischen Einsatzzweck hat, müssen die für den spezifischen Einsatzzweck relevanten Konzepte aus der Gesamtmenge der Konzepte einer existierenden Ontologie extrahiert werden. Je größer die Ontologie, desto aufwendiger ist dieser Prozess und erschwert somit die Wiederverwendung. Aus implizitem Wissen resultierende Design-Entscheidungen, welche nicht dokumentiert werden und keiner Quelle entnommen werden können, erschweren die Wiederverwendung zusätzlich, da sie nur schwer nachvollziehbar sind. Das Problem der Wiederverwendung wurde auch in [Dot+18] angeführt, wobei die Autoren argumentieren, dass ohne eine anerkannte Basis von Ontologien die Akzeptanz der Technologie in der

Domäne der Automatisierungstechnik enorm erschwert ist. Dem Problembereich der TBox-Entwicklung soll sich der zweite Methodenteil, Kapitel III dieser Arbeit, widmen.

In [Los13] wird beschrieben, dass die Erstellung der ABox ein vergleichsweise aufwendiger, manuell geprägter Prozess ist. Betrachtet man die für die ABox-Erstellung genutzten Werkzeuge in Tabelle I.4, so wird die ABox mit Ausnahme von Mapping-basierten Ansätzen stets mit Werkzeugen erstellt, die ein hohes Maß an ontologischem Wissen erfordern [Blf+17], also an Ontologie-Experten gerichtet sind. Dies kann als ein Grund für die in [DiLi15] beschriebene Problematik gelten, dass der Umgang mit Ontologien in der Automatisierungstechnik noch nicht strukturiert genug ist, um im industriellen Umfeld, also durch nicht-Ontologie-Experten eingesetzt zu werden. Erschwerend kommt hinzu, dass außer Mapping-basierten Ansätzen und Eigenentwicklungen keine Methodik zur ABox-Erstellung von den Autoren in Tabelle I.4 erkennbar war, was ebenfalls den Einsatz im industriellen Umfeld erschweren dürfte. Insbesondere dann, wenn ein Mapping-basierter Ansatz nicht oder nur teilweise in Frage kommt. Dem Problembereich der ABox-Entwicklung soll sich der dritte Methodenteil, Kapitel IV dieser Arbeit, widmen.

I-4. Lösungsansatz, Forschungsteilziele und Anforderungen

Die nachfolgenden Abschnitte erläutern, basierend auf der Analyse des Stands der Wissenschaft in Abschnitt I-3, die Forschungsteilziele dieser Arbeit. Hierzu wird Abschnitt I-4.1 zunächst eine Überblick über das Lösungskonzept dieser Arbeit gegeben, bevor die Abschnitte I-4.2 - I-4.4 die Forschungsteilziele dieser Arbeit sowie die daraus abgeleiteten Anforderungen darstellen.

I-4.1. Methodischer Überblick

Der im Rahmen dieser Arbeit verfolgte Lösungsansatz ist in Abbildung I.12 dargestellt und orientiert sich an dem in [StSt09] beschriebenen allgemeinen Vorgehen zur Entwicklung einer Ontologie und wird mit Hilfe sogenannter Methodenbausteine entsprechend [Dau+16a] beschrieben. Jeder Methodenbaustein adressiert eine der in Abschnitt I-3.3.2 identifizierten Lücken. Darüber hinaus hat jeder Methodenbaustein ein- und ausgehende Entwicklungsartefakte, welche entweder bei Durchführung eines Entwicklungsprojektes bereits existent sind oder im Rahmen eines Entwicklungsprojektes erstellt werden. Des Weiteren können Artefakte entweder informal, semi-formal oder formal sein. Als informal werden im Rahmen dieser Arbeit all jene Artefakte bezeichnet, welche rein textuell oder grafisch ohne Modellierungsvorschrift erstellt wurden. Als semi-formal werden all jene Artefakte bezeichnet, welche einer Metamodellierungsvorschriften folgen (bspw. der UML oder SysML, siehe hierfür [WeSo14]), welche maschinenlesbar ist. Letztlich werden als formal im Rahmen dieser Arbeit OWL-Ontologien bezeichnet. Betrachtet man den Lösungsansatz in Abbildung I.12, so sind drei Methodenbausteine ersichtlich, welche in den Teilen II-IV dieser Arbeit behandelt werden. Je ein Methodenbaustein wird somit durch je ein Forschungsteilziel adressiert, welche in den nachfolgenden Abschnitten I-4.2 - I-4.4 näher beschrieben werden.

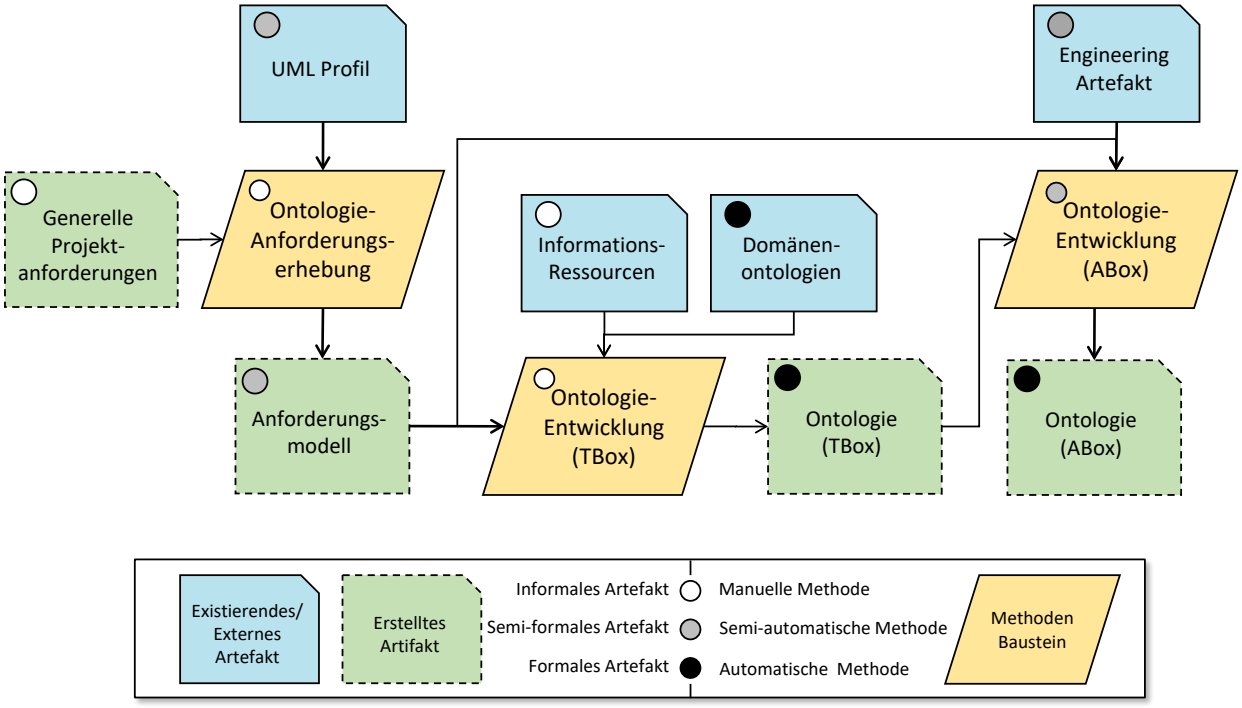

Abbildung I.12.: Methodischer Lösungsansatz der vorliegenden Arbeit

I-4.2. Forschungsteilziel I: Ontologie-Anforderungserhebung

Das Forschungsteilziel I der vorliegenden Arbeit ist in Definition I.2 dargestellt und bezieht sich auf eine Methode zur Anforderungserhebung an Ontologien.

Definition I.2 (Forschungsteilziel I): *Entwurf einer Methode zur Definition und Dokumentation ontologischer Anforderungen, mit deren Hilfe sowohl die Entwicklung der TBox zielgerichtet durchgeführt als auch eine Verifikation der resultierenden Entwicklungsergebnisse durchgeführt werden kann. Dabei soll soweit möglich auf eine modellbasierte Entwicklungsmethodik zurückgegriffen werden, um die Anforderungserhebung der Ontologie in den Entwicklungsprozess der sie nutzenden Systeme zu integrieren. Darüber hinaus soll die Methodik hinsichtlich des notwendigen Aufwands zur Durchführung möglichst schlank gestaltet sein, sodass sie beispielsweise im Rahmen von Projekt-Workshops durchführbar ist.*

Aus dem in Definition I.2 genannten Forschungsteilziel lassen sich funktionale (FA) und nicht-funktionale (NFA) Anforderungen für das zu entwickelnde Artefakt (A1) ableiten, welche nachfolgend erläutert werden.

A1.FA1 Verknüpfung der Entwicklungsziele von Ontologie und Applikation: Im Rahmen der Systementwicklung kann ein Entwicklungsziel definiert werden, um eine explizite Betrachtung der Stakeholder-Intentionen vorzunehmen und somit das Entwicklungsziel des Systems an den Zielen der Stakeholder auszurichten [PoRu11]. Im Rahmen der Ontologie-Entwicklung werden aus dem gleichen Grund Ziele modelliert [FGJ97]. Unter der Annahme, dass die zu entwickelnde Ontologie ein technisches Problem lösen soll, welches sich aus einem ebenfalls zu entwickelnden System ergibt, ist das Entwicklungsziel des Systems struk-

turiert mit dem Entwicklungsziel der Ontologie zu verknüpfen. Auf diese Weise wird eine Verknüpfung der Entwicklung der Ontologie mit der Entwicklung des Systems hergestellt, da sie über eine Hierarchie miteinander verbunden sind. Ändern sich die Entwicklungsziele des Systems, können sich demnach auch die Entwicklungsziele der Ontologie ändern.

A1.FA2 Nutzung oder Erweiterung von Artefakten der Systementwicklung: Unter der Annahme, dass bei der Systementwicklung ein modellbasierter Entwicklungsansatz basierend auf Modellen der UML / SysML erfolgt, können diese Artefakte für die Ontologie-Anforderungserhebung herangezogen werden. Dabei sollen die Artefakte hinsichtlich des Daten-, Informations- bzw. Wissensbedarfs des betrachteten Systems analysiert werden und basierend darauf die Anforderungen an die Ontologie abgeleitet werden. Auf diese Weise soll die Anzahl an Ontologie-spezifischen Anforderungsartefakten so gering wie möglich gehalten werden und stattdessen eher eine Integration der ontologischen Anforderungen in die Artefakte der Systementwicklung vollzogen werden.

A1.FA3 Modellierung von Funktionsstrukturen und Funktionsverhalten (Informationsfluss): Unter der Annahme, dass im Rahmen der System-Anforderungserhebung eine Funktionsmodellierung durchgeführt wird, können Funktionsmodelle geeignet sein, um den Informationsaustausch von Systemen, welcher den Informationsbedarf der Ontologie determiniert, für die Ontologie-Entwicklung zu spezifizieren. Hierbei kann der Informationsfluss zwischen verschiedenen Funktionalitäten in einer statisch strukturellen Perspektive und einer ablauforientierten Perspektive modelliert werden [Wey10]. Beides ist notwendig, da nicht unbedingt alle Inhalte der zu entwickelnden Ontologie statisch vorliegen, sondern evtl. durch die Ausführung von Funktionen eines Systems modifiziert werden können.

A1.FA4 Nachverfolgbarkeit ontologischer Anforderungen: Im Rahmen der Entwicklung eines technischen Systems können Änderungen an den entwickelten Artefakten des Systems entstehen, bspw. durch veränderte Anforderungen der Stakeholder oder technische Erkenntnisse im Entwicklungsprozess [UTY16]. Wurden Artefakte eines Systems für die Herleitung ontologischer Anforderungen genutzt, welche sich verändert haben, können sich daraus auch Änderungen an den ontologischen Anforderungen ergeben. Daher sind die Anforderungen an die Ontologie so aufzunehmen, dass eine Nachverfolgbarkeit ontologischer Anforderungen entsteht. Auf diese Weise soll bei Änderungen an den relevanten System-Artefakten der Einfluss auf die ontologischen Artefakte evaluiert werden können.

A1.NFA1 Leichtgewichtiger Ansatz, Nutzung einer domänenspezifischen Sprache: Unter der Annahme, dass bei der Systementwicklung ein modellbasierter Entwicklungsansatz basierend auf Modellen der UML / SysML erfolgt, kann ein Profil durch simple Integration in die Modell-Toolbox die notwendigen Modellinhalte für die Definition ontologischer Anforderungen liefern, ohne dass ein Software-Entwickler oder Domänenexperte neue Modellarten erlernen muss. Daher ist ein UML-Profil zu entwerfen, welches für die Anforderungserhebung genutzt werden kann. Leichtgewichtig soll jedoch auch die Durchführung der Methodik sein und auch im Rahmen von Expertenworkshops durchführbar sein, sodass die notwendigen Fachexperten nicht unverhältnismäßig stark involviert werden.

A1.NFA2 Abstraktion von realen Systemen: Eine Ontologie soll den Daten-, Informations- bzw. Wissensbedarf eines zu entwickelnden Systems abdecken [JaUs99], ent-

sprechend dem jeweiligen Anwendungsfall. Um die Anforderungen an die zu entwickelnde Ontologie zu erheben, ist eine Analyse der die Ontologie nutzenden Systeme unabdingbar [GFC04]. Da jedoch die Entwicklung einer Ontologie mit einem nicht unerheblichen Aufwand verbunden ist [Dot+18] und somit eine längere Zeit in Anspruch nehmen kann, würde eine Wasserfallstrategie zu höheren Projektlaufzeiten führen. Um dem entgegen zu wirken, soll im Rahmen der Anforderungserhebung eine Abstraktion von realen Systemen erfolgen, sodass die Ontologie-Entwicklung nicht zwangsläufig erst in einer späten Phase der Systementwicklung beginnen muss und stattdessen auch in frühen Phasen der Systementwicklung (z.B. Anforderungserhebung) beginnen kann.

I-4.3. Forschungsteilziel II: Ontologie-Entwicklung TBox

Das Forschungsziel II dieser Arbeit ist in Definition I.3 dargestellt und bezieht sich auf eine Entwicklungsmethode für die TBox einer Ontologie.

Definition I.3 (Forschungsteilziel II): *Entwurf einer Methode zur Entwicklung der terminologischen Box (TBox) einer Ontologie, die nachweislich (verifizierbar) den mit der Methodik aus Forschungsteilziel I (Definition I.2) spezifizierten Anforderungen genügt. Dabei soll die zu entwickelnde Methodik einen besonderen Fokus auf die Wiederverwendung der erstellten ontologischen Artefakte legen und das Wissen von Domänenexperten als Grundlage nutzen.*

Aus dem oben genannten Forschungsteilziel II lassen sich funktionale (FA) und nichtfunktionale (NFA) Anforderungen für das zu entwickelnde Artefakt (A2) ableiten, welche auf den Veröffentlichungen [Hil+18*a; Hil+18*b] beruhen und nachfolgend erläutert werden.

A2.FA1 Modularisierung erstellter TBoxen: Modularisierung ist ein bewährtes Konzept im Rahmen der Entwicklung technischer Systeme, um einzelne Einheiten zu kapseln und somit deren Wiederverwendung in anderen Anwendungsfällen zu steigern [ElM06]. Wie in Abschnitt I-3.3.2 angeführt, sind die Ontologien der analysierten Veröffentlichungen stets nur als ein „großes“ Modell verfügbar. Die Wiederverwendung dieser Ontologie ohne größere Veränderungen ist dann auf den exakt gleichen Anwendungsfall begrenzt. Unter der Annahme, dass das Konzept der Modularisierung auch auf die Entwicklung von Ontologien anwendbar ist, sind die durch die Methodik zu erstellenden Ontologien modular aufzubauen, um die Wiederverwendung der resultierenden Ontologien zu erhöhen.

A2.FA2 Modularisierung erstellter ABoxen: Unter der Annahme, dass eine modulare Erstellung der TBox von Ontologien möglich ist, soll anhand der TBox-Module eine Herleitung von verschiedenen Modulen der ABox im Sinne von Best Practices bzw. Vorlagen zur Nutzung der TBox möglich sein. Diese Anforderung ist der Entwicklungsmethodik der TBox zuzuordnen, da bei der Entwicklung einer Terminologie bereits deren Verwendung vorgeschlagen werden sollte. Dies mit dem Ziel, die Wiederverwendung durch eine kürzere Einarbeitungszeit in die Nutzung der Ontologie zu fördern.

A2.FA3 Dokumentation: Wie in Abschnitt I-3.3.2 angeführt, kann die Veröffentlichungspraxis von Ontologien die Wiederverwendung erschweren. Hinzu kommt,

dass Ontologien als Form eines Wissensmodells im Allgemeinen veränderlich sind und Versionierung unterworfen sind, ebenso wie Industriestandards von Zeit zu Zeit in neuen Versionen erscheinen. Die Methodik soll daher explizit den einfachen Zugang zu aktuellen Versionen ermöglichen und die Dokumentation bzw. Dissemination explizit berücksichtigen.

A2.FA4 Nutzung von Informationsressourcen der Automatisierungstechnik: Wie in [Hil+17*a] angeführt, beinhalten insbesondere Industriestandards bereits viel Wissen, welches für die Entwicklung einer Ontologie herangezogen werden kann. Wie in Abschnitt I-3.3.1 erkennbar, nutzen viele Autoren das Wissen aus Industriestandards. Deren Nutzung hat jedoch einen weiteren Vorteil hinsichtlich der Wiederverwendung. Unter der Annahme, dass bei der Erstellung einer Ontologie aus einem Industriestandard die wesentlichen, oder alle, Konzepte übernommen und nicht inhaltlich verändert werden, dient der Industriestandard als Dokumentation der Inhalte der Ontologie und ist somit die i.d.R. leichter verständliche, informelle Variante der Ontologie. Um die Wissensquellen der Domäne der Automatisierungstechnik zu nutzen und zur Steigerung der Wiederverwendung der resultierenden Ontologien sind daher Industriestandards als bevorzugte Informationsressource zur Entwicklung der Ontologie im Rahmen der Methodik zu wählen.

A2.NFA1 Domänenexperten-zentrische Entwicklung: Ein wesentliches Problem bei der Entwicklung von Ontologien ist der Mangel an Ontologie-Experten in der industriellen Praxis [NiMi16; Pet+17]. Darüber hinaus können Ontologie-Experten nicht das nötige Wissen über alle Wissensbereiche besitzen, die in einem Entwicklungsprojekt von Relevanz sein können. Im Gegensatz dazu besitzen Domänenexperten meist ein detailliertes Wissen über relevante Industriestandards und deren Inhalte. Um den Mangel an Ontologie-Experten zu berücksichtigen und die Stärke existierender Domänenexperten zu nutzen, soll die Methodik Domänenexperten-zentrisch sein. Das bedeutet, dass Domänenexperten so viele Tätigkeiten im Ontologie-Entwicklungsprozess wie möglich einnehmen sollen.

A2.NFA2 Nutzung von Beschreibungsmitteln, welche für alle Stakeholder verständlich sind: Wie in [JaUs99] erläutert, sind verschiedene Stakeholder an der Entwicklung einer Ontologie beteiligt. Um dem Hintergrundwissen der unterschiedlichen Stakeholder Rechnung zu tragen, müssen unterschiedliche Abstraktionsebenen bei der Beschreibung der ontologischen Artefakte genutzt werden. Wie in Abschnitt I-2.2.3 beschrieben, bieten sich hier auf dem ontologischen Spektrum mehrere Möglichkeiten an. Da Lightweight-Ontologien in Form von UML-Klassendiagrammen sowohl von Domänenexperten verstanden werden (siehe z.B. [Pfr+16]) als auch von Ontologie-Experten genutzt werden [JSL13], stellen sie einen angemessenen Kompromiss aus Ausdrucksstärke und Verständlichkeit für den Zweck der Präsentation und Diskussion ontologischer Artefakte dar. Die Methodik soll daher als Zwischenstufe Lightweight-Ontologien als Artefakt erzeugen, welche als Diskussionsbasis und zur Dokumentation in Projekten eingesetzt werden können.

I-4.4. Forschungsteilziel III: Ontologie-Entwicklung ABox

Das Forschungsziel III dieser Arbeit ist in Definition I.4 dargestellt und bezieht sich auf eine Entwicklungsmethode für die ABox einer Ontologie.

Definition I.4 (Forschungsteilziel III): *Entwurf einer Methode zur Entwicklung der assertionalen Box (ABox) einer Ontologie, mit deren Hilfe die ABox einer Ontologie soweit möglich durch Domänenexperten und Software-Entwickler erstellt werden kann. Dabei soll die Methodik je nach Projektanforderungen automatische und semi-automatische Bestandteile berücksichtigen und damit über ein spezifisches manuelles oder automatisiertes Modellierungsverfahren hinaus ein übergreifendes Vorgehen bereitstellen.*

Aus dem oben genannten Forschungsteilziel III lassen sich funktionale (FA) und nichtfunktionale (NFA) Anforderungen für das zu entwickelnde Artefakt (A3) ableiten, welche nachfolgend erläutert werden.

A3.FA1 Formalisierung von Daten, Informationen und Wissen aus Planung und Betrieb: Die Daten, Informationen und das Wissen, welche über ein System formalisiert werden sollen und von einer bestimmten Applikation konsumiert werden sollen, können zu unterschiedlichen Zeitpunkten entstehen. So wird die Struktur des Systems i.d.R. einmal zur Entwicklungszeit definiert und evtl. bei späteren Rekonfigurationen angepasst. Messwerte hingegen, welche bspw. den Zustand des Systems repräsentieren, werden zur Betriebszeit erzeugt und höchstens zur Entwicklungszeit beschrieben. Die zu entwickelnde Methode soll Daten, Informationen und Wissen über ein System aus Entwicklung und Betrieb formalisieren. Dabei soll die jedoch insbesondere die korrekte Aufbereitung der ABox-Inhalte für den jeweiligen Anwendungskontext betrachtet werden, nicht aber die spätere Nutzung der ABox-Inhalte.

A3.FA2 Einbindung automatisierter Ansätze: Wie in Abschnitt I-3.3 zu erkennen, nutzen bereits einige Arbeiten in der Domäne der Automatisierungstechnik Mappingansätze, um bspw. aus Engineering-Artefakten die ABox der Ontologie automatisiert zu erstellen. Da, wie in Abschnitt I-2.1.2.2 dargestellt, eine Vielzahl an Datenformaten im Rahmen des Engineerings existiert und, wie Abschnitt IV-2 darstellen wird, auch bereits eine Vielzahl an Mappingansätzen existiert, soll die zu entwickelnde Methodik eine systematische Einbindung existierender Mappingansätze vornehmen. Auf diese Weise soll bei entsprechender Eignung ein existierender Mappingansatz ausgewählt werden können, um im Rahmen eines Projektes zur Erstellung der ABox genutzt zu werden.

A3.FA3 Semi-automatisierte ABox-Erstellung: Unter der Annahme, dass die Artefakte, die durch Mappings in ABox-Inhalte überführt werden sollen, entweder nicht vorliegen oder in einer nicht automatisiert überführbaren Struktur vorliegen (bspw. aufgrund der Nutzung proprietärer Datenformate), muss die Methodik mindestens eine semi-automatisierte Erstellung der ABox-Inhalte vorsehen. Auf diese Weise sollen auch bei nicht-Vorliegen von für eine automatisierte Erstellung geeigneten Engineering-Artefakten eine zumindest semi-Automatisierung von repetitiven Modellierungstätigkeiten möglich sein.

A3.FA4 Serialisierung der ABox in unterschiedlichen Formaten: In unterschiedlichen Entwicklungsumgebungen sind unterschiedliche Datenformate üblich. Im Engineering-Kontext hat sich beispielsweise XML zur Modellierung von Engineering-Informationen etabliert, während in der Web-Entwicklung oftmals JSON-basierte Formate zum Datenaustausch zwischen Applikationen und APIs genutzt werden, da ihre Verwendung in einer Javascript-basierten Umgebung vergleichsweise einfach und schlank ist. Um den Anwendungskontext der

zu entwickelnden Methode nicht einzuschränken, soll die entwickelte ABox in unterschiedliche Formate (mindestens XML und JSON) serialisiert werden können.

A3.FA5 Integration in das Engineering von fertigungstechnischen Produktionssystemen: Da im Rahmen der Methodik eine ABox für ein automatisiertes, fertigungstechnisches System erstellt werden soll, ist eine methodische Integration in ein bestehendes Engineering-Vorgehen vorzunehmen. Dies erfolgt mit dem Ziel, einen Leitfaden für die Modellierung der ABox zu erstellen, welcher sich möglichst leicht in bestehende Abläufe integrieren lässt. Als Referenz für den bestehenden Ablauf soll der funktionale Ansatz entsprechend [Sch18] genutzt werden, da dieser bereits einen gewerkeübergreifenden Ansatz zur Informationsmodellierung bereitstellt, welcher lediglich um Ontologie-spezifische Aspekte ergänzt werden muss.

A3.FA6 Modellierung struktureller, funktionaler und verhaltensorientierter Fakten eines Produktionssystems: Im Fokus der vorliegenden Arbeit ist die Domäne der Automatisierungstechnik. Wie in Abschnitt I-2.1 dargestellt, lässt sich ein Produktionssystem in drei wesentlichen Sichten beschreiben, welche sukzessive im Engineering des Produktionssystem erarbeitet werden. Die vorliegende Methodik soll die Abbildung der im Rahmen des Engineerings erarbeiteten Sichten in der ABox ermöglichen.

A3.NFA1: Erzeugung konsistenter und kohärenter ABox-Inhalte: Unter der Annahme, dass nicht Ontologie-Experten, sondern Domänenexperten und Software-Entwickler letztendlich die ABox erstellen, muss die Methode sicherstellen, dass nur TBox-konforme ABox Inhalte erstellt werden, sodass die Konsistenz der ABox-Inhalte sichergestellt ist. Darüber hinaus muss die Methodik sicherstellen, dass die Verknüpfung (Kohärenz) der einzelnen Tripel in der ABox so sichergestellt wird, dass unerwünschte Verknüpfungen vermieden und Erwünschte ermöglicht werden.

A3.NFA2: Erzeugung verständlicher ABox-Inhalte: Unter der Annahme, dass Domänenexperten und Software-Entwickler mit dem Aufbau von Graphen-basierten Datenformaten nicht vertraut sind, muss die Methodik eine den Präferenzen der Nutzer entsprechende Aufbereitung der ABox, bspw. mit einem Software-Werkzeug, vornehmen. Dies kann bspw. tabellarisch oder, falls die TBox dies zulässt, grafisch sein. Dies erfolgt mit dem Ziel, den Inhalt der ABox verständlich aufzubereiten, sodass Domänenexperten und Software-Entwickler zu deren Erstellung befähigt werden.

II. Ontologie-Anforderungserhebung

Das vorliegende Kapitel widmet sich der Entwicklung einer Menge von Artefakten zur Erfüllung von Forschungsteilziel I. Hierzu stellt Abschnitt II-1 zunächst die Grundlagen vor. Abschnitt II-2 stellt dann den relevanten Stand der Wissenschaft zur Anforderungserhebung an Ontologien dar. Im Anschluss widmet sich Abschnitt II-3 der Entwicklung eines UML-Profils, welches essentieller Bestandteil der Methode zur Erreichung des Forschungsteilziels I ist. In Abschnitt II-4 wird dann eine Methode vorgestellt, mit deren Hilfe zunächst die Ziele der Ontologie- und Systementwicklung aufgenommen, anschließend die Anforderungen an die Systementwicklung und schließlich die Anforderungen an die Ontologie-Entwicklung aufgenommen werden können.

II-1. Grundlagen zur modellbasierten Anforderungserhebung

Der vorliegende Abschnitt definiert Grundlagen zur modellbasierten Anforderungserhebung. Zur Strukturierung dieser Grundlagenabschnitte werden nachfolgende Wissensfragen genutzt:

1. Wie werden Anforderungen an ein System hergeleitet und welche Betrachtungsweisen sind hierbei von Relevanz?
2. Welche Perspektiven müssen bei der Herleitung von Anforderungen betrachtet werden?
3. Welche Modellierungssprachen können hierfür in Frage kommen?
4. Was ist ein UML-Profil und wie wird es erstellt?
5. Was sind relevante Begriffe und Zusammenhänge im Rahmen der Herleitung von Anforderungen?

Zur Beantwortung der ersten Frage erläutert Abschnitt II-1.1 den Begriff des „Kontexts“ eines Systems. Dieser Begriff wird weiter diversifiziert in Abschnitt II-1.2 zur Beantwortung der Frage 2. Abschnitt II-1.3 stellt anschließend Modellierungssprachen und UML-Profile zur Beantwortung der Fragen 3 bis 4 dar. Abschließend führt Abschnitt II-1.4 eine Auswahl von Metamodellen ein, welche relevante Begriffe und Zusammenhänge darstellen, die für die Herleitung von Anforderungen herangezogen werden können.

II-1.1. System und Systemkontext

Die modellbasierte Spezifikation von Anforderungen an technische Systeme entsprechend [PoRu11] betrachtet die Dokumentation von Anforderungen durch den Einsatz von definierten Modellsemantiken mit dem Ziel, dass eine Anforderungsdokumentation durch verschiedene Stakeholder auf die gleiche Weise interpretiert wird. Zur Charakterisierung der Anforderungen an Systeme im Allgemeinen ist der Begriff des Kontexts von hoher Relevanz, welcher im Zusammenhang mit dem Engineering alles bezeichnet, was den Lösungsraum im Entwicklungsprozess einschränkt, ohne dabei selbst ein expliziter Teil der zu entwickelnden Lösung zu sein [Wey10]. Oftmals wird der Begriff auch plakativ beschrieben, wie z.B. in [Gau05] als: „the fully relevant environment to which we are designing“. Der Kontext definiert somit die Umgebung, in welche das zu konstruierende System eingebettet werden soll. Ursprünglich entstammt die Betrachtung von Systemen und Systemkontextgrenzen der Softwareentwicklung und findet beispielsweise Anwendung im Requirements Engineering [PoRu11], aber auch im Bereich des Systems Engineering zum Beispiel in [DTW12] oder unter der Bezeichnung „Umfeld“ in [GaBe12].

Es ist die Aufgabe des Requirements Engineerings im Rahmen der System- und Kontextabgrenzung, das System von seiner relevanten und irrelevanten Umgebung abzugrenzen. Auf diese Weise sollen Bereiche identifiziert werden, die Anforderungen an ein „System under Development“ (SUD) stellen. Dabei werden materielle und immaterielle Aspekte identifiziert, denen eine Beziehung zum System zugeordnet werden kann. Es wird eine Sollperspektive eingenommen, bei welcher eine Annahme über die zukünftige Integration des Systems in die Realität getroffen wird. Somit ist es notwendig, einen Realitätsausschnitt zu identifizieren, der einen relevanten Einfluss auf das System und auf seine Anforderungen hat. Der für die Anforderungen des Systems relevante Ausschnitt der Realität wird daher als Systemkontext bezeichnet, wobei er denjenigen Teil der Systemumgebung definiert, der für die Definition und das Verständnis der Anforderungen des betrachteten Systems relevant ist. Der Ursprung der Anforderungen eines Systems liegt somit im Systemkontext des geplanten Systems. Bspw. fordern Stakeholder die Verwendung einschlägiger Standards im Rahmen der Entwicklung oder gesetzliche Richtlinien bestimmen funktionale Merkmale oder Qualitäten, die das SUD an seinen Schnittstellen aufweisen soll. Eine Anforderung ist also in einem spezifischen Kontext definiert und kann auch nur für diesen Kontext richtig interpretiert werden. [PoRu11]

II-1.2. Perspektiven des operationellen Kontextes

Entsprechend [Dau+16c] kann der Systemkontext unterschieden werden in einen Wissenskontext (englisch: „context of knowledge“) und den operationellen Kontext (englisch: „operational context“). Der Wissenskontext beschreibt insbesondere Aspekte, welche die Entwicklung des SUD selbst beeinflussen, z.B. Gesetze, Normen und Richtlinien sowie ethische Aspekte. Im Rahmen dieser Arbeit ist jedoch insbesondere der operationelle Kontext von Interesse. Der operationelle Kontext ist definiert als „der Teil des Systemkontexts, der im Betrieb des Systems entweder Ursprung von Stimuli des Systems ist oder auf den sich die Wirkung der Reaktionen des Systems bezieht“ [Wey10]. Im Rahmen des Requirements Engineerings wird, analog zur System-Betrachtung durch [Pat82], eine Komplexitätsreduktion

durch die Einführung einzelner Perspektiven als sinnvoll erachtet [Tar03]. Dabei werden in jeder Perspektive bestimmte Abstraktionen genutzt, die die relevanten Eigenschaften jeweils so verkürzen, dass insgesamt eine möglichst überschneidungsfreie Betrachtung aller Eigenschaften des Systems in einzelnen Sichten entsteht. Entsprechend [Wey10] können diese Perspektiven unterteilt werden in eine statisch-strukturelle, ablauforientierte sowie zustandsorientierte Perspektive. Diese Perspektiven entsprechend inhaltlich der Unterteilung in Struktur (statisch-strukturelle Perspektive), Funktion (ablauforientierte Perspektive) und Verhalten (zustandsorientierte Perspektive) entsprechend [Pat82], siehe auch Abbildung II.1.

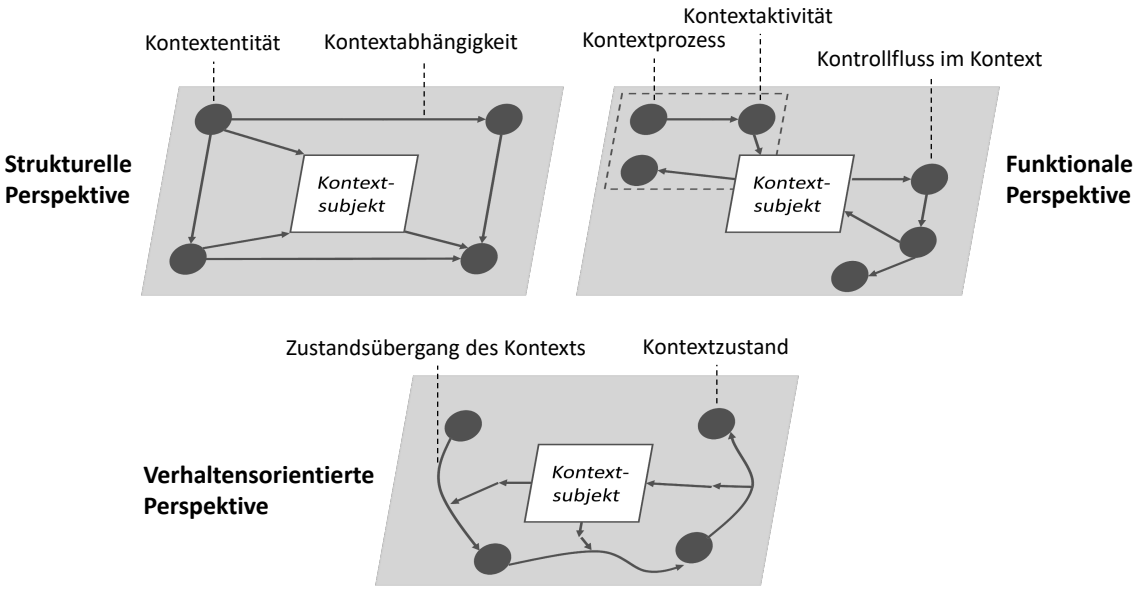

Abbildung II.1.: Perspektiven auf den operationellen Kontext in Anlehnung an [Wey10]

In jeder Perspektive werden Kontextaspekte betrachtet, welche für das SUD von Relevanz sind, da diese Aspekte entweder Ursprung eines Stimulus an das SUD oder Ziel einer Einwirkung durch das SUD sind. Die in der funktionsorientierten Perspektive betrachteten Kontextaspekte sind Kontextaktivitäten. Letztere stehen durch einen Kontrollfluss miteinander in Beziehung, wobei das SUD in diesen Kontrollfluss integriert ist. Die in der strukturellen Perspektive betrachteten Kontextaspekte sind Kontextentitäten, welche materielle oder immaterielle Objekte, Personen oder technische Systeme im Umfeld des SUD sein können. Zwischen den Kontextentitäten können Beziehungen im Sinne von Abhängigkeiten bestehen. Die in der verhaltensorientierten Perspektive betrachteten Kontextaspekte sind Kontextzustände, welche dadurch charakterisiert sind, dass sie eine spezifische Ausprägung von Eigenschaften besitzen, welche der operationelle Kontext des SUD zu einem bestimmten Zeitpunkt besitzen kann. Kontextzustände stehen über Zustandsübergänge miteinander in Beziehung, wobei Kontextaktivitäten einen Zustandsübergang bedingen können. [Wey10]

II-1.3. Modellierungssprachen im Kontext der modellbasierten Entwicklung

Wie in Abschnitt I-2.1.1 dargestellt, werden im Rahmen des Systems Engineerings unter Nutzung eines modellbasierten Ansatzes bestimmte Modellierungssprachen genutzt. Die nachfolgenden Abschnitte enthalten eine kurze Vorstellung der Modellierungsmöglichkeiten der UML, SysML, sowie Erweiterungsmöglichkeiten mit Hilfe des UML-Profilmechanismus.

II-1.3.1. Unified Modeling Language

Die Unified Modeling Language (UML) bezeichnet eine visuelle Modellierungssprache, die für Systemarchitekten, Software-Ingenieure und Software-Entwickler eine Menge an Modellen für die Analyse, das Design und die Implementierung von Software-basierten Systemen bieten soll [Obj17@]. UML ist von der Object Management Group (OMG) in [Obj17@] standardisiert und stellt folgendes bereit:

- Eine formale Definition eines gemeinsamen „Meta Object Facility“ (MOF) basierten Metamodells (siehe unten), welches die abstrakte Syntax der UML spezifiziert, wobei die abstrakte Syntax die existierenden Modellkonzepte, -attribute und -relationen sowie Regeln zu deren Nutzung definiert.

- Eine detaillierte Beschreibung der Semantik jedes UML-Modellkonzeptes, welche technologieunabhängig beschreibt, wie UML-Konzepte implementiert werden sollen.

- Eine Spezifikation von menschenlesbaren Notationselementen für die Repräsentation individueller UML-Modellkonzepte, sowie Regeln für deren Kombination in einer Vielzahl unterschiedlicher Diagrammarten, korrespondierend zur Vielzahl unterschiedlicher Aspekte eines zu modellierenden Systems.

Dabei liefert der Standard [Obj17@] eine Vielzahl an Diagrammen zur Modellierung struktureller und verhaltensorientierter Sichtweisen auf Systeme. Für alle Diagrammarten wird in [Obj17@] definiert, wie diese in XML Metadata Interchange (XMI) überführt werden, was einen Austausch zwischen verschiedenen Modellierungswerkzeugen ermöglicht. Um die Semantik der einzelnen Modellkonzepte eindeutig zu spezifizieren, bedient sich die OMG hierbei des Konzepts der Metamodellierung. Hierbei wird mit verschiedenen Modellierungsebenen die Semantik eines Modells in der nächsttieferen Ebene durch ein Metamodell dargestellt, wobei entsprechend [Obj16@] zwar eine unbegrenzte Menge an Modellierungsebenen beschrieben werden kann, sich in der Praxis aber ein 4-Ebenenkonzept durchgesetzt hat. Letzteres stellt das „Meta Object Facility“ (MOF) als oberste Ebene dar und die mit den Diagrammen beschrieben Software als unterste Ebene [Hen11], siehe auch Abbildung II.2. Demnach definiert das MOF als Metametamodell die Semantik aller Metamodelle, wobei bei der Erstellung eines Metamodells (z.B. Metamodell des UML-Klassendiagramms) Modellkonzepte des Metametamodells instanziiert werden. Bei der Erstellung eines Modells werden dann Instanzen des Metamodells angewandt (z.B. Klassen, Assoziationen etc.) um die nächsttiefere Ebene zu beschreiben, in diesem Falle Software (bspw. eine Klasse in der Programmiersprache Java).

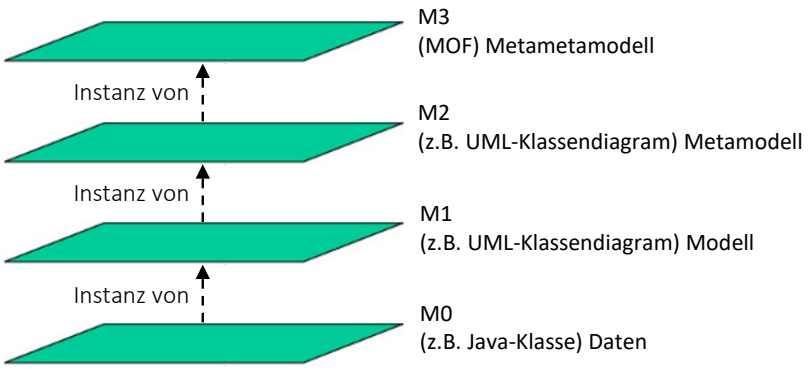

Abbildung II.2.: 4-Ebenenkonzept der OMG in Anlehnung an [Hen11]

II-1.3.2. System Modeling Language (SysML)

Die System Modeling Language (SysML) erweitert die UML zur Beschreibung von Systemen, die neben Software auch Hardware beinhalten, wodurch neben dem Software-Engineering auch das Systems Engineering mit den Domänen Automotive, Luftfahrt, Kommunikation und Informationssysteme angesprochen werden sollen [Obj18@]. SysML kann darüber hinaus auch zur Beschreibung mechatronischer Systeme im Bereich der Automatisierungstechnik genutzt werden [Vog+14b]. SysML nutzt dabei bestehende Diagramme der UML und erweitert diese über den Profilmechanismus des Standards [Obj17@] (siehe auch nachfolgender Abschnitt), führt aber auch neue Diagrammarten ein, namentlich das Anforderungsdiagramm und das Parameterdiagram.

II-1.3.3. Profile im Kontext von UML

Wie im vorangegangenen Abschnitt erwähnt, nutzt die OMG zur Definition der Semantik aller Diagramme das Metamodellierungskonzept. Bei der Erstellung einer neuen Modellsemantik kommt laut [FuVa04] entweder ein „Heavyweight“- oder „Lightweight“-Ansatz in Frage, wobei für einen Heavyweight-Ansatz für eine neue Modellart auch ein neues Metamodell, oder sogar Metametamodell, entworfen werden muss und für einen Lightweight-Ansatz die existierenden Metamodelle der OMG erweitert werden. Werden Metamodelle der OMG erweitert um eine Anpassung für bestimmte Zwecke zu erzielen oder die Modellierungsfähigkeiten eines Diagramms generell zu erweitern, wird dies mit Hilfe eines Profils durchgeführt [Obj17@]. Ein prominentes Beispiel für dieses Verfahren ist SysML, welches eine Erweiterung zu UML darstellt und über den Profilmechanismus definiert ist [WeSo14]. Ein Profil kann dabei lediglich neue Modellelemente in einem Metamodell (z.B. Metamodell des UML-Klassendiagramms) ergänzen oder die Nutzung bestehender oder neuer Modellkonzepte weiter eingrenzen, jedoch können die Bestandteile der standardisierten Metamodelle nicht entfernt oder verändert werden [Obj17@]. In Anlehnung an [FuVa04] können folgende Modellkonzepte zur Erstellung eines Profils genutzt werden:

- Stereotypen: Ein Stereotyp ist definiert durch eine Menge von Metamodellelementen, zu denen es hinzugefügt werden kann.

- Constraints: Einschränkungen (englisch: „Constraints“) können die Nutzung von existierenden Metamodellelementen oder Stereotypen beschränken.
- Tagged Values: Markierte Werte (englisch: „tagged values“) sind zusätzliche Attribute, die einem Stereotypen oder Metamodellelement hinzugefügt werden.

Ein vereinfachter Ablauf zur Erstellung von Profilen ist in [FuVa04] dargestellt und besteht aus den folgenden vier Schritten:

- Definition eines Metamodells über die zu ergänzenden Modellkonzepte sowie Relationen, Attribute und Einschränkungen zwischen den Modellkonzpten
- Definition von Stereotypen für alle relevanten Modellkonzepte und Relationen
- Definition von Attributen des Metamodells als “tagged values”
- Definition von Einschränkungen der Nutzung der Stereotypen mit Hilfe der Object Constraint Language, welche in [Obj14@] definiert ist

II-1.4. Metamodelle mit Relevanz für die Anforderungserhebung

Die nachfolgenden Abschnitte stellen Metamodelle vor, welche im Rahmen der Anforderungserhebung genutzt werden, um Begriffe zu strukturieren und deren Zusammenhänge aufzuzeigen.

II-1.4.1. Funktionsmetamodell nach Ludewig et al.

In [Lud+18] wird ein Metamodell vorgestellt, welches zur Beschreibung von Funktionen im Rahmen der modellbasierten Entwicklung genutzt werden kann. Dabei leiten die Autoren die Klassen und Beziehungen des Metamodells aus dem Systems Engineering, dem Maschinenbau und dem Software-Engineering her und begründen somit deren Relevanz zur Beschreibung von Systemen in verschiedenen Domänen. Die Autoren verweisen bei der Erläuterung des Metamodells explizit darauf, dass bei der Beschreibung von Funktionen eine strukturelle und verhaltensorientierte Perspektive von Relevanz ist. Die strukturelle Perspektive betrachtet die Verknüpfung von Funktionen untereinander sowie die Nachrichten („Message“), die zwischen den Ports der Funktionen ausgetauscht werden. Das Funktionsverhalten bezieht sich hingegen auf die Abhängigkeiten zwischen Funktionen, z.B. im Sinne von Vorbedingungen, die erfüllt sein müssen, damit eine Funktion ausgeführt werden kann. Auf eine mögliche Implementierung des Metamodells, z.B. in SysML-Diagrammen, gehen die Autoren nicht ein. [Lud+18]

II-1.4.2. Funktionsmetamodell nach Lamm und Weilkiens

In [LaWe10] wird ein SysML-basierter Ansatz vorgestellt, welcher zur Modellierung von funktionalen Architekturen in SysML genutzt werden kann. Die Autoren führen hierzu ein Metamodell ein. Der Fokus des Modells liegt dabei auf der Beschreibung von Funktionshierarchien und den Informations-, Energie- und Materialflüssen zwischen den Funktionen. Zur Erkennung relevanter Funktionen sollen zunächst UML-Use-Case-Diagramme erstellt werden, für welche dann im nächsten Schritt SysML-Blockdiagramme in einer hierarchischen Struktur erstellt werden. Letztere dient dann als Funktionshierarchie. Mit Hilfe von internen Blockdiagrammen können dann Informations-, Energie- und Materialflüsse abgebildet werden. [LaWe10]

II-1.4.3. Funktionsmetamodell nach Scholz

In [Sch18] wird ein Funktionsmetamodell vorgestellt, welches im wesentlichen auf Elementen der Formalisierten Prozessbeschreibung entsprechend [VDI/VDE 3682-1] beruht. Als grafische Notation wird im Ansatz des Autors auch die Formalisierte Prozessbeschreibung genutzt, wobei dies mit dem Ziel geschieht, Funktionen fertigungstechnischer Produktionssysteme zu modellieren, welche dann zur Dokumentation von Anforderungen an das fertigungstechnische Produktionssystem dienen. Dabei nutzt der Autor das Funktionsmodell, um es mit einem Struktur- und einem Verhaltensmodell zu verbinden, wodurch eine integrierte Beschreibung verschiedener Sichten auf das fertigungstechnische Produktionssystem gelingt. [Sch18]

II-1.4.4. Strukturmetamodell nach Regnat et al.

Das in [Reg+18] vorgestellte Metamodell zur Beschreibung der Struktur von Kollaborativen Systemgruppen entstammt dem Projekt CrESt (Collaborative Embedded Systems), welches die Entwicklung von softwareintensiven kollaborierenden Embedded Systems (KES) fokussiert. Der grundlegende Gedanke stellt eine (kollaborative) Systemgruppe in den Vordergrund, welche einen bestimmten Zweck erfüllen soll. Eine Systemgruppe besteht aus mehreren KES, welche einen Beitrag zur Zielerreichung leisten können. Ob ein KES dazu in der Lage ist, entscheidet die Tatsache, ob das KES eine bestimmte Rolle in der Systemgruppe einnehmen kann. Die Rollen dienen dabei lediglich als eine Art „Anforderungscontainer“, welcher die Anforderungen an das KES so spezifiziert, dass die Erfüllung der Summe aller Anforderungen aus allen Rollen die Erreichung des Ziels der Systemgruppe ermöglicht. Eine Rolle kann dabei entweder optional oder verpflichtend sein. Die Definition der Struktur der Systemgruppe über Rollen hat dabei den entscheidenden Vorteil, dass für die Beschreibung der Systemgruppe lediglich die Spezifikation von Anforderungen notwendig ist und keinerlei Spezifikation der eigentlichen Systeme. Auf diese Weise kann die Gestalt der Systemgruppe vergleichsweise lösungsneutral beschrieben werden. [Reg+18]

II-1.4.5. Kontextmetamodell nach Daun et al.

In [Dau+16b] wird ein Metamodell zur Beschreibung des Kontextes von Software-intensiven Embedded Systems vorgestellt, welches im Wesentlichen die in Abschnitt II-1.1 vorgestellten Aspekte beschreibt. Im Rahmen der Beschreibung mittels des Metamodells wird zunächst eine Trennung in System und Kontext vorgenommen. Der Kontext wird dann weiterhin unterteilt in einen Wissenskontext und einen operationellen Kontext. Des Weiteren besteht der Kontext aus Kontextobjekten, während in der Kontextbetrachtung das betrachtete System als Kontextsubjekt bezeichnet wird. Der Kontext und die irrelevante Umgebung bestimmen dann die Gesamtheit der Umgebung. [Dau+16b]

II-2. Stand der Wissenschaft zur Anforderungserhebung an Ontologien

In den nachfolgenden Abschnitten II-2.1 bis II-2.4 werden bestehende Ansätze zur Ontologie-Entwicklung, bzw. die darin enthaltene Methodik zur Ontologie-Anforderungserhebung, vorgestellt. Anschließend erfolgt in Abschnitt II-2.5 eine Bewertung hinsichtlich ihres Beitrages zur Erreichung von Forschungsteilziel I (siehe Abschnitt I-4.2), sodass die für die vorliegende Arbeit relevanten Aspekte wiederverwendet werden können.

Die Strukturierung der existierenden Ansätze wird nachfolgend erläutert. Entsprechend [SiLu14] kann in Bezug auf Ontologie-Entwicklungsmethoden grundsätzlich in zwei Hauptkategorien von Ansätzen unterschieden werden, dezentrale und zentrale Entwicklungsansätze. Dezentrale Ansätze, welche insbesondere in jüngeren Publikationen zu finden sind [KoVo06; AuHe07; Bra+07], berücksichtigen die Ontologie-Anforderungserhebung nicht in dem Sinne, dass die für die Realisierung eines spezifischen Anwendungsfalls relevanten Anforderungen an die Ontologie erhoben werden, sondern konzentrieren sich generell auf die Formalisierung des Wissens einer Domäne. Aufgrund des mangelnden Bezugs zu einem zu entwickelnden System werden nachfolgend lediglich Ansätze berücksichtigt, welche in einer zentralen Entwicklungsumgebung (bspw. von einem Projektteam) eingesetzt werden können. Letztere können in folgende Kategorien unterteilt werden:

- Ansätze mit einem Fokus auf die Definition von Prozessschritten, siehe Abschnitt II-2.1
- Ansätze mit einem Fokus auf dokumentenbasierte Aufnahme und Strukturierung von Ontologie-Anforderungen, siehe Abschnitt II-2.2
- System-orientierte Ansätze, welche ein zu entwickelndes System als Grundlage für die Ontologie-Entwicklung definieren, siehe Abschnitt II-2.3
- Testgetriebene Ansätze, welche das Test-Driven-Design Konzept auf die Ontologie-Entwicklung im Allgemeinen anwenden, siehe Abschnitt II-2.4

In den folgenden Abschnitten II-2.1 bis II-2.4 wird stets nur ein Vertreter der jeweiligen

Kategorie vorgestellt, wobei auf weitere Vertreter verwiesen wird.

II-2.1. Ontologie-Anforderungserhebung nach Uschold und Gruninger

Die in [UsGr96] vorgestellte Methode steht stellvertretend für eine Kategorie von frühen Ontologie-Entwicklungsmethoden (siehe z.B. auch [FGJ97; GrFo95]), welche relevante Grundlagen definieren, die sich auch in späteren Arbeiten wiederfinden und auch für die vorliegende Arbeit von Relevanz sind. Die in [UsGr96] vorgestellte Entwicklungsmethode für Ontologien (siehe Abbildung II.3) nutzt zur Anforderungserhebung eine Kombination aus Szenarien und Kompetenzfragen (englisch: „Competency question“, kurz „CQ“) sowie dazugehörigen Antworten. Die Szenarien können dabei als motivierende Szenarien verstanden werden, welche von Industriepartnern vorgestellt werden und einen Bedarf nach expliziter, semantischer Spezifikation bestimmter Sachverhalte aufzeigen [UsGr96]. Dabei können mehrere Szenarien vorgestellt werden, wobei deren Spezifikation keinem Formalismus unterliegt.

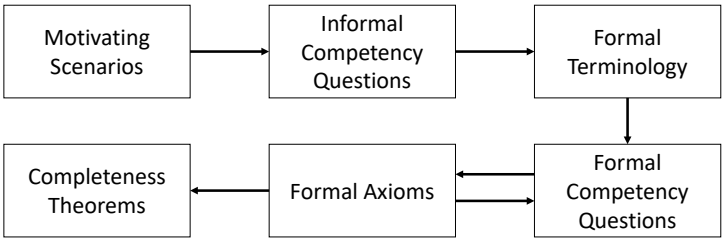

Abbildung II.3.: Ontologieentwicklungsprozess [UsGr96], basierend auf [GrFo95]

Diese Szenarien können dann untersucht werden, um Kompetenzfragen und Antworten zu erheben, welche als Anforderungen an die semantische Modellierung neuer Begriffe und Relationen oder Abwandlung bestehender Ontologien dienen. Die Autoren schlagen darüber hinaus auch vor, Hierarchien von Fragen zu bilden, wobei für die Beantwortung einer Frage mehrere Unterfragen beantwortet werden müssen. Abbildung II.4 stellt die Struktur einer Kompetenzfrage dar, wobei ersichtlich wird, dass Kompetenzfragen auch Kompetenzfragen bedingen können („Rationale“). Bestandteile einer Kompetenzfrage sind die notwendigen Eingangsdaten, Annahmen, Beschränkungen und soweit möglich auch Abfragen an bestimmte involvierte Systeme. [UsGr96]

II-2.2. Ontologie-Anforderungserhebung nach Suárez-Figueroa et al.

Der in [SGV09] vorgestellte Ansatz steht stellvertretend für Ansätze (siehe z.B. auch [SSS09; Sta+01]), welche auf der Kategorie von Anforderungserhebungsmethodik aus Abschnitt II-2.1 aufbauen und durch eine dokumentenbasierte Aufnahme und Strukturierung die Ontologie-Anforderungen erheben.

Hierzu wird in [SGV09] ein Vorgehen zur Erhebung von Anforderungen an eine Ontologie beschrieben. Die Anforderungen an die Ontologie werden mit Hilfe von Kompetenzfragen und

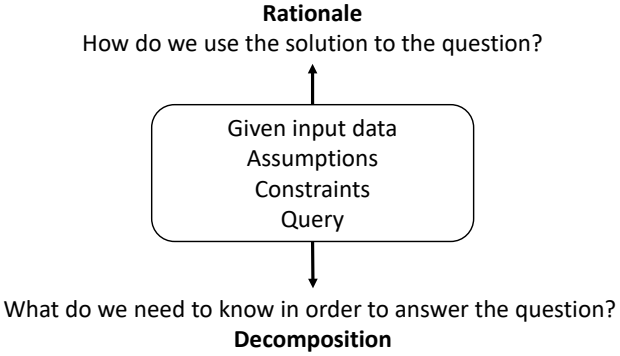

Abbildung II.4.: Struktur einer Kompetenzfrage [UsGr96]

Antworten erhoben, wobei jede Kompetenzfrage als eine informelle Art von Abfrage an die spätere Ontologie verstanden werden kann. Die dazugehörige Antwort dient dementsprechend als Rückgabe dieser Anfrage. An der Anforderungserhebung beteiligt sind Nutzer der Ontologie, Domänenexperten und Ontologen. Ausgangspunkt für die Definition der ontologischen Anforderungen ist ein „Set of ontological needs“, welches durch Interviews und Befragungen der Domänenexperten sowie der späteren Nutzer der Ontologie in Erfahrung gebracht wird. Um von diesen Anforderungen auf die Kompetenzfragen und Antworten zu kommen, werden mehrere Schritte durchgeführt, welche der sukzessiven textuellen Detaillierung dienen. Hierzu werden unter anderem der Zweck der Ontologie, Gruppen von Endnutzern und Nutzungsarten identifiziert und textuell spezifiziert. Des Weiteren enthält der Ansatz Tätigkeiten zur Verwaltung der anfallenden Resultate (z.B. Priorisierung oder Validierung). [SGV09]

Der Ansatz der Nutzung von Kompetenzfragen und Antworten wird in [SPH12] aufgegriffen, um die Ontologie während des Erstellungsprozesses hinsichtlich der Anforderungserfüllung zu verifizieren. Dies wird mit Hilfe von SPARQL-Abfragen (siehe auch Abschnitt III-1.4) durchgeführt, wobei die Abfragen einer Formalisierung der Kompetenzfragen entsprechen, deren Rückgaben den gewünschten Antworten entsprechen. [SPH12]

II-2.3. Ontologie-Anforderungserhebung nach Nicola et al.

Der in [NMN09] vorgestellte Ansatz steht stellvertretend für System-orientierte Ansätze (siehe z.B. auch [NiMi16; Leu06@; ADH19]), welche ein zu entwickelndes System als Grundlage für die Ontologie-Entwicklung definieren und ein Vorgehen basierend auf Konzepten des Software-Engineering zugrunde legen.

Der Ansatz in [NMN09], welcher in Abbildung II.5 dargestellt ist, verbindet die modellbasierte Anforderungserhebung des technischen Systems mit der Anforderungserhebung an Ontologien, indem ähnlich des Ansatzes von [FGJ97] zunächst die betrachtete Domäne sowie der Umfang festgelegt werden. Im nächsten Schritt wird das Ziel der Ontologie-Entwicklung mit dem Ziel der Systementwicklung über einen gemeinsamen „Business Purpose“ verbunden, der als Fließtext definiert wird. Basierend darauf werden Storyboards definiert, welche die Nutzung der Ontologie textuell beschreiben. Dabei wird eine Sequenz von Aktivitäten beschrieben, in welchen die Ontologie eingesetzt werden soll. Aus den Fließtexten der Story-

boards wird anschließend ein Lexikon mit relevanten Begriffen erstellt, welches das Vokabular für die Erstellung von Kompetenzfragen und Antworten liefert. Nachdem die ontologischen Anforderungen spezifiziert wurden, werden in einer anschließenden Analysephase die Anwendungszenarien mit Hilfe von UML-Modellen erstellt, wobei Use-Case, Aktivitäts- und Klassendiagramme zum Einsatz kommen. Die Begriffe des Lexikons werden dabei in den erstellten Diagrammen wiederverwendet. Die für die Bearbeitung benötigten Informationen werden aus Interviews von Domänenexperten sowie Applikations-spezifischen Dokumenten gezogen. Die beteiligten Stakeholder sind Domänenexperten und Wissensingenieure, wobei Domänenexperten in frühen Schritten und Wissensingenieure in der späteren Schritten des Prozesses stärker involviert sind.

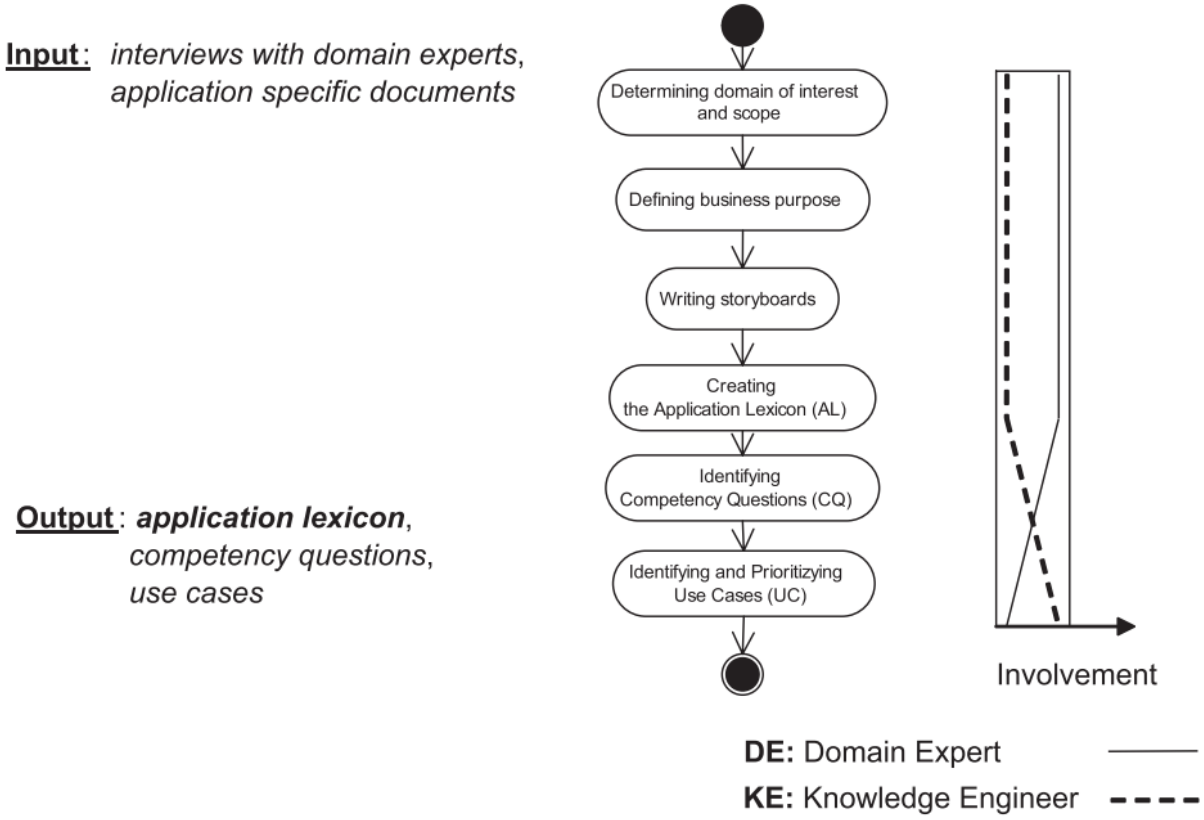

Abbildung II.5.: Erhebung ontologischer Anforderungen nach [NMN09]

II-2.4. Ontologie-Anforderungserhebung nach Peroni

Der in [Per17] vorgestellte Ansatz steht stellvertretend für Test-getriebene Entwicklungsansätze (englisch: „Test-Driven-Design“, siehe z.B. auch [VrGa06; Pre+09; KeŁa16]) für die Ontologie-Entwicklung. Diese Kategorie von Ansätzen stellt eine Art Spezialfall der Software-Engineering-basierten Ansätze dar, wobei hier das Konzept des Test-Driven-Design des Software-Engineering (siehe [JaSa05]) auf die Ontologie-Entwicklung angewandt wird.

Der Ansatz in [Per17] besteht im Wesentlichen aus drei Schritten: (1) der Definition eines

Testfalls, welcher auch die Anforderungen an die Ontologie beinhaltet, (2) Verknüpfung einer Teil-Ontologie für den Testfall mit der bisher erstellten Ontologie, (3) Überarbeitung der bisherigen Testfälle, falls Anpassungen aufgrund der Integration einer Teil-Ontologie notwendig sind. Wesentlich für die Anforderungserhebung ist der erste Schritt. Hierzu wird zunächst ein motivierendes Szenario in textueller Form spezifiziert. Aus diesem motivierenden Szenario wird ein Glossar von Begriffen sowie eine Menge von Kompetenzfragen und Antworten extrahiert, wobei auch hier eine rein textuelle Spezifikation vorgenommen wird. Diese Kompetenzfragen und Antworten werden direkt, ohne die Existenz einer TBox, in SPARQL-Abfragen überführt und somit formalisiert. Motivierendes Szenario, Kompetenzfragen und Antworten, das Glossar und die SPARQL-Abfragen stellen dann einen Testfall für die zu entwickelnde Ontologie dar. Im Rahmen des zweiten Schritts gilt es dann, eine TBox als Teil-Ontologie zu entwickeln, welche ein positives Ergebnis für den vorher definierten Test liefert. Hierzu wird solange an der Teil-Ontologie gearbeitet, bis diese ein positives Ergebnis für den Testfall liefert. [Per17]

II-2.5. Bewertung und Auswahl der Ontologie-Anforderungserhebungsmethoden

Die in den bisherigen Abschnitten vorgestellten Ansätze zur Ontologie-Anforderungserhebung werden in diesem Abschnitt dahingehend untersucht, inwiefern sie für die Erreichung des Forschungsteilziels I herangezogen werden können. Hierzu werden die Ansätze mit den Anforderungen abgeglichen, welche für die Erreichung des Forschungsteilziels I erfüllt sein müssen. Basierend auf diesem Abgleich wird diskutiert, welchen spezifischen Beitrag ein Ansatz für die Erreichung des Forschungsteilziels I im Rahmen dieser Arbeit leistet. Die Gegenüberstellung zwischen Ansätzen und Anforderungen ist in Tabelle II.1 zusammengefasst, wobei die Einträge nachfolgend Zeilenweise erläutert werden.

Tabelle II.1.: Bewertung der Ontologie-Anforderungserhebungsmethoden

Referenz	**Anforderung**					
	A1.FA1	A1.FA2	A1.FA3	A1.FA4	A1.NFA1	A1.NFA2
Uschold und Gruninger [UsGr96]	(X)	-	-	-	n.a	X
Suárez-Figueroa et al. [SGV09]	-	-	(X)	(X)	(X)	-
Nicola et al. [NMN09]	X	X	X	-	(X)	-
Peroni [Per17]	(X)	-	-	-	n.a	X

„X“ - erfüllt; „(X)“ - teilweise erfüllt; „-“ - nicht erfüllt; „n.a.“ - nicht anwendbar

Der Ansatz von [UsGr96] nutzt motivierende Szenarien als Grundlage für die Herleitung von Kompetenzfragen und Antworten. Diese Szenarien können eine Verknüpfung des Entwicklungsziels der Ontologie und der Applikation (A1.FA1) darstellen. Dies ist allerdings nur dann zutreffend, wenn die gewählten Szenarien auch die Ziele der Applikation widerspiegeln

und aus diesen Zielen dann Ziele für die Ontologie-Entwicklung abgeleitet werden können. Da dies methodisch nicht berücksichtigt ist, ist die Anforderungen (A1.FA1) nur teilweise erfüllt. Der Ansatz der Nutzung von Kompetenzfragen und Antworten, welche aus den motivierenden Szenarien abgeleitet werden, entstammt der Arbeit von [UsGr96] und ist noch immer Gegenstand aktueller Forschung (siehe [KHH17]). Dies ist in der Tatsache begründet, dass Kompetenzfragen und Antworten als informelle Anforderungspezifikation keinerlei Voraussetzungen hinsichtlich der Modellierungsfähigkeiten eines Stakeholders definieren und sich dennoch vergleichsweise gut in formelle Anforderungen in Form von SPARQL-Anfragen überführen lassen, wie in [SPH12] dargestellt. Aus diesem Grund werden Kompetenzfragen und Antworten auch im Rahmen der vorliegenden Arbeit zur Modellierung der funktionalen Anforderungen an die zu entwickelnde Ontologie genutzt. Die Abstraktion von realen Systemen (A1.NFA2) ist im Ansatz [UsGr96] als erfüllt anzusehen, da der Bezug zu Applikationen lediglich über textuelle, motivierende Szenarien erfolgt und keinerlei Bezug zu Entwicklungsartefakten besteht. Die Existenz eines realen Systems ist somit nicht notwendig. Die verbleibenden Anforderungen sind als nicht erfüllt zu betrachten, da keine Artefakte der Systementwicklung genutzt oder erweitert werden (A1.FA2) und somit auch keine Modellierung von Funktionsstrukturen und Funktionsverhalten erfolgt (A1.FA3). Folglich ist die Beurteilung der Anforderung des leichtgewichtigen Ansatzes bzw. der Nutzung einer domänenspezifischen Sprache (A1.NFA1) nicht möglich. Darüber hinaus wird eine Nachverfolgbarkeit ontologischer Anforderungen nicht in Betracht gezogen (A1.FA4).

Von besonderer Relevanz für die vorliegende Arbeit sind die Ansätze von [SGV09] und [SPH12]. Zwar wird in den genannten Arbeiten keine explizite Verknüpfung der Entwicklungsziele von System und Ontologie vorgenommen (A1.FA1), auch wird nicht von realen Systemen abstrahiert (A1.NFA2), was die Existenz des Systems im Rahmen der Ontologie-Entwicklung erfordert. Darüber hinaus werden auch keine Artefakte der Systementwicklung herangezogen (A1.FA2). Allerdings ist das in [SGV09] vorgestellte Vorgehen sehr strukturiert und bietet erste Anknüpfungspunkte für die Modellierung von Systemfunktionalitäten (A1.FA3) mit Hilfe von UML-Modellen. Dabei bezieht sich die Modellierung jedoch lediglich auf UML-Anwendungsfalldiagramme und somit die strukturelle Sicht auf den Informationsfluss, weshalb die Anforderung als teilweise erfüllt anzusehen ist. Die Nutzung der UML-Modelle im Rahmen der Anforderungspezifikation wird darüber hinaus nicht konsequent verfolgt, da für die Nachverfolgung ontologischer Anforderungen (A1.FA4) ein manuell zu pflegendes Textdokument genutzt wird. Der Ansatz erfüllt die Anforderung (A1.NFA1) nur teilweise, da der Ansatz insofern leichtgewichtig ist, als das er im Rahmen von Workshops einsetzbar ist, jedoch für die Verwendung der UML-Modelle keinerlei domänenspezifische Sprache einführt. Zusätzlich angemerkt sei, dass der in [SPH12] vorgestellte Verifikationsansatz im weiteren Verlauf dieser Arbeit aufgegriffen wird.

Des Weiteren besitzt die Arbeit von [NMN09] besonderes Gewicht für die vorliegende Arbeit, aufgrund der engen methodischen Verknüpfung mit der Entwicklung des die Ontologie nutzenden Systems. Die Verknüpfung des Entwicklungsziels der Ontologie und der Applikation (A1.FA1) wird über eine Definition des „business purpose“ der Applikation vorgenommen, anhand welcher die nachfolgenden Tätigkeiten der Anforderungserhebung an die Ontologie ausgerichtet sind. Des Weiteren werden konsequent Artefakte aus der Systementwicklung genutzt und Anforderungen an die Ontologie daraus extrahiert (A1.FA2). Die Modellierung der Funktionsstrukturen der Applikation erfolgt mit Hilfe von UML-Anwendungsfalldiagrammen

und die Modellierung des Funktionsverhaltens der Applikation erfolgt mit Hilfe von UML-Sequenzdiagrammen, weshalb Anforderung A1.FA3 als erfüllt zu betrachten ist. Die Abstraktion von realen Systemen (A1.NFA2) ist nicht erkennbar, was dazu führt, dass die Ontologie-Entwicklung erst beginnen kann, wenn die Spezifikation der Struktur und des Verhalten des Systems bereits in einem fortgeschrittenen Stadium sind. Darüber hinaus ist im Ansatz von [NMN09] keine modellbasierte Spezifikation ontologischer Anforderungen vorgesehen, was dazu führt, dass für diese keine Nachverfolgbarkeit gegeben ist (A1.FA4). Da keine modellbasierte Spezifikation der ontologischen Anforderungen vorgesehen ist, ist auch Anforderung (A1.NFA1) nur als teilweise erfüllt anzusehen. Dies jedoch nur unter der Annahme, dass die Erstellung der UML-Modelle in Workshops durchführbar ist. Im Rahmen dieser Arbeit wird der Ansatz von [NMN09] zur Verknüpfung der Entwicklungsziele von Ontologie und System aufgegriffen und erweitert. Auch wird die Vorgehensweise zur Modellierung der Funktionalitäten des Systems aufgegriffen und entsprechend erweitert.

Die Test-getriebenen Ansätze, stellvertretend anhand [Per17] vorgestellt, erfüllen lediglich teilweise die Anforderungen nach der Verknüpfung von Entwicklungszielen der Applikation und Ontologie, da sie wie im Ansatz von [UsGr96] motivierende Szenarien als Ausgangspunkt nutzen. Da keinerlei Nutzung von Artefakten der Systementwicklung erkennbar ist, sind die Anforderungen A1.FA2 und A1.FA3 als nicht erfüllt anzusehen. Aufgrund mangelnder Berücksichtigung von Artefakten der Systementwicklung und der Nutzung von motivierenden Szenarien ist jedoch analog zur Bewertung von [UsGr96] die Abstraktion von realen Systemen als erfüllt anzusehen. Des Weiteren wird keine Nachverfolgbarkeit ontologischer Anforderungen (A1.FA4) betrachtet, weder durch strukturierte textuelle Dokumentation noch durch Modellierung der Anforderung und Verknüpfung mit Artefakten der Systementwicklung. Da keine Artefakte der Systementwicklung betrachtet werden, kann A1.NFA1 nicht angewandt werden. Die Test-getriebenen Ansätze, stellvertretend anhand [Per17] vorgestellt, werden im Rahmen der Methodik zur Anforderungserhebung lediglich dahingehend berücksichtigt, dass sich die motivierenden Szenarien im Sinne der modellierten System-Anforderungen wiederfinden. Eine vollständige Berücksichtigung eines Test-getriebenen Ansatzes wird an dieser Stelle als nicht zielführend erachtet, da eine Formalisierung der Kompetenzfragen in Testfälle im Sinne von SPARQL-Abfragen zu diesem Zeitpunkt als verfrüht anzusehen ist. Dieser Sachverhalt wird im Abschnitt III-2 nochmals aufgegriffen.

II-3. Metamodell und UML-Profile zur modellbasierten Anforderungserhebung

In den folgenden Abschnitten werden die für die Anforderungserhebung an System und Ontologien zu nutzenden UML-Profile hergeleitet. Hierzu wird zunächst in Abschnitt II-3.1 das zugrundeliegende Metamodell eingeführt, welches auf den in Abschnitt II-1.4 dargestellten Arbeiten beruht, bzw. aus diesen hergeleitet wird. Auf Basis der Konzepte des Metamodells werden in Abschnitt II-3.2 die für die Anforderungserhebung relevanten Profile hergeleitet.

II-3.1. Begriffliche Definitionen und Metamodell

In den nachfolgenden Abschnitten wird schrittweise das Metamodell vorgestellt, welches auf den in Abschnitt II-2 vorgestellten Arbeiten beruht und für die Erstellung des UML-Profils in Abschnitt II-3.2 genutzt wird. Hierzu wird Abschnitt II-3.1.1 zunächst auf die im Rahmen dieser Arbeit angewandte Beschreibung von System und Kontext eingehen. Im Anschluss definiert Abschnitt II-3.1.2 die relevanten Aspekte zur Beschreibung funktionaler Beziehungen. Abschnitt II-3.1.3 stellt dann abschließend die Beschreibung ontologischer Anforderungen vor.

II-3.1.1. Beschreibung von System und Kontext

In Bezug auf die Modellierung von System und Kontext wird nachfolgend das in Abbildung II.6 dargestellte Metamodell beschrieben. Das Metamodell basiert im Wesentlichen auf dem Kontextmodell nach [Dau+16b] und dem Strukturmetamodell nach [Reg+18]. Das Kontextmodell nach [Dau+16b] wurde ergänzt um eine Assoziation `exchanges data / information / knowledge with`, welche zum Ausdruck bringt, dass Kontextsubjekt und Kontextobjekt über einen Austausch von Daten, Information oder Wissen interagieren. Darüber hinaus wird angenommen, dass die Betrachtung von Kontextobjekten und Kontextsubjekten nur für Systeme durchgeführt wird. Des Weiteren wird im weiteren Verlauf der Arbeit lediglich der operationelle Kontext eines Kontextsubjekts betrachtet, welcher in Anlehnung an [Wey10] in Definition II.1 formuliert und nachfolgend kurz Kontext genannt wird.

Definition II.1 (Operationeller Kontext kollaborativer Systeme): *Der operationelle Kontext eines Systems bezieht sich auf den Teil des Kontexts des Systems, welcher zu einem spezifischen Zeitpunkt zur Betriebszeit des Systems von Relevanz für die Funktionserfüllung des Systems ist. Dabei kann der operationelle Kontext in struktureller, funktionaler und verhaltensorientierter Sichtweise beschrieben werden.*

Da die Betrachtung von Kollaborativen eingebetteten Systemen nicht im Fokus dieser Arbeit steht, wurde aus dem Metamodell nach Regnat der Begriff der Kollaboration sowie die Trennung in Kollaboration und Kooperation entfernt. Beibehalten wurde das Konzept der Systemgruppe, welche sich aus einer definierten Anzahl von Rollen zusammensetzt, welche optional oder obligatorisch erforderlich sein können. Definition II.2 stellt den Begriff der Rolle in Anlehnung an die Verwendung im Rahmen der Automation Markup Language (AML) entsprechend [DIN EN 62714-1] dar und Definition II.3 den Begriff der Systemgruppe, welcher auf dem Rollenbegriff aufbaut.

Definition II.2 (Rolle): *Eine Rolle beschreibt die Anforderungen an eine Funktionalität in einer Gruppe von Systemen, ohne auf die Details der Implementierung der Funktionalität einzugehen. Dabei kann eine Rolle durch ein System eingenommen werden, sofern das System den Anforderungen der Rolle genügt.*

Definition II.3 (Systemgruppe): *Eine Systemgruppe ist eine Gruppe von Systemen, in welcher jedes System eine bestimmte Teilfunktionalität übernimmt, sodass die Summe aller*

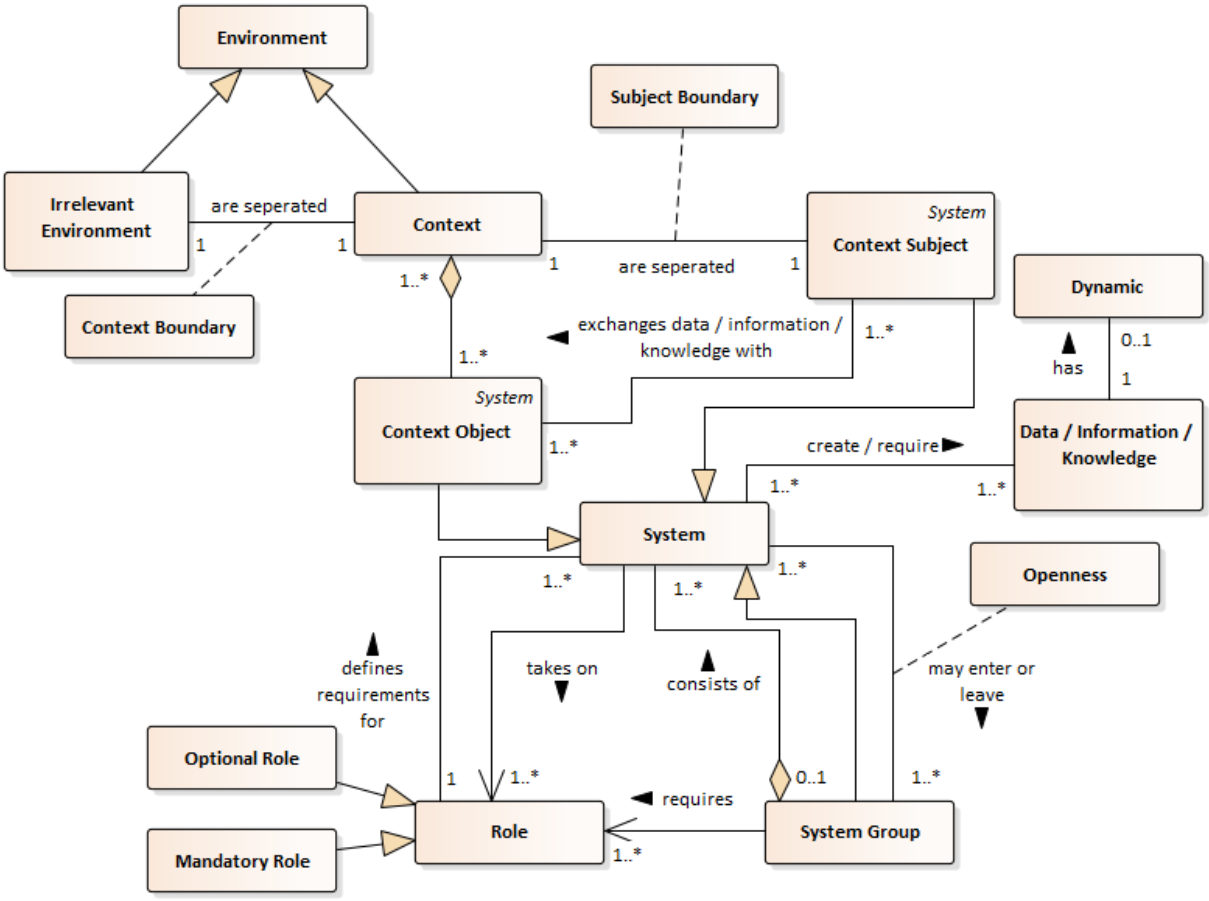

Abbildung II.6.: Metamodell zur System- und Kontextbeschreibung

Teilfunktionalitäten der von der Systemgruppe erwarteten Gesamtfunktionalität entspricht.

Demnach besteht eine Systemgruppe aus einer fest definierten Menge von Rollen, basierend auf der zu erzielenden Gesamtfunktionalität, welche auf die Rollen aufgeteilt wird. Da es sich bei der Definition von Rollen nur um Anforderungen an Systeme handelt und nicht um real existierende Systeme, ist es grundsätzlich möglich, dass diese Rollen zur Betriebszeit durch unterschiedliche Systeme eingenommen werden, sofern diese den Anforderungen der Rolle genügen. Dieser Zusammenhang wird in Anlehnung an [Tör+18*] als Offenheit der Systemgruppe bezeichnet und ist Definition II.4 dargestellt.

Definition II.4 (Offenheit): *Die Morphologie (i.e.S. die Topologie) der Systemgruppe ist nicht fest und kann sich im Laufe des Betriebs der beteiligten Systeme ändern, indem z.B. neue Systeme die Gruppe betreten und andere die Gruppe verlassen.*

Um im Rahmen der Anforderungsdefinition den Begriff der Offenheit zu charakterisieren, wird eine Kennzahl eingeführt, die die Häufigkeit der wechselnden Besetzung einer Rolle charakterisiert. Dies ist in Formel II.1 dargestellt, wobei β die Kennzahl für Offenheit beschreibt, indem die zu erwartende Zahl der Änderungen n durch eine für alle Rollen geltenden zeitlichen Bezugseinheit T (z.B. 365 Tage) geteilt wird.

$$\beta = \frac{n}{T} \tag{F.II.1}$$

Die Daten, Informationen bzw. das Wissen, welche in der Kontextbetrachtung zwischen Kontextsubjekt und Kontextobjekt ausgetauscht werden, werden grundsätzlich durch Systeme erzeugt bzw. konsumiert. Die Daten, Informationen und das Wissen sind dabei zeitlichen Änderungen unterworfen, welche als Dynamik bezeichnet wird. Dies ist in Definition II.5 dargestellt, welche in Anlehnung an [Gra18] erstellt wird.

Definition II.5 (Dynamik): *Die Dynamik eines Daten-, Informations- oder Wissensobjektes beschreibt dessen Änderung über die Zeit. Ist keine Änderung über die Zeit anzunehmen, wird von statischen Daten, Information bzw. Wissen gesprochen. Wird im Rahmen der Anforderungserhebung die Änderung über die Zeit als nicht relevant, aber existent eingestuft, wird von quasi-statischen Daten, Information oder Wissen gesprochen.*

Um im Rahmen der Anforderungserhebung an die Ontologie diesen Sachverhalt zu charakterisieren und zu bestimmen, in welcher Zeit α eine Anfrage durchzuführen ist, werden die Echtzeitkriterien aus der Domäne der Automatisierungstechnik herangezogen und auf SPARQL-Abfragen bezogen. Formel II.2 stellt die Kriterien dar, wobei dies anhand von Abbildung II.7 nachfolgend erläutert wird.

$$\begin{aligned} \alpha &\leq t_m \\ \alpha &\leq t_a \\ \alpha &\leq t_s \end{aligned} \tag{F.II.2}$$

Im Rahmen der Bestimmung der Echtzeit wird das zu automatisierende System hinsichtlich der kürzesten Meldezeit t_m, der kürzesten Antwortzeit t_a und der Systemtaktzeit t_s analysiert. Angenommen, es existiert ein Daten-, Informations- oder Wissensobjekt O, welches lediglich die Werte 0 oder 1 annehmen kann, so bezieht sich die kürzeste Meldezeit auf den kürzesten Zeitraum in dem 0 oder 1 gemeldet wird. Wobei angenommen wird, dass diese Meldung von Relevanz für das System ist. Die kürzeste Antwortzeit hingegen bezieht sich auf die kürzeste, erlaubte Zeit zwischen einer Abfrage und der Rückgabe mit Hilfe einer SPARQL-Abfrage. Die Systemtaktzeit hingegen bezieht sich auf die kürzeste Zeit zwischen zwei Abfragen auf das gleiche Objekt.

II-3.1.2. Funktionale Beschreibungen

Die im Rahmen dieser Arbeit genutzte funktionale Beschreibung basiert im Wesentlichen auf dem Metamodell nach [Lud+18] und dem Metamodell nach [Sch18]. Das daraus erstellte Metamodell ist in Abbildung II.8 zu finden und wird nachfolgend erläutert.

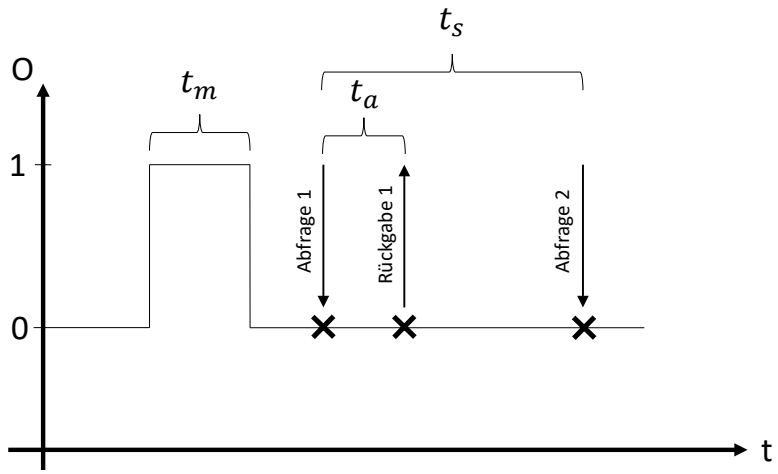

Abbildung II.7.: Beispielhafte Darstellung der Dynamik eines Daten-, Informations- oder Wissensobjektes

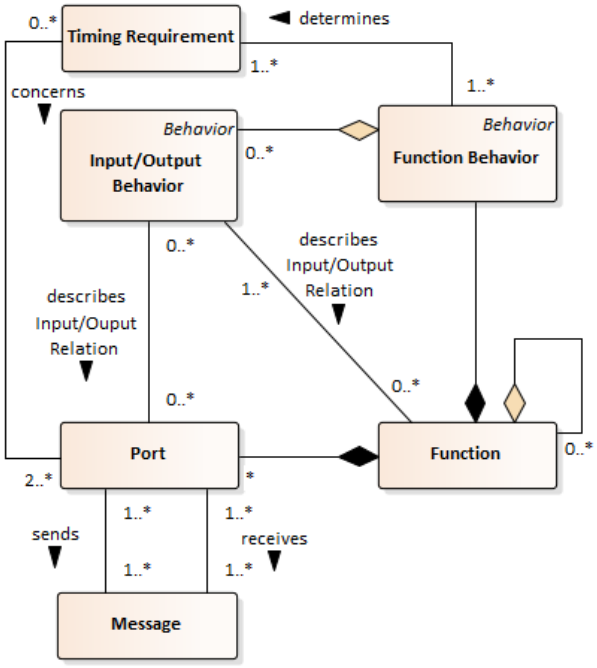

Abbildung II.8.: Metamodell zur Funktionsbeschreibung

Die Begriffe und Relationen des Metamodells wurden aus [Lud+18] übernommen, wobei auf eine Betrachtung von Abhängigkeiten (englisch: „Dependencies“) verzichtet wurde. Entsprechend [Lud+18] besitzt eine Funktion In- und Outputports, welche die Ein- und Ausgaben der Funktion spezifizieren, wobei zwischen den Ports unterschiedlicher Funktionen Nachrichten ausgetauscht werden. Dies stellt die in Definition II.6 beschriebene Funktionsstruktur dar.

Definition II.6 (Funktionsstruktur): *Die Funktionsstruktur bezeichnet eine statisch-*

strukturelle Darstellung der Funktionen eines Systems sowie eine Zuordnung der Ein- und Ausgaben anhand von ausgetauschten Nachrichten zwischen Ports unterschiedlicher Funktionen.

Die Verarbeitung der Ein- zu Ausgaben wird durch ein gewünschtes Verhalten, das Funktionsverhalten, beschrieben. Aus dem Funktionsverhalten lassen sich zeitliche Anforderungen (englisch: „Timing Requirement“) ableiten. Dies stellt das in Definition II.7 beschriebene Funktionsverhalten dar.

Definition II.7 (Funktionsverhalten): *Das Funktionsverhalten definiert die gewünschte Verarbeitung der Ein- und Ausgaben einer Funktion in einer ablauforientierten Sichtweise.*

II-3.1.3. Beschreibung ontologischer Anforderungen

Um eine Beschreibung ontologischer Anforderungen in die Beschreibung von System, Kontext und Funktion zu integrieren, wird eine Kombination der bisher vorgestellten Metamodelle vorgenommen, sowie eine Ergänzung um Konzepte zur Beschreibung ontologischer Anforderungen. Dies ist in Abbildung II.9 dargestellt und wird nachfolgend erläutert. Demnach beinhaltet eine Rolle eine Menge von Funktionen. Die Nachrichten, welche zwischen den Ports unterschiedlicher Funktionen ausgetauscht werden, enthalten dann die Daten, Informationen bzw. das Wissen, welches durch Systeme erzeugt bzw. konsumiert wird.

Für die Erhebung ontologischer Anforderungen ist nun der Nachrichteninhalt von besonderem Interesse, der Daten, Informationen bzw. Wissen enthält, welches durch die zu entwickelnde Ontologie beschrieben werden soll. Eine bestimmte Nachricht stellt damit einen Punkt ontologischen Interesses dar, welcher entsprechend [Hil+18*a] für die Herleitung ontologischer Anforderungen genutzt werden kann. Für diesen „Point of Interest“ können nun funktionale und nicht-funktionale Anforderungen entsprechend [SGV09] erhoben werden. Dieser Sachverhalt ist in Definition II.8 dargestellt.

Definition II.8 (Punkt ontologischen Interesses): *Ein Punkt ontologischen Interesses (POI) beschreibt einen Teil eines Artefaktes des Systemanforderungsmodells, aus welchem sich relevante Anforderungen an die zu entwickelnde Ontologie ableiten lassen. Die ontologischen Anforderungen werden dabei unterteilt in funktionale und nicht-funktionale ontologische Anforderungen.*

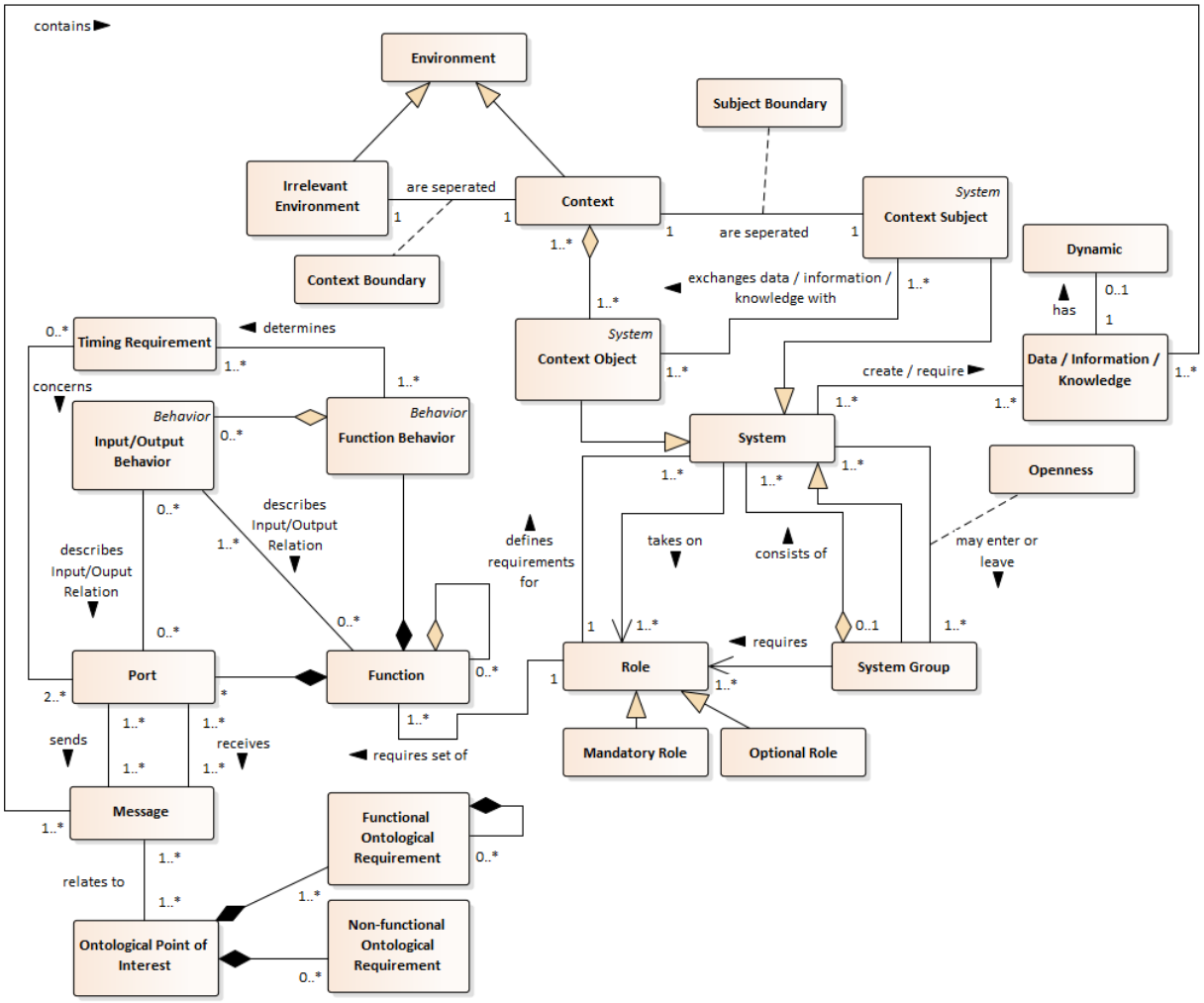

Abbildung II.9.: Metamodell zu System, Kontext, Funktion und ontologischen Anforderungen

Funktionale Anforderungen beziehen sich hierbei auf den Inhalt, den es im Rahmen der TBox- und ABox-Entwicklung zu berücksichtigen gilt. Die Anforderungen an diesen Inhalt werden in Form von Kompetenzfragen und Antworten formuliert, welche im späteren Verlauf des Entwicklungsprozesses durch eine SPARQL-Abfrage formalisiert und damit verifiziert werden können. Die Kompetenzfragen und Antworten werden so formuliert, dass erkennbar wird, welche Daten, Informationen bzw. Wissen in der Nachricht enthalten sein sollen. Dabei wird die Struktur der Fragen und Antworten nicht eingeschränkt, um Domänenexperten und Software-Entwickler nicht in ihrem Kreativprozess zu behindern.

Die nicht-funktionalen Anforderungen hingegen beziehen sich auf generelle Aspekte, die unabhängig vom Inhalt der Ontologie zu berücksichtigen sind. Dies betrifft beispielsweise die zu nutzende Sprache (bspw. Englisch oder Deutsch) für die Modellierung oder Einschränkungen hinsichtlich der zur Verfügung stehenden Zeit für Abfragen (Dynamik der Informationsobjekte). Die Definition für ontologische Anforderungen ist in Definition II.9 dargestellt.

Definition II.9 (Funktionale und nicht-funktionale ontologische Anforderungen):

Ontologische Anforderungen werden durch funktionale und nicht-funktionale Anforderungen beschrieben. Funktionale Anforderungen beziehen sich dabei auf den Inhalt der zu entwickelnden TBox und ABox und werden in Form von Kompetenzfragen und Antworten formuliert. Nicht-funktionale Anforderungen hingegen sind generelle Einschränkungen für die Modellierung, die nicht auf einen bestimmten Inhalt abzielen.

II-3.2. Profildiagramme

In diesem Abschnitt wird das im Rahmen der Anforderungserhebung zu nutzende Profil vorgestellt. Das Profil basiert dabei auf den Begriffen und Relationen des Metamodells, welches im vorherigen Abschnitt vorgestellt wurde. Dabei unterteilt sich das Profil in vier verschiedene Teilprofile, welche später in der Anforderungserhebungsmethodik auch einzeln eingesetzt werden. Abbildung II.10 stellt einen Überblick über die erstellten Profildiagramme und darin genutzten Begriffe mit Hilfe eines UML-Paketdiagramms vor. Die dargestellten Modelle wurden mit Hilfe des Werkzeugs Enterprise Architect [Spa18@] erstellt. Die nachfolgenden Abschnitte widmen sich der Erläuterung der einzelnen Profildiagramme. Aufgrund des hohen Platzbedarfs für die Abbildungen der einzelnen Profildiagramme in den folgenden Abschnitten sind die referenzierten Abbildungen im Anhang B zu finden.

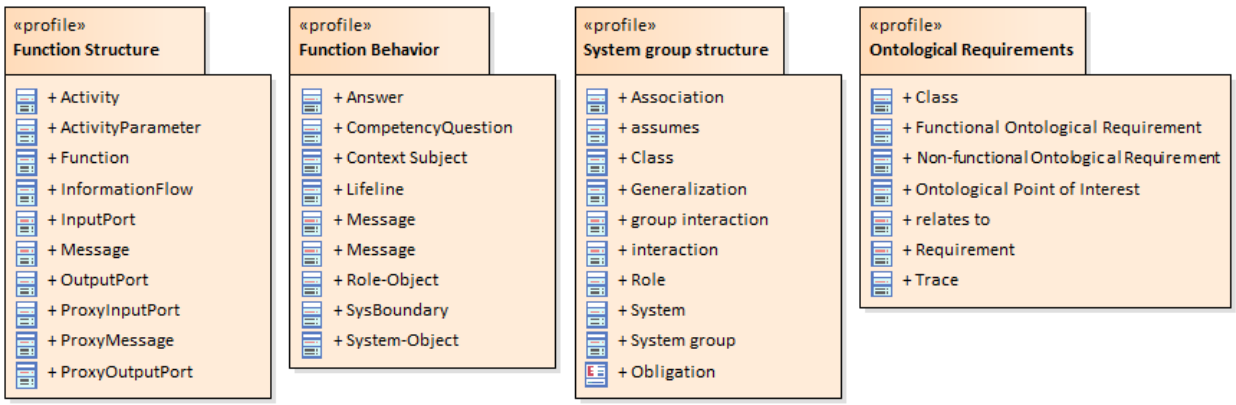

Abbildung II.10.: Überblick der erstellten Profildiagramme

II-3.2.1. Profil für ontologische Anforderungen

Für die Modellierung ontologischer Anforderungen wird im Rahmen dieser Arbeit auf den SysML-Anforderungsdiagrammen aufgebaut, wobei das entsprechende Profil in Abbildung B.1 zu finden ist. Dabei wird definiert, dass ein Punkt ontologischen Interesses eine Erweiterung der Metaklasse `Requirement` ist. Nicht-funktionale Anforderungen sind als Spezialisierungen dieser Punkte zu erachten. Funktionale Anforderungen werden hingegen als Spezialisierungen von UML-Klassen verstanden, da dies die Definition von UML-Operationen erlaubt. Funktionale ontologische Anforderungen können dann durch eine Kompetenzfrage und Antwort beschrieben werden, welche als UML-Operation modelliert werden. Hierbei kann dem Verhalten der Operation eine informelle Kompetenzfrage und Antwort zugewiesen wer-

den. Für nicht-funktionale ontologische Anforderungen gilt, dass eine textuelle Beschreibung der Anforderung zu erstellen ist. Zur Integration in andere Diagrammarten wird darüber hinaus der Verbinder `relates to` eingeführt, welcher eine Erweiterung der UML-Metaklasse `Trace` ist. Dabei kann ein `relates to` lediglich einen Punkt ontologischen Interesses, bzw. dessen Spezialisierungen, als Ursprung haben.

II-3.2.2. Funktionsstrukturprofil

Das im Rahmen dieser Arbeit genutzte Profil für die Modellierung von Funktionsstrukturen basiert auf der Nutzung von UML-Aktivitätsdiagrammen und ist in Abbildung B.2 dargestellt. Dabei werden Funktionen als Erweiterung von Aktivitäten betrachtet. Die für die In- und Outputbeschreibung relevanten Ports werden als Erweiterung der Aktivitätsparameter modelliert. Zur Darstellung des Nachrichtenaustausches zwischen Ports in Form von Nachrichten wird die Metaklasse `InformationFlow` der UML erweitert, welche die Typisierung von Nachrichten mit Hilfe von UML-Klassen erlaubt. Um eine Unterscheidung der Semantik von Nachrichten zwischen Funktionen und innerhalb von Funktionen zu erreichen, wird ein Stellvertreter `Proxy` eingeführt. Eine Nachricht kann demnach nur zwischen Out- und Inputports angelegt werden. Sollte eine Funktion mit dem in UML-Aktivitätsdiagrammen üblichen Kompositionsmechanismus zergliedert werden, müssen Proxy-Ports und Proxy Nachrichten genutzt werden, welche eine weniger restriktive Semantik besitzen. Der in Abbildung B.1 eingeführte Verbinder `relates to` wird im vorliegenden Profil wiederverwendet und darf eine Funktion als Ziel aufweisen. Darüber hinaus kann Ports ein entsprechendes `Timing Requirement` in Form eines Tags hinzugefügt werden. Dieses entspricht dem schärfsten Kriterium aus Formel II.2 und wird bei der Betrachtung des Funktionsverhaltens festgelegt und in der Funktionsstruktur dokumentiert.

II-3.2.3. Funktionsverhaltensprofil

Für die Darstellung des Funktionsverhaltens im Rahmen dieser Arbeit wird ein Profil basierend auf UML-Sequenzdiagrammen genutzt, welches in Abbildung B.3 zu finden ist. Die Metaklasse `Lifeline` wird hierbei erweitert um die Stereotypen `Role-Object` und `System-Object`, welche jeweils entsprechend die Instanzebene einer Rolle bzw. Systems darstellen. Darüber hinaus werden zwei Spezialisierungen der UML-Metaklasse `Message` eingeführt. Diese werden genutzt, um Kompetenzfragen und Antworten im Sequenzdiagramm zu modellieren, wobei eine Kompetenzfrage stets eine Rolle oder ein System als Ursprung hat, das in seinem Programmablauf eine Anfrage an eine Ontologie stellen soll. Das Ziel einer Kompetenzfrage ist dann demnach eine funktionale ontologische Anforderung aus dem in Abbildung B.1 dargestellten Profil. Eine Antwort auf eine Kompetenzfrage ist dann entsprechend als Rückgabe dieser Anfrage zu verstehen. Des Weiteren wird der Stereotyp `Kontextsubjekt` als Erweiterung der Systemgrenze eingeführt. Da es sich bei funktionalen ontologischen Anforderungen um eine Spezialisierung einer UML-Klasse handelt, können deren UML-Operationen genutzt werden, um unterschiedliche Anfragen an die Ontologie sowohl informal als Kompetenzfragen und Antworten als auch im späteren Verlauf formal als SPARQL-Abfrage zu modellieren.

II-3.2.4. Systemgruppenstrukturprofil

Das im Rahmen dieser Arbeit genutzt Profil zur Modellierung der Systemgruppenstruktur basiert auf einer Erweiterung des UML-Klassendiagramms und ist in Abbildung B.4 dargestellt. Hierbei sind die `System group`, das `System` und die `Role` jeweils Erweiterungen der UML-Metaklasse `Class`. Die `assumes` Verbindung wird als Erweiterung der Metaklasse `Generalization` modelliert, sodass durch die vorgeschriebene Verwendung alle Methoden und Attribute einer Rolle bei Übernahme der Rolle auf das entsprechende System übergehen und in diesem Sinne von dem System zu implementieren sind. Zwischen Rollen kann darüber hinaus eine `group interaction` angelegt werden, welche eine Erweiterung der UML-Metaklasse `Association` ist und die Interaktion zwischen zwei Rollen charakterisieren soll. Die Verbindung `interaction`, welche ebenfalls eine Erweiterung der UML-Metaklasse `Association` ist, wird verwendet, um alle anderen Interaktionen zwischen Systemen und Rollen, oder Systemen und Systemen zu modellieren. Des Weiteren wird die in Formel II.1 definierte Rollenbesetzung pro Zeiteinheit einer Rolle in Form eines Tags dokumentiert. Die Notwendigkeit einer Rolle in einer Systemgruppe wird darüber hinaus ebenfalls als Auswahl zwischen `Mandatory` und `Optional` der Rolle als Tag hinzugefügt.

II-4. Modellierungsmethodik

Der vorliegende Abschnitt befasst sich mit der Definition der zur Erreichung des Forschungsteilziels I notwendigen Methodik. Hierzu stellt Abschnitt II-4.1 zunächst einen methodischen Überblick vor, wobei einerseits eine ablauforientierte und andererseits eine Artefaktflussorientierte Beschreibung vorgenommen wird. Die Abschnitt II-4.2 bis II-4.4 widmen sich dann der detaillierten Beschreibung der Methodik.

II-4.1. Methodischer Überblick

Der grundlegende Ablauf der Anforderungserhebung ist angelehnt an die Vorgehensweise zur Anforderungserhebung der [VDI 4499-3], wonach zunächst Anwenderanforderungen, dann System-Anforderungen und dann technische Anforderungen erhoben werden. In Anlehnung an diese Unterteilung fasst Abbildung II.11 die vorliegende Methodik zusammen. Die erste Phase der Methodik widmet sich der Modellierung von Zielen der Ontologieentwicklung. Hierzu wird das Entwicklungsziel in Form einer User Story entsprechend [Wau+14] aufgenommen und um Ontologie-bezogene Aspekte ergänzt. Auf diese Weise werden „Business-Purpose“ der Ontologie und des Systems miteinander verbunden und der Ansatz nach [NMN09] entsprechend erweitert. Hierbei wird mit Hilfe der User Story eine effiziente und strukturierte Art der Beschreibung der Entwicklungsziele vorgenommen, wobei zusätzlich relevante Akteure und Aktionen mit Relevanz für die Systementwicklung identifiziert werden.

Nachdem mit Hilfe von User Storys die wesentlichen Aktionen der Akteure aufgenommen wurden, können die dafür notwendigen Systemfunktionalitäten entsprechend [SGV09] mit

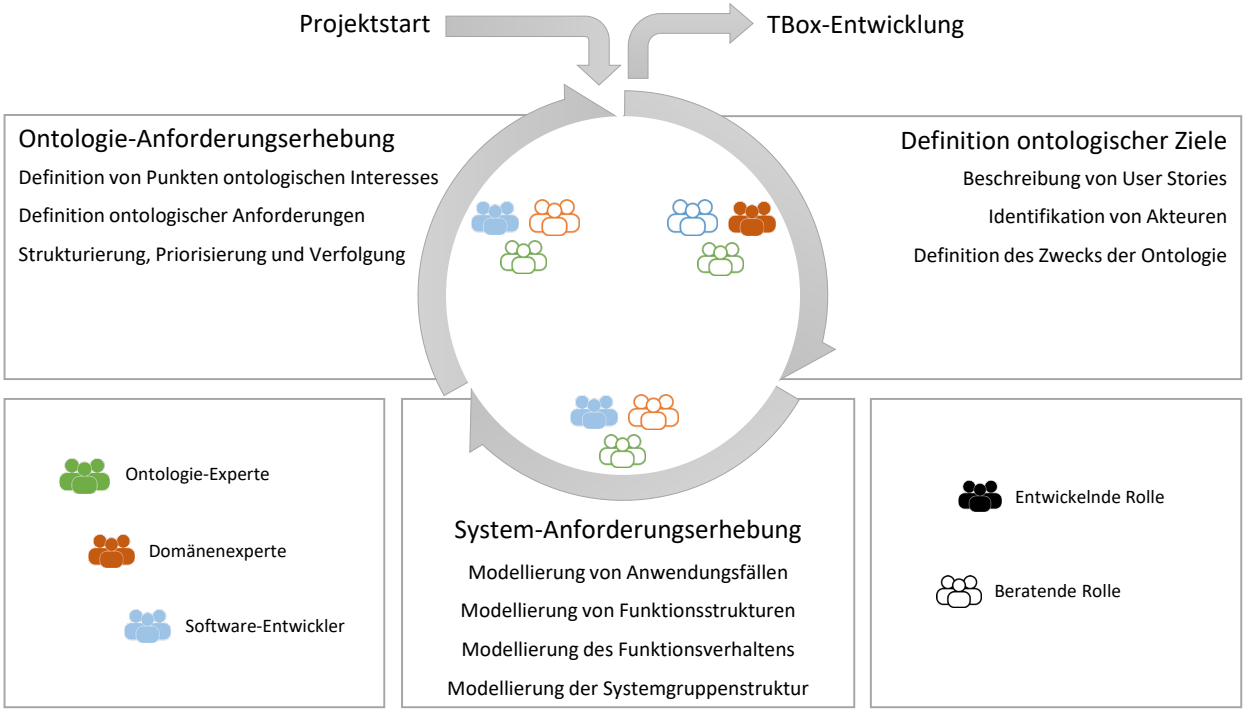

Abbildung II.11.: Iterativer Anforderungserhebungsprozess

Hilfe von UML-Anwendungsfalldiagramme modelliert werden. Ausgehend von den erstellten Anwendungsfalldiagrammen erfolgt dann eine detailliertere Betrachtung der im Anwendungsfall genannten Funktionalitäten. Hierzu wird das in Abschnitt II-3.2 eingeführte Profil herangezogen, um relevante Funktionsstrukturen und relevantes Funktionsverhalten zu modellieren. Das Profil bietet darüber hinaus einen vordefinierten Satz an Modellelementen für die Ontologie-Anforderungserhebung, womit eine Modellierung von ontologischen „Points of Interest“ (POIs) ermöglicht wird, welche für die Ontologie-Anforderungserhebung von Interesse sind. Letzterer widmet sich die dritte Phase der Methodik, wobei anhand der definierten POIs die Definition von Anforderungen an die Ontologie in Form von funktionalen und nicht-funktionalen Anforderungen erfolgen kann. An dieser Stelle sei darauf hingewiesen, dass das beschriebene Vorgehen iterativ durchgeführt werden muss. Dies ist in der Tatsache begründet, dass lediglich System-Anforderungen erhoben werden und keine Artefakte aus der eigentlichen Systementwicklung vorliegen müssen. Mit fortschreitender Entwicklung des die Ontologie nutzenden Systems kann eine Überarbeitung der System-Anforderungen und damit einhergehend der Ontologie-Anforderungen notwendig werden.

Betrachtet man den Artefaktfluss zwischen den einzelnen Methoden, kann Abbildung II.12 herangezogen werden. Demnach besteht die Methode aus drei wesentlichen Methodenbausteinen, welche bestimmte In- und Output-Artefakte besitzen. Die Methodik beginnt mit generellen Projektanforderungen, welche das in [SGV09] genannte „Set of ontological needs“ beinhalten. Letzteres kann beispielsweise durch Interviews und Befragungen von Domänenexperten und Software-Entwicklern in Erfahrung gebracht werden. Domänenexperten sind hierbei dazu in der Lage, die generellen Projektanforderungen zu interpretieren und zu erläutern, aber auch Priorisierungen durchzuführen und treten daher als entwickelnde Rolle auf. Wiederverwendet werden kann im ersten Methodenbaustein die in in Abschnitt II-4.2 vorgestellte

Vorlage für die Erfassung von User Storys. Resultat des ersten Methodenbausteins sind dann sowohl ontologische als auch systembezogene Entwicklungsziele. Der zweite Methodenbaustein befasst sich mit der Modellierung der System-Anforderungen, wobei die Modelle mit Hilfe des Profils aus Abschnitt II-3.2 erstellt werden und die Grundlage für den dritten Methodenbaustein liefern. Diese Aufgabe wird durch Software-Entwickler als entwickelnde Rolle übernommen, da sie die notwendige Fähigkeit besitzen, die systembezogenen Entwicklungsziele in UML-Modelle zu übersetzen und dabei auch entsprechende softwaretechnische Konzeptarbeit zu übernehmen. Der dritte Methodenbaustein widmet sich dann der eigentlichen Ontologie-Anforderungserhebung. Das Resultat dieses Methodenbausteins ist ein Ontologie-Anforderungsmodell, welches durch Software-Entwickler erarbeitet wird.

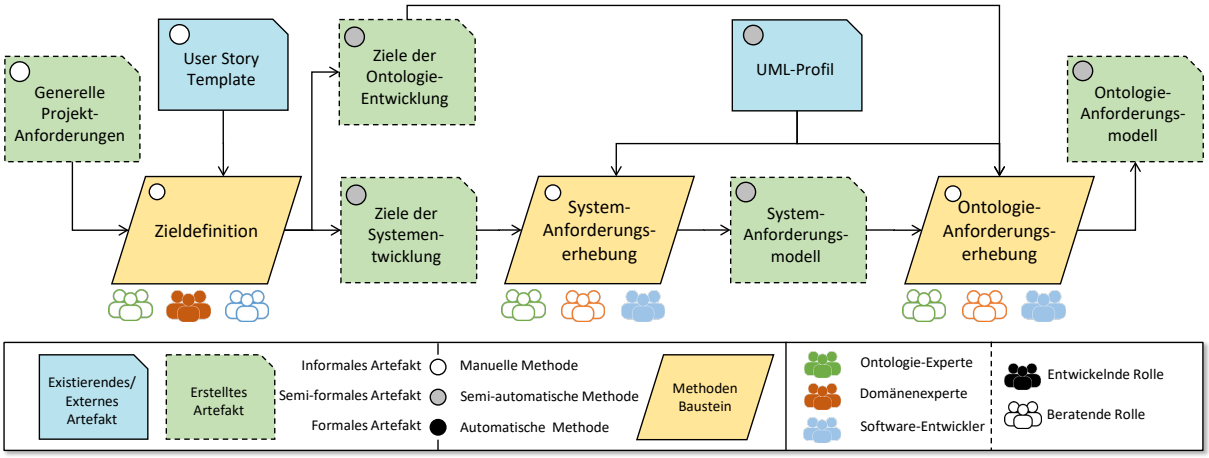

Abbildung II.12.: Artefaktflussorientierter Überblick zur Anforderungserhebung an Ontologien

II-4.2. Definition ontologischer Ziele

Im ersten Schritt dieser Phase werden im Rahmen eines Entwicklungsprojektes User Stories definiert. Dies erfolgt mit dem Ziel, sowohl einzelne Akteure (Stakeholder) und ihre individuellen Ziele zu identifizieren, als auch die Funktionalitäten der zu entwickelnden Systeme auf einer vergleichsweise hohen Abstraktionsebene. Des Weiteren kann mit Hilfe der User Story bereits der grundlegende Zweck der Ontologieentwicklung abgeleitet werden. Im Rahmen der gesamten Methodik dienen User Stories daher als Einstiegspunkt für Iterationen. Für die Erstellung der User Story wird die in Abbildung II.13 gezeigte Vorlage genutzt, deren Bestandteile (siehe farbliche Markierungen in Abbildung II.13) nachfolgend vorgestellt werden.

Neben einer Identifikationsnummer und einem Titel beinhaltet jede User Story eine Beschreibung, welche der Standardnotation von [Wau+14] folgt und Aufschluss auf die Fragen „Wer“, „Was“ und „Warum“ gibt. Die Frage „Wer“ wird dadurch beantwortet, indem ein entsprechender Akteur definiert wird, welcher entweder direkt mit der Ontologie oder mit einem die Ontologie nutzenden System interagiert. Das „Was“ hingegen wird durch die Nennung einer Aktion bzw. abstrakten Software-Funktion beantwortet, was auch den Ausgangspunkt für die nachfolgende Anwendungsfallmodellierung liefert. Das „Warum“ hingegen definiert das

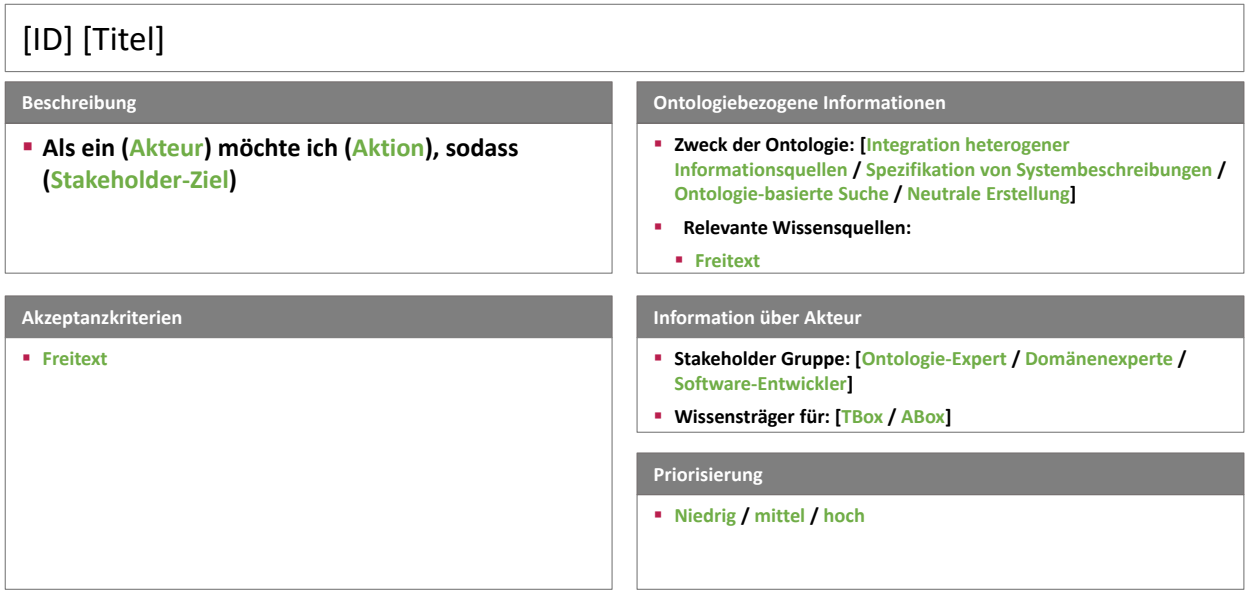

Abbildung II.13.: User Story Vorlage

Ziel, welches der Akteur mit der Aktion erreichen möchte. Die in der Vorlage zu definierenden Akzeptanzkriterien beschreiben aus Sicht des Akteurs, unter welchen Bedingungen dieses Ziel als erreicht gilt. Basierend auf dieser Beschreibung kann der Zweck der Ontologie nach [SGV09] abgeleitet werden, wobei dieser entsprechend der vom Akteur gewünschten Aktion kategorisiert wird, sowie bereits bekannte Wissensquellen für die spätere Ontologieentwicklung genannt werden. Neben der Möglichkeit zur Priorisierung der User Story sind darüber hinaus für jeden Akteur Informationen zu ergänzen, welche Stakeholder-Rolle er in der Ontologieentwicklung spielt und ob er als Wissensträger für TBox- bzw. ABox-Entwicklung gilt.

Die User Stories werden darüber hinaus genutzt, um etwaig disjunkte Systemfunktionalitäten zu identifizieren, sodass eine parallele Erhebung der Anforderungen an System bzw. Ontologie erfolgen kann oder eine entsprechende Priorisierung. Dies erfolgt mit Hilfe des von [Gre19%] entwickelten Klassifikationsschemas, welches in Tabelle II.2 zu finden ist. Daraus geht hervor, dass immer dann ein neuer Entwicklungspfad zu eröffnen ist, wenn sich die gewünschte Aktion des Akteurs signifikant von bisherigen Aktionen unterscheidet und somit nicht in die nachfolgenden Anforderungsmodelle integrierbar ist, da es sich im Folgenden um unterschiedliche Funktionalitäten handelt.

Tabelle II.2.: Ermittlung von Entwicklungspfaden anhand spezifizierter User Stories [Gre19%]

Akteur	Aktion	Ziel	Entwicklungspfad
identisch	Identisch	identisch	gemeinsam
identisch	Identisch	verschieden	gemeinsam
identisch	verschieden	Identisch	neu
identisch	verschieden	verschieden	neu
verschieden	Identisch	Identisch	gemeinsam
verschieden	identisch	verschieden	gemeinsam
verschieden	verschieden	Identisch	neu
verschieden	verschieden	verschieden	neu

II-4.3. Definition von Systemanforderungen

Der vorliegende Abschnitt stellt die Methodik zur System-Anforderungserhebung dar. Hierzu wird im Abschnitt II-4.3.1 zunächst die im Rahmen der Methodik intendierte Verwendung von Anwendungsfalldiagrammen vorgestellt, bevor die Abschnitte II-4.3.2 bis II-4.3.4 auf die Verwendung des in Abschnitt II-3.2 vorgestellten Profils eingehen.

II-4.3.1. Modellierung von Anwendungsfällen

Basierend auf den zuvor erstellten User Stories können die Funktionalitäten einer Software abgeleitet werden. Entsprechend [Wau+14] beschreibt eine User Story eine Funktionalität, die für einen Akteur von Wert bzw. Nutzen ist. Diese Funktionalität, welche in der `Aktion` der User Story enthalten ist, wird im vorliegenden Schritt mit einem Anwendungsfalldiagramm hierarchisch strukturiert. Ein Anwendungsfalldiagramm bildet generell Anforderungen an ein Systems ab, indem es die wesentlichen Funktionalitäten eines Systems darstellt, mit denen Akteure interagieren [Obj17@]. Das betrachtete System wird hierbei als Kontextsubjekt bezeichnet und mit einem Rahmen abgegrenzt. Die für die Realisierung der vorher definierten User Story relevanten Funktionalitäten werden mit Hilfe von Relationen zu Akteuren oder „beinhaltet“ (englisch `includes`) bzw. „erweitert“ (englisch: `extends`) Relationen zu anderen Funktionalitäten strukturiert. Die für diesen Schritt relevanten Modellelemente sind in Abbildung II.14 exemplarisch dargestellt. Es sei darauf hingewiesen, dass zu diesem Zeitpunkt noch keine Festlegung erfolgt, ob es sich um ein einzelnes System oder eine Systemgruppe handelt. Das Ergebnis dieses Vorgehens ist lediglich die Ausgangslage für die Modellierung von Funktionsstrukturen und insbesondere des Nachrichtenaustausches zwischen Funktionen, wobei im Falle der Anwendungsfallmodellierung lediglich die hierarchische Struktur der Funktionen des Kontextsubjekt modelliert werden, nicht jedoch der Informationsfluss zwischen Funktionen bzw. das erwünschte Verhalten der Funktionen.

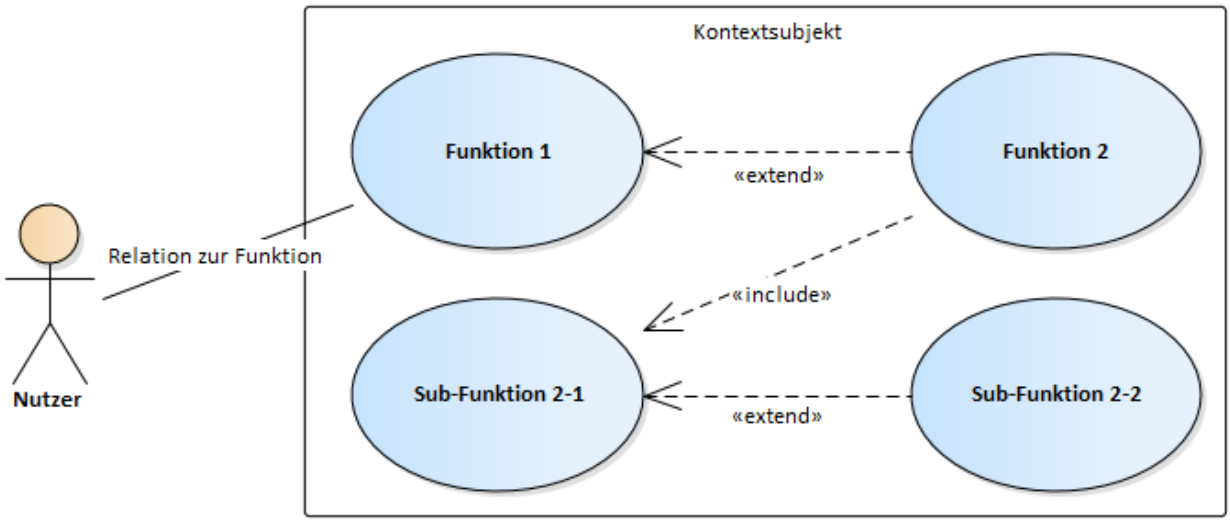

Abbildung II.14.: Relevante Modellelemente des Anwendungsfalldiagramms und deren Nutzung

II-4.3.2. Modellierung von Funktionsstrukturen

Basierend auf dem zuvor erstellten Anwendungsfalldiagramm wird in diesem Schritt ein Funktionsdiagramm mit Hilfe des in den vorangegangenen Abschnitten erstellten Profils auf Aktivitätsdiagrammbasis erstellt. Dabei wird das Ziel verfolgt, die im Anwendungsfall identifizierten Funktionen durch einen Austausch von Nachrichten miteinander in Beziehung zu setzen. Dies erfolgt zunächst auf rein struktureller Ebene durch die Verknüpfung der In- und Outputports der Funktionen, ohne eine Modellierung des gewünschten Verhaltens einer Funktion. Bei der Durchführung der Modellierung werden zunächst die im Anwendungsfall identifizierten Funktionen mit Hilfe des Funktions-Stereotyps dargestellt, siehe auch Abbildung II.15. Vordergründig sollten dabei die Funktionen auf der niedrigsten hierarchischen Ebene betrachtet werden. Es ist jedoch im Rahmen der Modellierung mit UML-Aktivitätsdiagrammen auch möglich, eine entsprechende Komposition bzw. Dekomposition durchzuführen. Da das Funktionsstrukturdiagramm diesen Mechanismus nicht einschränkt, ist auch die Nutzung von Funktionen auf höheren Hierarchieebenen des Anwendungsfalldiagramms möglich. Grundsätzlich sollte das Funktionsstrukturmodell alle Funktionen des Anwendungsfalldiagramms abbilden, kann jedoch auch mehr Funktionen abbilden, da es sich auf einer detaillierten Ebene mit der Modellierung der Funktionen befasst. Der Funktion werden nach Erstellung In- und Outputports hinzugefügt, wobei diese nach Möglichkeit zu typisieren sind. Zwischen den In- bzw. Outputports zweier Funktionen wird dann mit Hilfe des `Message`-Stereotypen eine Verbindung erstellt, welche Herkunft und Ziel einer Nachricht als Flussrelation darstellt, wobei die Verbindung mit einem entsprechenden Typen von Daten, Information oder Wissen (in Abbildungen abgekürzt mit `DIW`) zu versehen ist. Um das Funktionsverhalten einer Funktion zu modellieren, wird einer definierten Funktion ein Sub-Diagramm in Form eines Sequenzdiagramms zugewiesen. Abbildung II.15 stellt die relevanten Modellelemente exemplarisch in ihrem jeweiligen Zusammenhang dar, wobei auf das Beispiel in Abbildung II.14 zurückgegriffen wurde.

II-4.3.3. Modellierung des Funktionsverhaltens

Mit der bisherigen Funktionsmodellierung ist eine Hierarchisierung von Funktionen, ein Nachrichtenfluss zwischen In- und Outputports unterschiedlicher Funktionen sowie eine Beschreibung der In- und Outputports möglich. Insbesondere im Falle mehrerer In- und Outputs einer Funktion ergeben sich Fragestellungen hinsichtlich des gewünschten Verhaltens einer Funktion. Hierzu wird ein Funktionsverhaltensdiagramm, mit Hilfe des in den vorangegangenen Abschnitten erstellten Profils auf Sequenzdiagrammbasis, erstellt. Da es sich bei der Spezifikation des Funktionsverhaltens um Anforderungen handelt, ist diese Betrachtung als Szenariobetrachtung zu verstehen [PoRu11]. In einem Szenario wird beschrieben, wie das gewünschte Verhalten eines Systems in bestimmten Situationen sein soll. Während die Funktionsstruktur also die Funktionen in einer statisch-strukturellen Perspektive darstellt, können für eine Funktion nun diverse Funktionsverhaltensdiagramme erstellt werden, je nach der Anzahl möglicher Szenarien. Für Funktion 2 in Abbildung II.15 könnten beispielsweise zwei Szenarien aufgrund der Verzweigung des Nachrichtenflusses nach Sub-Funktion 2-1 angelegt werden. Ausgangspunkt für die Modellierung ist das zuvor erstellte Funktionsstrukturdiagramm und eine dort ausgewählte Funktion, deren Funktionsverhalten spezifiziert werden

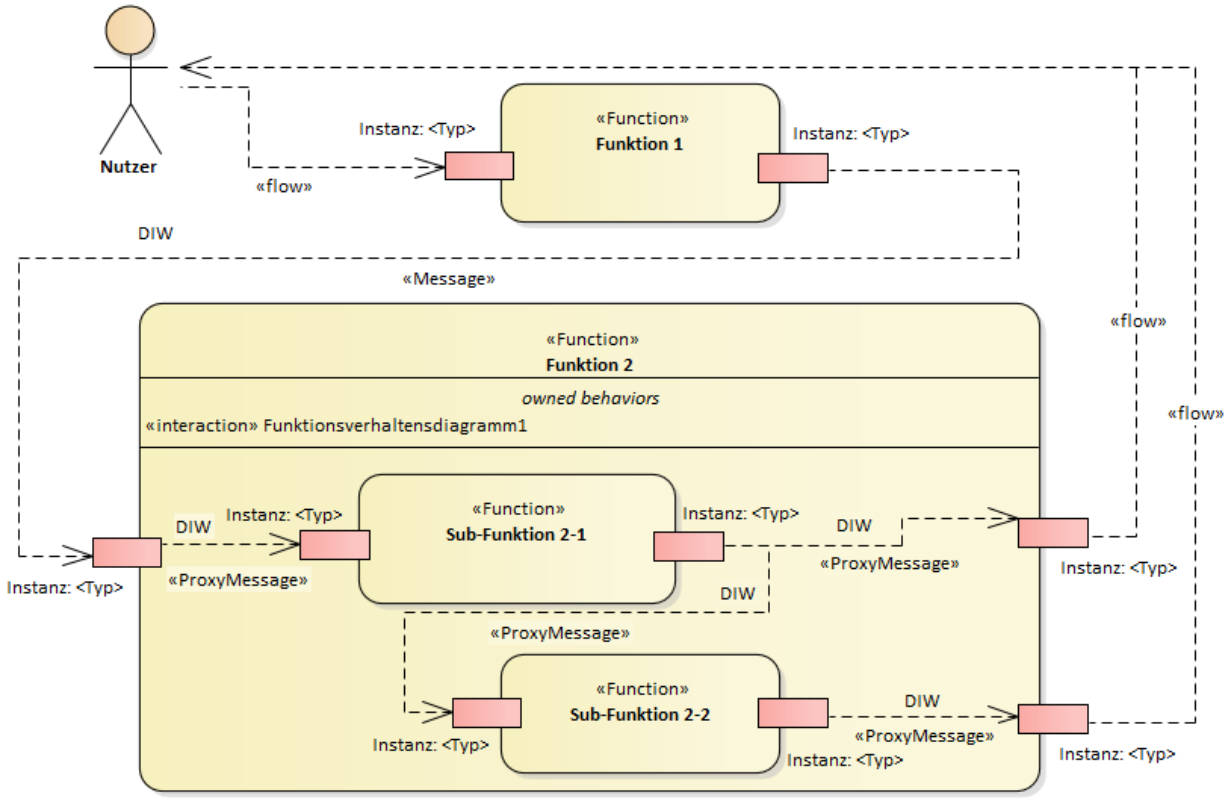

Abbildung II.15.: Relevante Modellelemente des Funktionsstrukturdiagramms und deren Nutzung

soll. Hierzu wird zunächst das Kontextsubjekt durch Festlegung der betrachteten Funktion definiert. Anschließend wird die gewünschte Menge an Rollenobjekten und Systemobjekten erstellt und über Nachrichten miteinander in Beziehung gesetzt, wobei die in den Nachrichten beinhalteten Methodenaufrufe berücksichtigt werden. Hierbei wird für existierende Systeme der `System-Object` Stereotyp genutzt und für noch nicht entwickelte Systeme der `Role-Object` Stereotyp. Nachdem eine entsprechende Menge an Szenarien modelliert wurde, um das Funktionsverhalten zu spezifizieren, kann eine Typisierung der Rollen und Systeme in der Systemgruppenstruktur erfolgen. Abbildung II.16 stellt die relevanten Modellelemente in ihrem jeweiligen Zusammenhang dar.

II-4.3.4. Modellierung der Systemgruppenstruktur

Um das in den verschiedenen Szenarienbetrachtungen modellierte, gewünschte Funktionsverhalten zusammenzufassen und Elementen in einer Systemstruktur zuzuordnen, wird eine Systemgruppenstruktur erstellt. Hierbei wird eine Systemgruppe erstellt, welche als eine Komposition von Typen von Rollen zu verstehen ist. Hierzu werden die Funktionsverhaltensdiagramme analysiert und die darin enthaltenen Objekte auf Typen aufgeteilt, wodurch sich eine Menge an existierenden Systemen und abstrakten Rollen ergibt. Der `group interaction`

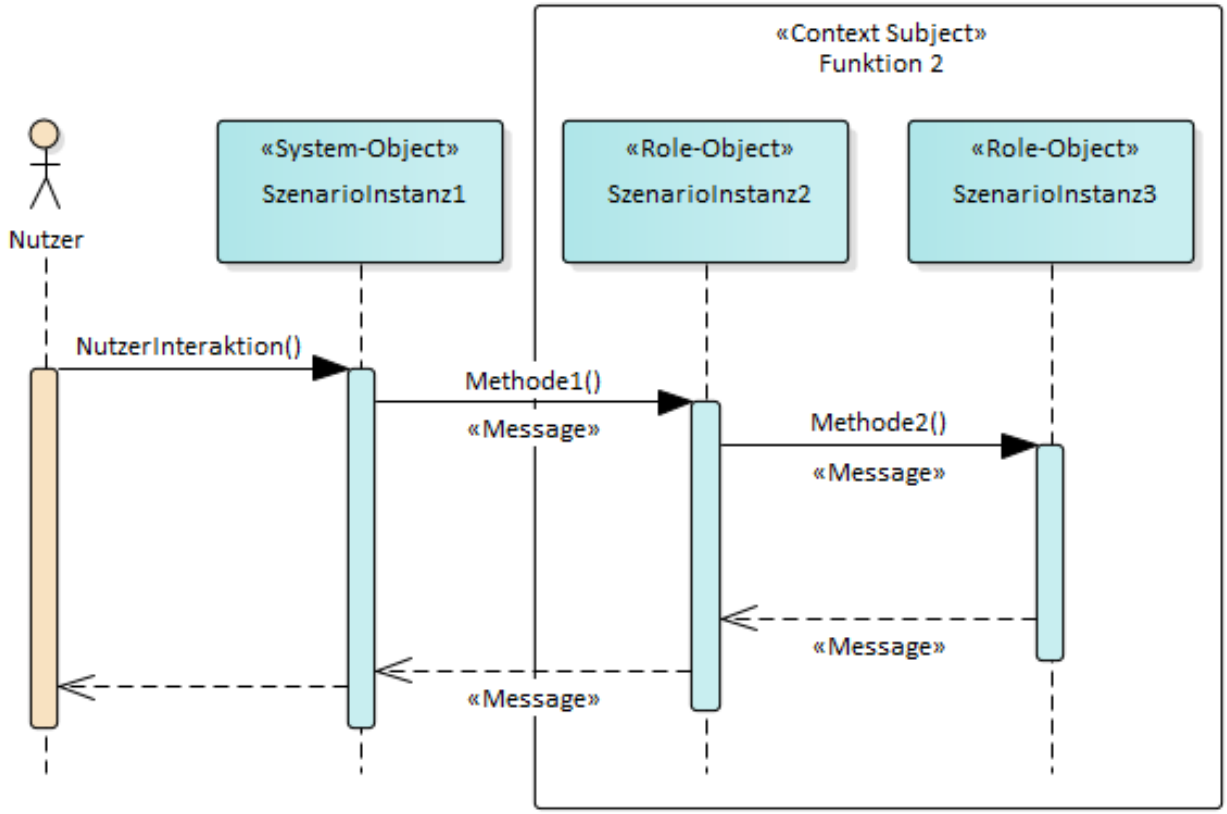

Abbildung II.16.: Relevante Modellelemente des Funktionsverhaltensdiagramms und deren Nutzung

Verbinder wird dann genutzt, um die Interaktion von Rollen innerhalb oder Rollen anderer Systemgruppen zu charakterisieren. Der `interaction` Verbinder hingegen wird genutzt, um Interaktionen zwischen allen anderen Entitäten (z.B. Systeme oder Nutzer) oder Interaktionen zwischen allen anderen Entitäten und den Rollen darzustellen. Über die Typisierung der Nachrichten der Funktionsverhaltensdiagramme erhalten die Rollen und Systeme die entsprechenden Methodenaufrufe aller ihrer Instanzen. Auf diese Weise fasst das Strukturdiagramm die notwendigen Methoden zusammen, die für die Implementierung einer entsprechenden Funktion (siehe Funktionsstrukturdiagramm) nötig sind. Sollte im weiteren Verlauf der Systementwicklung einer Rolle ein entsprechendes System zugeordnet werden, kann dies über den `assumes` Verbinder erfolgen, welcher dann zu einer Übernahme aller Methoden führt. Abbildung II.17 stellt die relevanten Modellelemente in ihrem jeweiligen Zusammenhang dar.

II-4.4. Definition ontologischer Anforderungen

Basierend auf den bisher erstellten Anforderungsmodellen können nun die ontologischen Anforderungen erhoben werden. In den beiden nachfolgenden Abschnitten wird hierzu zunächst dargestellt, wie die Anforderungen an die Ontologie erhoben werden, bevor dargestellt wird, wie diese strukturiert und verfolgt werden.

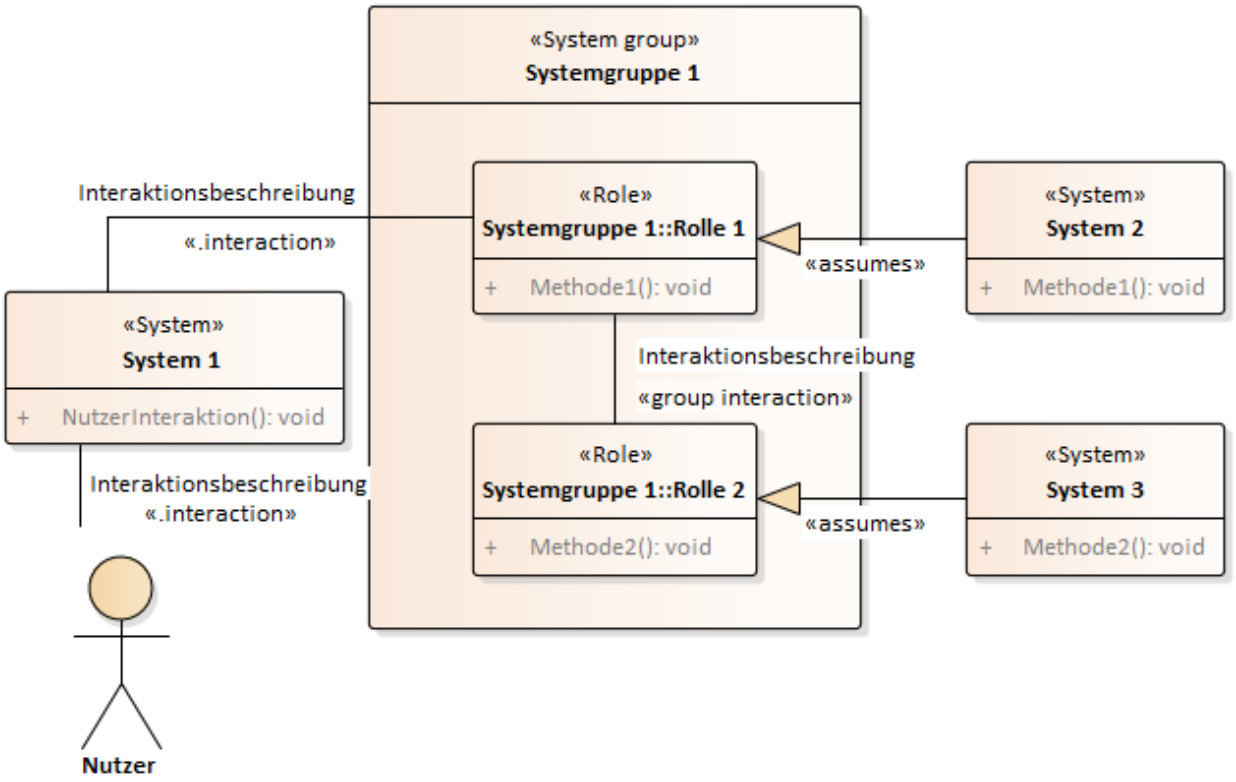

Abbildung II.17.: Relevante Modellelemente des Systemgruppenstrukturdiagramms und deren Nutzung

II-4.4.1. Erhebung funktionaler und nicht-funktionaler ontologischer Anforderungen

Für die Erhebung ontologischer Anforderungen ist es, wie in [Hil+18*a] beschrieben, zunächst notwendig, Punkte ontologischen Interesses zu identifizieren. Hierzu wird zunächst das Funktionsstrukturdiagramm dahingehend untersucht, dass Funktionen identifiziert werden, welche eine Ontologie für ihre Realisierung benötigen. Dem entsprechenden Funktionsstrukturdiagramm wird dann ein Punkt ontologischen Interesses hinzugefügt und mit der entsprechenden Funktion über die `relates to` Verbindung verknüpft. Dies ist in Abbildung II.18 dargestellt. Dabei sollte der Punkt ontologischen Interesses sich auf eine Art von Nachricht beziehen, welche die Funktion empfängt oder erzeugt. Ein Indiz für die Notwendigkeit einer Ontologie, und damit für einen POI, kann die Interaktion zwischen verschiedenen Rollen sein, welche als offen entsprechend Definition II.4 gelten. In diesem Falle kann die Ontologie dazu genutzt werden, die ausgetauschten Daten, Informationen bzw. das Wissen zu spezifizieren. Weitere Indizien für die Notwendigkeit einer Ontologie sind in Abschnitt I-2.2.2 zu finden. Die für eine Funktion relevanten nicht-funktionalen ontologischen Anforderungen können mit Hilfe des SysML-Stereotyps `Containment` an einen Punkt ontologischen Interesses angefügt werden. Nachdem Punkte ontologischen Interesses identifiziert und mit den entsprechenden Funktionen verbunden wurden, kann das entsprechende Funktionsverhalten untersucht werden.

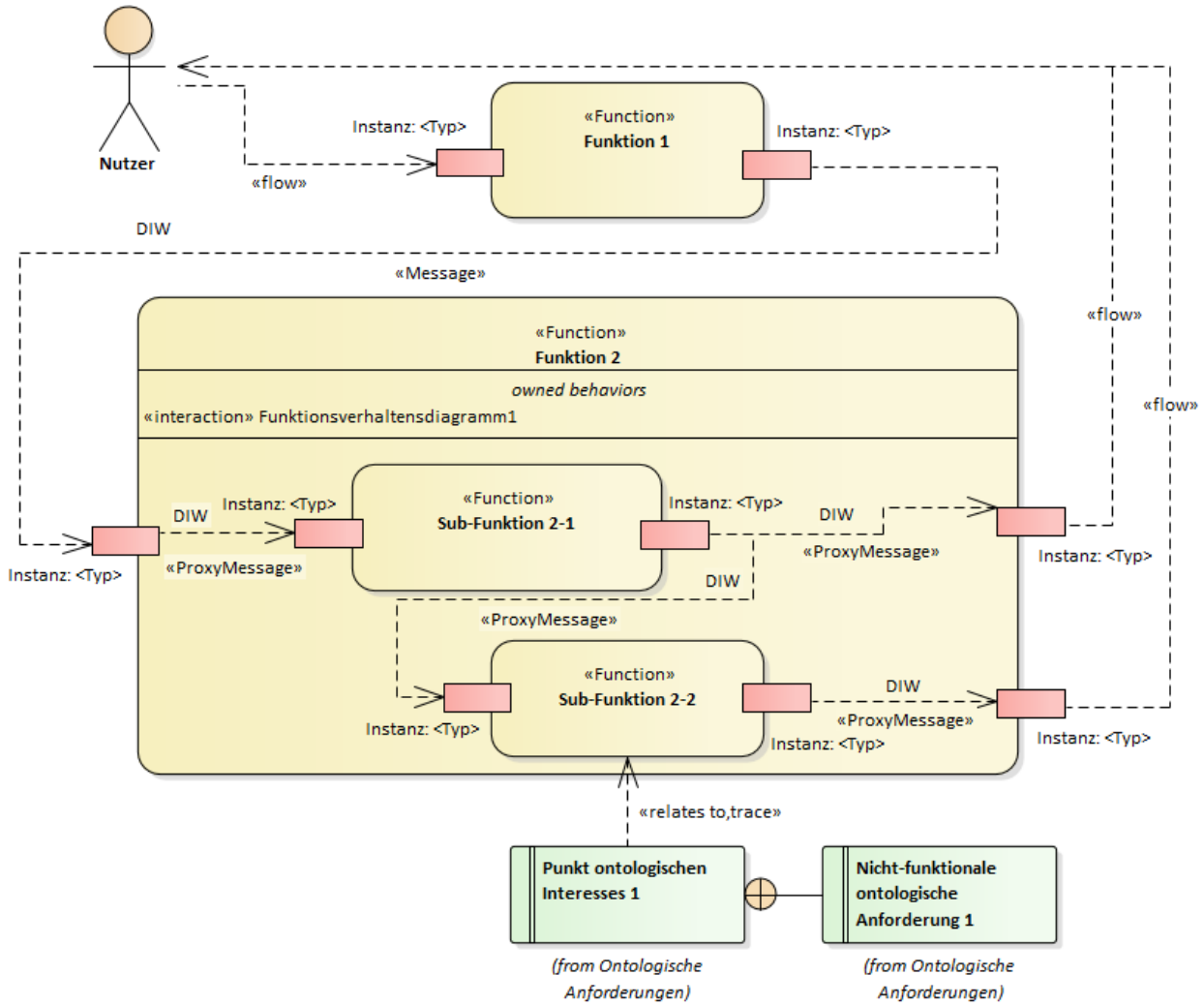

Abbildung II.18.: Ontologische Anforderungen im Funktionsstrukturdiagramm

Bei der Untersuchung des Funktionsverhaltens einer Funktion werden alle Funktionsverhaltensdiagramme hinsichtlich zweier Fälle untersucht. Im ersten Fall enthält eine Nachricht Daten, Informationen oder Wissen, für deren Interpretation eine Ontologie notwendig ist. Im zweiten Fall führt der Empfang einer Nachricht dazu, dass eine Anfrage nach Daten, Informationen oder Wissen an eine Ontologie durchgeführt werden muss. Um von beiden Fällen zu abstrahieren, wird zunächst eine funktionale ontologische Anforderung (FOA) hinzugefügt, welche über eine Lebenslinie verfügt, siehe auch Abbildung II.19.

Die Benennung sollte sich dabei an einer Wissenskategorie orientieren (z.B. Prozesswissen), da eine FOA im folgenden mehrere Kompetenzfragen beinhalten kann. Zum entsprechenden Zeitpunkt in der Lebenslinie eines Objektes kann ein System-Objekt oder ein Rollen-Objekt mit dem `CompetencyQuestion` Verbinder mit der FOA verbunden werden. Hierzu wird eine Anfrage in Form einer UML-Operation für die FOA definiert, die durch eine Kompetenzfrage spezifiziert wird. Der `Answer` Verbinder wird hingegen von der FOA zum System- oder einem Rollen-Objekt gezogen und dient lediglich der Visualisierung, da die Spezifikation der Ant-

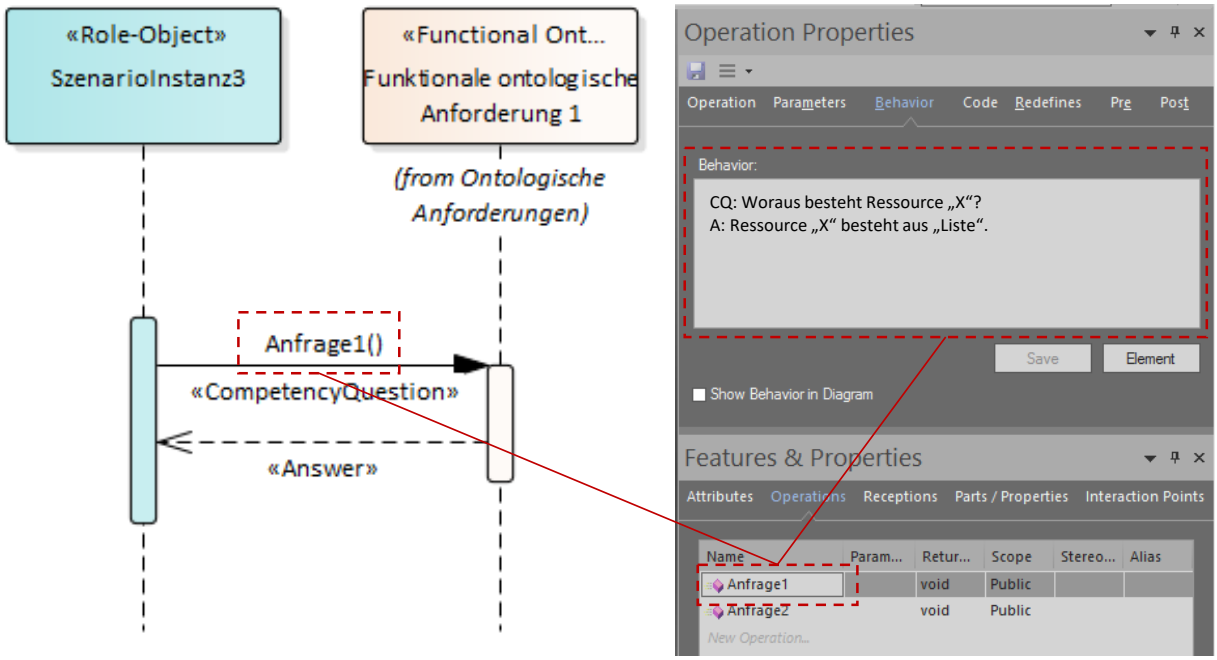

Abbildung II.19.: Ontologische Anforderungen im Funktionsverhaltensdiagramm

wort in der vorher definierten UML-Operation erfolgt. Zur Spezifikation der Kompetenzfrage kann die Operation, welche dem Nachrichten-Stereotyp `CompetencyQuestion` zugeordnet wird, weiter beschrieben werden. Hierzu ist es möglich, die Kompetenzfrage und Antwort in der Verhaltensspezifikation der Operation zu definieren. Dies ist auf der rechten Seite der Abbildung II.19 zu sehen. Auf diese Weise sammelt eine FOA diverse Anfragen einer Wissenskategorie in Form von UML-Operationen, die durch Kompetenzfragen und Antworten spezifiziert und im Rahmen der Ontologieentwicklung aufgegriffen werden. Darüber hinaus kann, zu einem späteren Entwicklungszeitpunkt, die Anfrage an die FOA durch formelle SPARQL-Abfragen ersetzt und in die Systementwicklung integriert werden.

II-4.4.2. Strukturierung, Priorisierung und Verfolgung ontologischer Anforderungen

Nachdem die funktionalen und nicht-funktionalen ontologischen Anforderungen erhoben wurden, können sie strukturiert, priorisiert und verfolgt werden. Hierzu bietet es sich zunächst an, ein SysML-Anforderungsdiagramm zu erstellen (siehe Abbildung II.20), in welchem alle bisher erstellten ontologischen Anforderungen strukturiert werden. Dies geschieht über die `Containment` Verbindung, analog zum Vorgehen in der Funktionsstrukturbeschreibung. Auf diese Weise wird ersichtlich, welche ontologischen Anforderungen für welche Funktionen von Relevanz sind.

Auf Basis dieses Diagramms ist auch eine Priorisierung der Anforderungen möglich. Dabei kann die Priorisierung der Anforderungen über mehrere Kriterien erfolgen:

- Bewertung der Relevanz für die Systementwicklung (Vermeidung von Wartezeiten)

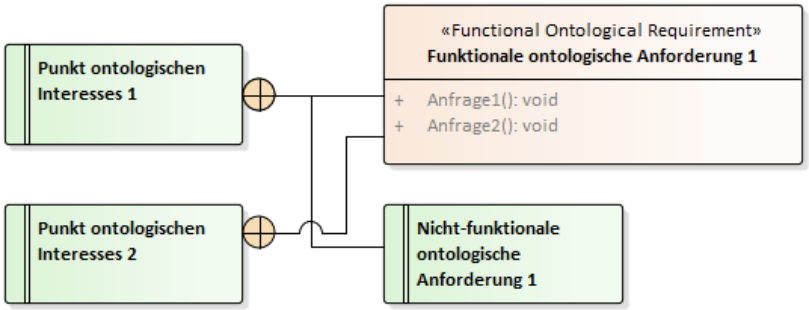

Abbildung II.20.: Strukturierung ontologischer Anforderungen im Anforderungsdiagramm

- Gesamtzahl der Operationen pro funktionaler, ontologischer Anforderung
- Gesamtzahl der Aufrufe aller Operationen pro funktionaler, ontologischer Anforderung

Die festgelegte Priorisierung definiert dann die Bearbeitungsreihenfolge in der Ontologieentwicklung. Zur Nachverfolgung der ontologischen Anforderungen während der Ontologieentwicklung ist es möglich, die im Anforderungsdiagramm enthaltenen ontologischen Anforderungen hinsichtlich ihrer Verwendung in anderen Diagrammen zu analysieren. Beispielsweise wurde das Modellement „Funktionale ontologische Anforderung I“ in Diagrammen der Abbildung II.19 und II.20 verwendet. Sofern es sich um dasselbe Modellelement handelt, ist eine Analyse der Verwendung dieses Modellelements durch Software-Werkzeuge wie Enterprise Architect [Spa18@] möglich. Dies ist exemplarisch in Abbildung II.21 dargestellt. Wie der Abbildung zu entnehmen ist, werden alle Beziehungen („Relationships“) des Modellelements angezeigt, so wie sie auch in den Abbildungen II.19 und II.20 auftauchen. Auf diese Weise ist eine Nachverfolgbarkeit von ontologischen Anforderungen in dem Sinne gegeben, als das bei Änderungen an Systemanforderungen erkannt werden kann, welche ontologischen Anforderungen hierdurch bedingt werden. Darüber hinaus kann auch in umgekehrter Richtung erkannt werden, für welche Systemanforderungen welche ontologischen Anforderungen von Relevanz sind. Durch die Verwendung der funktionalen ontologischen Anforderungen in den Funktionsverhaltensdiagrammen ist die Nachverfolgbarkeit auf Ebene der funktionalen ontologischen Anforderungen automatisch gegeben und bedarf keinerlei zusätzlicher Aufwände.

Relationships

Relationship	Stereotype	Source	Target	Target Type	Target Stereotype
Sequence	CompetencyQuestion	(SzenarioInstanz3)	(Funktionale ontologische Anforderung 1)	Class	Functional Ontological Requirement
Sequence	Answer	(Funktionale ontologische Anforderung...	(SzenarioInstanz3)	Sequence	Role-Object
Nesting		Funktionale ontologische Anforderung 1	Punkt ontologischen Interesses 2	Requirement	Ontological Point of Interest
Nesting		Funktionale ontologische Anforderung 1	Punkt ontologischen Interesses 1	Requirement	Ontological Point of Interest

Abbildung II.21.: Nachverfolgung ontologischer Anforderungen

III. Ontologie-Entwicklung TBox

Das vorliegende Kapitel widmet sich der Entwicklung einer Menge von Artefakten zur Erfüllung von Forschungsteilziel II. Hierzu stellt Abschnitt III-1 zunächst die relevanten Grundlagen zu Semantic Web Technologien vor, bevor Abschnitt III-2 den Stand der Wissenschaft darstellt. Im Anschluss widmet sich der Abschnitt III-3 der Darstellung der Methodik, welche zur Erreichung von Forschungsteilziel II entwickelt wurde. Des Weiteren wird Abschnitt III-4 eine Menge ontologischer Muster darstellen, welche als relevant für die betrachtete Domäne der Automatisierungstechnik angesehen wurden und den Einstieg in die Nutzung der Methode aus Abschnitt III-3 erleichtern.

III-1. Grundlagen zu Semantic Web Technologien

Im Rahmen der vorliegenden Arbeit sind einige Technologien im Kontext von Heavyweight-Ontologien von besonderer Bedeutung und sollen in den nachfolgenden Abschnitten beschrieben werden. Eine Übersicht kann mit Hilfe des Semantic Web Stack in Abbildung III.1 gegeben werden. In der Abbildung ist ersichtlich, welche Technologien auf anderen Technologien aufbauen, im Sinne einer Hierarchie. Die Technologien, welche für die vorliegende Arbeit von besonderer Bedeutung sind, sind das „Resource Description Framework“ (RDF), das „Resource Description Framework Schema“ (RDFS), sowie die „Web Ontology Language“ (OWL). Darüber hinaus kommt die RDF-basierte Abfragesprache „SPARQL Protocol and RDF Query Language“ (SPARQL) hinzu.

III-1.1. RDF - Resource Description Framework

Das „Resource Description Framework“ (RDF) entsprechend [CWL14@] ist ein Rahmenwerk für die Repräsentation von Informationen im Web. Der Standard definiert eine abstrakte Syntax im Sinne eines Datenmodells, welches dazu dient, alle RDF-basierten Sprachen und Spezifikationen zu verbinden. Die abstrakte Syntax besitzt zwei wesentliche Strukturen. Zunächst besteht ein RDF-Graph stets aus einer Menge von Subjekt-Prädikat-Objekt Aussagen (Tripel), in denen jedes Element entweder ein Identifier, leerer Knoten oder datentypisiertes Literal ist. Alle drei werden in der Menge der Subjekt-Prädikat-Objekt-Aussagen dazu genutzt, Ressourcen (z.B. Webseiten, Dokumente, Objekte) im Web zu beschreiben. RDF-Datenmengen werden dazu genutzt, Sammlungen von RDF-Graphen zu organisieren, einen Default-Graphen und keinen oder viele benannte Graphen (englisch: „named graphs“)

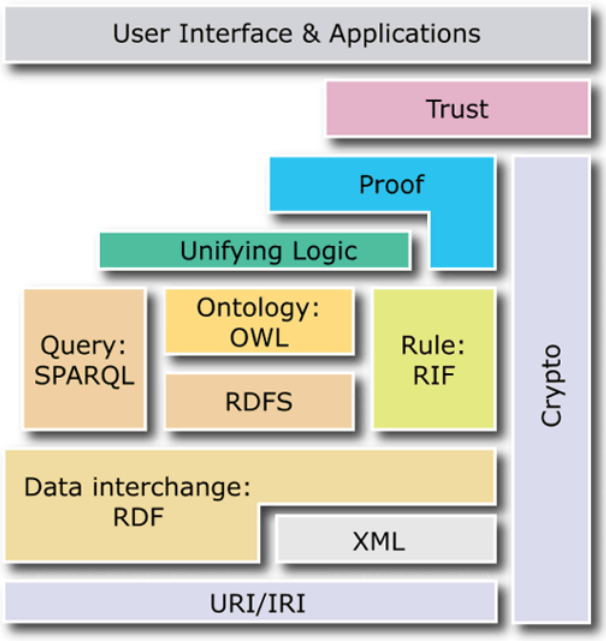

Abbildung III.1.: Semantic Web Layer Stack [CuBl15]

zusammenzufassen. Zusätzlich definiert der Standard eine Menge an nutzbaren Modellelementen [CWL14@].

Als Identifier werden „Internationalized Resource Identifier“ (IRI) genutzt, welche eine Erweiterung der „Uniform Resource Identifier“ (URI) darstellen, die für die eindeutige Identifikation von Ressourcen genutzt werden. Während URIs lediglich eine Teilmenge der ASCII Zeichen enthalten dürfen, können IRIs Zeichen aus dem „Universal Coded Character Set“ (UCS) enthalten. [Sak+18]

Für die Serialisierung von RDF-basierten Datenmodellen kommen diverse Datenformate in Frage (XML, JSON, etc.) [Hit+08]. Im Rahmen dieser Arbeit wird die entsprechend [W3C11@] standardisierte Art der Serialisierung genutzt: Turtle. Turtle ist eine einfache textuelle Serialisierung der Subjekt-Prädikat-Objekt-Aussagen, der Tripel. Ein Beispiel für die bisherigen Erläuterungen ist in Abbildung III.2 zu finden. Entsprechend [CWL14@] kann ein Subjekt entweder IRI oder leerer Knoten sein. Ein Prädikat kann stets nur IRI sein, wohingegen ein Objekt entweder IRI oder leerer Knoten oder Literal sein kann. Darüber hinaus wird ein default-Graph definiert, sowie optional weitere Named-Graphs. In diesem fiktiven Beispiel ist erkennbar, dass der erste Tripel, welcher im Default-Graph liegt, eine Verbindung zwischen dem Prozess und dem Produkt beschreibt. Zusätzlich sind zwei Named-Graphs definiert, welche den Prozess bzw. das Produkt beschreiben.

Um eine Menge von Modellelementen, z.B. die Modellelemente des Beispiels in Abbildung III.2, zu definieren, wird eine Unterteilung des IRI in einen Namensraum und einen Bezeichner vorgenommen [CWL14@]. Hierbei spricht man von der Fragmentierung der Identifier, wobei Namensraum und Bezeichner meist durch eine Raute getrennt werden. Dies erlaubt auch eine verkürzte Schreibweise, indem eine Abkürzung (Prefix) für den Namensraum definiert wird und jedem entsprechenden Bezeichner vorangestellt wird. Dies ist in Listing III.1

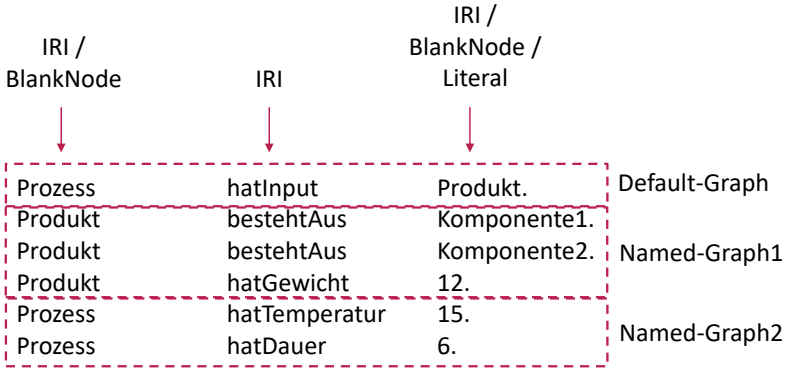

Abbildung III.2.: Beispielhafte Darstellung einer RDF-Serialisierung in Turtle

für den ersten Tripel aus Abbildung III.2 dargestellt, wobei oben die volle Schreibweise und unten die abgekürzte Schreibweise dargestellt ist. Zur eindeutigen Identifikation der Modellelemente eines Standards der W3C erhält i.d.R. jeder Standard seinen eigenen Namensraum (z.B. für RDF: „http://www.w3.org/1999/02/22-rdf-syntax-ns#“). Dass der Namensraum gleichzeitig auch einem „Uniform Resource Locator“ (URL) entspricht, der von der W3C reserviert und verwaltet wird, hat den Vorteil, dass hinter dieser URL weitere maschinenlesbare Informationen über den Standard hinterlegt werden können.

Listing III.1: Fragmentierung von IRIs in Namensraum und Bezeichner

```
%Vollstaendige Schreibweise:
<exProzesse#Prozess> <exProzesse#hatInput> <exProdukte#Produkt>.
----------------------
%Verkuerzte Schreibweise:
PREFIX P: <exProzesse#>
PREFIX I: <exProdukte#>

P:Prozess P:hatInput I:Produkt.
```

III-1.2. RDFS - Resource Description Framework Schema

Das „Resource Description Framework Schema“ (RDFS) entsprechend [BrGu14@] stellt ein Vokabular zur semantischen Anreicherung von RDF dar und ist in dem Sinne eine semantische Erweiterung von RDF. Das RDFS stellt Mechanismen zur Beschreibung von Gruppen von Ressourcen und Relationen bereit, indem bestimmte Arten von Ressourcen in RDFS definiert und beschrieben werden, sodass deren Semantik eindeutig definiert wird. Zu diesen Ressourcen gehören Klassen und Attribute (englisch: „properties“). Das Klassensystem in RDFS ist ähnlich zum Typensystem der objektorientierten Programmierung, besitzt jedoch einen wesentlichen Unterschied. Anstatt Klassen anhand der Properties zu definieren, welche deren Instanzen besitzen können, beschreibt RDFS die Properties im Sinne von Klassen von Ressourcen, auf die sie anwendbar sind. Dies schafft im Gegensatz zur Objektorientierung

den Vorteil, dass Attribute global ergänzt und modifiziert werden können, ohne die Klassen modifzieren zu müssen. [BrGu14@]

Durch die Spezifikation von Properties lässt sich beispielsweise festlegen, dass Subjekt und Objekt eines Prädikats stets von einem bestimmten Typen sein müssen. Darüber hinaus kann mit Hilfe der Definition von Klassen eine Gruppierung von Ressourcen vorgenommen werden, um anzuzeigen, dass eine Menge von Ressourcen alle vom selben Typen bzw. von derselben Klasse sind. Des Weiteren ist auch die Möglichkeit einer Hierarchisierung von Klassen und Properties gegeben. Hierbei ist die Logik so definiert, dass bspw. IRIs einer Sub-Klasse ebenfalls Mitglieder der Oberklassen sind. [Sak+18]

Durch die Einführung von Klassen und Properties ist in RDFS erstmals die Möglichkeit gegeben, zwischen TBox und ABox zu trennen, indem in der TBox das terminologische Wissen über Klassen und Properties dargestellt wird und in der ABox das Faktenwissen modelliert wird [Hit+08]. Ressourcen, die einen Fakt darstellen, der einer Klasse zugeordnet werden kann, werden auch als Individuen bezeichnet [HKR10]. Auf diese Weise werden einerseits Möglichkeiten des Schlussfolgerns basierend auf dem terminologischen Wissen geschaffen, andererseits aber auch die Möglichkeit zur Konsistenzprüfung, wobei Schlussfolgerungen auf die Zugehörigkeit zu Ressourcen im Sinne von Mengen begrenzt sind [Hit+08]. Listing III.2 zeigt ein fiktives Beispiel, in welchem in der TBox eine zweistufige Hierarchie von Fertigungsprozessen sowie die Definition eines Properties vorgenommen wird. Die ABox besteht nun zwar lediglich aus zwei Tripeln, jedoch mit Hilfe der TBox weitere Tripel durch Schlussfolgerung ergänzt werden. Da P:Bohren eine Sub-Klasse von `P:Trennverfahren` ist, kann geschlussfolgert werden, dass `P:Bohrprozess1` auch Teil der Klasse `P:Trennverfahren` ist. Da das Property `P:hasInput` als `rdfs:range` die Klasse `I:Produkt` aufweist, kann geschlussfolgert werden, dass `I:Platte` Teil der Klasse `I:Produkt` ist.

Listing III.2: RDFS Klassen und Properties am Beispiel

```
PREFIX P: <Prozesse#>
PREFIX I: <Produkte#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

%TBox
P:Trennverfahren rdf:type rdfs:Class.
P:Bohren rdf:type rdfs:Class.
P:Bohren rdfs:subClassOf P:Trennverfahren.
P:hasInput rdf:type rdf:Property.
P:hasInput rdfs:range I:Produkt.
P:hasInput rdfs:domain P:Trennverfahren.
I:Produkt rdf:type rdfs:Class.

%ABox
P:Bohrprozess1 rdf:type P:Bohren.
P:Bohrprozess1 P:hasInput I:Platte.

%ABox Tripel aus Schlussfolgerung
```

```
P:Bohrprozess1 rdf:type P:Trennverfahren.
I:Platte rdf:type I:Produkt.
```

III-1.3. OWL - Web Ontology Language

Die „Web Ontology Language“ (OWL) ist eine Ontologiesprache für das Semantic Web mit einer formalen (mathematischen) Bedeutung. OWL-Ontologien stellen im wesentlichen Klassen, Properties, Individuen und Datenwerte bereit und speichern diese in einem Dokument bzw. einer Datenbank. OWL-Ontologien können als Erweiterung zu RDFS und RDF im selben Dokument bzw. derselben Datenbank genutzt werden, um bestehendes Wissen oder bestehende Fakten weiter anzureichern. Die Syntax zur Darstellung von OWL-Ontologien, welche zum Austausch zwischen Werkzeugen genutzt wird, ist RDF. Demnach besteht auch eine OWL-Ontologie aus Subjekt-Prädikat-Objekt Aussagen, welche z.B. in RDF/XML oder Turtle serialisiert werden können. Während das Format RDF/XML die syntaktische Interoperabilität gewährleistet, stellt dieses Format keinerlei semantische Interpretation bereit. Dies erfolgt im Rahmen der Spezifikation durch eine mathematisch formale Definition der Modellelemente von OWL, z.B. Klassen und Properties. Diese Spezifikation nutzen Inferenzmaschinen und andere Werkzeuge um bspw. Klassenkonsistenz, Subsumption und Instanzabfragen zu ermöglichen. [W3C12@a]

Die RDF-basierte Semantik weist Modellelementen in RDF-Syntax eine formale Bedeutung zu, wobei diese Semantik vollständig kompatibel zur RDF-Spezifikation der W3C ist und somit Semantik in RDF-Dokumenten ergänzt. Die RDF-basierte Semantik definiert hierzu 78 Modellelemente und deren Semantik. Abbildung III.3 stellt eine Hierarchie der Bestandteile der RDF-basierten Semantik für eine Teilmenge der 78 Modellelemente dar. Ein gerichteter Pfeil zeigt auf die Oberklasse, ein Doppelpfeil zeigt semantische Äquivalenz an. [Sch+12@]

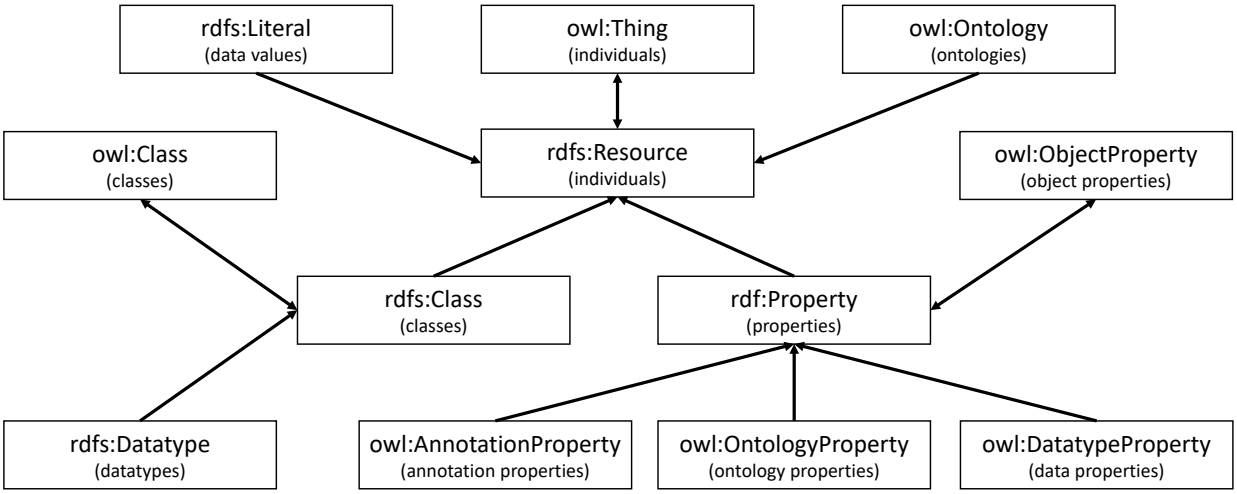

Abbildung III.3.: Hierarchie der Bestandteile der RDF-basierten OWL-Semantik [Sch+12@]

Neben der Einführung von Annotationen (`owl:AnnotationProperty`) werden Properties in OWL um `owl:ObjectProperties` und `owl:DatatypeProperties` erweitert, welche im Rahmen der Wissensmodellierung klassifiziert werden können. Die `owl:ObjectProperties` werden zwischen zwei Ressourcen angelegt und können mit Domain und Range des RDFS-Vokabulars näher beschrieben werden. Die `owl:DatatypeProperties` hingegen beschreiben Datenwerte, die einer Ressource zugewiesen werden können. Während die Domain eines `owl:DatatypeProperties` somit auf eine Klasse verweist, verweist die Range auf einen Datentypen, den ein Literal haben kann. Darüber hinaus können in OWL-Individuen der ABox als Spezialfall einer RDF-Ressource (`owl:NamedIndividual`) definiert werden. Listing III.3 stellt die Nutzung von `owl:ObjectProperties` und `owl:DatatypeProperties` an einem Beispiel dar.

Listing III.3: OWL-Individuen und Properties am Beispiel

```
PREFIX P: <Prozesse#>
PREFIX I: <Produkte#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

%TBox
P:Bohren rdf:type owl:Class.
P:hasInput rdf:type owl:ObjectProperty.
P:hasInput rdfs:range I:Produkt.
P:hasInput rdfs:domain P:Trennverfahren.
I:Produkt rdf:type rdfs:Class.
P:hasTemperature rdf:type owl:DatatypeProperty.
P:hasTemperature rdfs:domain P:Bohren.
P:hasTemperature rdfs:range xsd:float.

%ABox
P:Bohrprozess1 rdf:type owl:NamedIndividual.
P:Bohrprozess1 rdf:type P:Bohren.
P:Bohrprozess1 P:hasTemperature "55.123".
```

Hinsichtlich der Möglichkeiten zur automatisierten Schlussfolgerung mit einer Inferenzmaschine erweitert OWL die einfachen Möglichkeiten von RDFS - transitive Schlussfolgerung basierend auf Klassenhierarchien und die Möglichkeit der Schlussfolgerung von Klassenzugehörigkeit über Domain und Range eines Properties - um diverse weitere Möglichkeiten. Mögliche Schlussfolgerungen umfassen Subsumption, Klassenäquivalenz, Klassendisjunktheit, globale Konsistenz, Prüfung der Konsistenz einer Klasse, Zugehörigkeit von Instanzen zu Klassen, Bestimmung der Individuen einer Klasse. Für eine detaillierte Beschreibung der Inferenzarten sei auf [HKR10] verwiesen. Von besonderer Relevanz ist hierbei die Möglichkeit zur Prüfung der globalen Konsistenz. Diese erlaubt die Prüfung, ob eine Wissensbasis konsistent ist, bzw. frei von Widersprüchen, indem alle Tripel der TBox und ABox überprüft werden. Mögliche Widersprüche wären bspw. eine nicht erlaubte Klassenzugehörigkeit eines

Individuums, da zwei Klassen als verschieden (`owl:disjointWith`) definiert wurden. Ein weiteres Beispiel für einen derartigen Widerspruch wäre eine inkonsistente Verwendung der Properties. Wenn im Beispiel aus Listing III.3 dem Prädikat `P:hasTemperature` als Objekt das Individuum `P:Bohrprozess1` zugewiesen werden würde, wäre dies ein Widerspruch, da das Individuum `P:Bohrprozess1` nicht der Definition des Individuums `rdfs:range` des Prädikats entspricht. Auch kann die widersprüchliche Verwendung von komplementären Klassen oder Properties zu Inkonsistenzen führen. Definiert man bspw. die Klassen `P:Produkt` und `P:Trennverfahren` als Komplementäre (`owl:complementOf`) und würde ein Individuum beiden Klassen zuordnen, so wäre diese Beschreibung inkonsistent. [HKR10]

III-1.4. SPARQL - SPARQL Protocol And RDF Query Language

Die Abfragesprache „SPARQL Protocol And RDF Query Language“ (SPARQL) kann für die Abfrage von RDF-basierten Datenquellen genutzt werden. Dabei ist es irrelevant, ob die Daten nativ als RDF vorliegen oder mit Hilfe einer Middleware lediglich als RDF dargestellt werden (siehe z.B. [Gra18]). SPARQL bietet die Fähigkeiten für die Abfrage sogenannter Graphmuster (englisch: „graph pattern“) in RDF, wie auch das Testen von Werten und die Modellierung von Einschränkungen (englisch: „constraints“). Die Modellierung letzterer bezieht sich jedoch im Wesentlichen auf die Modellierung von Filtern und somit auf Einschränkungen der Ergebnisse einer SPARQL-Abfrage. Die Graphmuster liegen als Menge von Tripeln vor, in denen sich Variablen befinden, nach denen gefragt wird. Die Ergebnisse einer SPARQL-Abfrage können als Ergebnismenge oder als RDF-Graph dargestellt werden. Eine Abfrage besteht hierbei stets aus einer `SELECT` Klausel, in welcher die gesuchten Variablen deklariert werden und einer `WHERE` Klausel, in welcher das gesuchte Graphmuster definiert wird. [W3C08@]

Listing III.4 stellt ein Beispiel für eine SPARQL-Abfrage dar, welche sich auf das Beispiel in Listing III.3 bezieht. Für die deklarierte Variable `?x` würde entsprechend der Anfrage `P:Bohrprozess1` zurückgegeben werden, da dies der einzige Tripel ist, auf den das gesuchte Muster zutrifft.

Listing III.4: Beispiel einer SPARQL Abfrage

```
# Abfrage
SELECT ?x
WHERE {
    ?x rdf:type P:Bohren.
}

# Antwort als JSON
{
  "head": {
    "vars": [
      "x"
    ]
  },
```

```
  "results": {
    "bindings": [
      {
        "x": {
          "type": "uri",
          "value": "Prozesse#Bohrprozess1"
        }
      }
    ]
  }
}
```

Da es sich bei RDF um ein gerichtetes, beschriftetes Graphdatenformat handelt, handelt es sich bei SPARQL im Wesentlichen um eine Abfragesprache, welche die Übereinstimmung von gesuchten Graphen und existierenden Graphen (englisch:„graph matching“) überprüft. Da RDFS und OWL eine RDF-basierte Syntax und Semantik aufweisen, können SPARQL-Abfragen von einfachen Graphmustern hin zu sehr komplexen analytischen Abfragen reichen. Dies ist darin begründet, das eine SPARQL-Abfrage potentiell jedes Modellelement von RDF, RDFS und OWL im gesuchten Graphmuster nutzen kann. [Sak+18]

Neben der Abfrage von RDF-basierten Daten ist mit Hilfe von SPARQL auch eine Modifikation von RDF-basierten Daten möglich. Das Prinzip der Nutzung von Graphmustern bleibt dabei gleich, allerdings werden die definierten Graphmustern bei einem SPARQL-Update nicht abgefragt, sondern bspw. gelöscht (`DELETE`) oder dem Datensatz hinzugefügt (`INSERT`). [GPP13@]

III-2. Stand der Wissenschaft zur Ontologie-Entwicklung (TBox)

Im Bereich der Methoden zur Entwicklung von Ontologien sind entsprechend [SiLu14] drei Generationen von Entwicklungsmethoden erkennbar. Die erste Generation bezieht sich auf die Kernaktivitäten der Ontologieentwicklung (Begriffsbildung, Formalisierung, Integration und Implementierung [FGJ97]), wohingegen die zweite Generation vermehrt iterative Vorgehensweisen einführt, um der steigenden Komplexität der Ontologieentwicklung Rechnung zu tragen. Die dritte Generation hingegen verfolgt kollaborative Ansätze zur Ontologieentwicklung, die vermehrt auf die stärkere Involvierung von Domänenexperten setzen. Die in den nachfolgenden Abschnitten analysierten Ansätze sind insbesondere Repräsentanten der zweiten und dritten Generation, welche den in der vorliegenden Arbeit gestellten Anforderungen am ehesten entsprechen.

III-2.1. Methodik nach Presutti et al.

Das NeOn-Projekt sieht die Nutzung sogenannter Ontologie-Design-Muster (englisch: „Ontology Design Patterns“) vor, um die Wiederverwendung erstellter Ontologien zu verbessern. Ein Ontologie-Design-Muster (ODM) wird hierbei definiert als eine Lösung für ein wiederkehrendes Modellierungsproblem, wobei unterschiedliche Modellierungsprobleme durch unterschiedliche Arten von ODMs adressiert werden. Besonderer Fokus der Methodik sind sogenannte „Content Patterns“ (CP, nachfolgend einfach ODM). Jedes ODM wird separat modelliert, veröffentlicht und gepflegt, weshalb die Lösung für das wiederkehrende Modellierungsproblem separat von etwaigen Projektkontexten weiterentwickelt wird. Hinsichtlich der Methodik zur TBox-Entwicklung ist ein mehrstufiges Vorgehen zu erkennen, welches auch Domänenexperten berücksichtigt. Dabei orientiert sich das Vorgehen am agilen „eXtreme Programming“ Paradigma der Software-Entwicklung [Pre+09]. Im Rahmen des Vorgehens werden zunächst Anforderungen in Form von Tests erhoben, dann passende ODMs identifiziert, angepasst und veröffentlicht. Im Anschluss erfolgt eine Kombination der einzelnen ODMs und die Veröffentlichung des Gesamtresultats. Im Rahmen der Definition von Testfällen werden die in Form von Kompetenzfragen spezifizierten Fragen in SPARQL-SELECT Abfragen überführt. Diese Abfragen dienen dann als Testfall für die zu entwickelnde Ontologie. Demnach handelt es sich bei der Entwicklungsmethodik um einen Test-getriebenen Ansatz, der auf die Modularisierung der entwickelten TBox anhand von Testfällen setzt. [Pre+12]

III-2.2. Methodik nach Pinto et al.

Die DILIGENT-Methode ist eine Ontologie-Entwicklungsmethode mit einem Domänenexperten-zentrischem Ansatz, welcher in Abbildung III.4 dargestellt ist. Der Ansatz beruht auf fünf Schritten: der Erstellung (1), lokaler Anpassung (2), Analyse (3), Revision (4) und lokalen Updates (5) der Ontologie. Domänenexperten werden hier insbesondere in der ersten Phase, der Erstellung, mit eingebunden, indem sie ihr Wissen durch Expertenbefragungen und Interviews dem Ontologie-Experten derart darlegen, dass letzterer die Ontologie erstellen kann. Für die weiteren Phasen setzt der Ansatz auf die Anpassung der Ontologie durch die Software-Entwickler, welche die Ontologie an ihre Bedürfnisse anpassen. Dieser Prozess wird durch ein Kontrollgremium überwacht, welches die Änderungen mit den Domänenexperten abstimmt, sofern dies für sie von Relevanz ist. [PST04]

III-2.3. Methodik nach Peroni

Die von [Per17] vorgestellte Methodik ist ein Vertreter Test-getriebener Ansätze. Kern der Methodik stellt die Entwicklung sogenannter „Modelets“ dar, welche ähnlich der Methodik aus Abschnitt III-2.1 Module von Ontologien repräsentieren. Die Entwicklungsmethodik basiert auf drei Schritten. Zunächst werden basierend auf Kompetenzfragen und motivierenden Szenarien Testfälle erarbeitet. Diese Testfälle bestehen aus den motivierenden Szenarien,

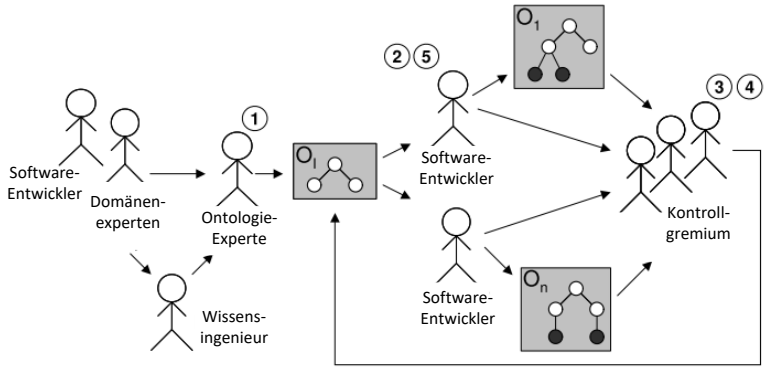

Abbildung III.4.: DILIGENT Methodik nach [PST04]

Kompetenzfragen, einem Glossar, der dazugehörigen TBox, einer exemplarischen ABox und einer Menge von SPARQL-Abfragen, welche auf den Kompetenzfragen beruhen. Für diese Testfälle werden Modelets entworfen. Im zweiten Schritt wird die Ontologie aus zwei oder mehr Modelets zusammengesetzt. Die vorher definierten Testfälle müssen dabei stets positive Ergebnisse liefern. Im dritten Schritt werden Verbesserungen an der TBox bzw. einzelnen Modelets vorgenommen, bevor eine neue Iteration beginnt. Im Rahmen der nächsten Iteration wird dann erneut ein Testfall hinzugefügt, wodurch die TBox mit jeder Iteration an Größe gewinnt und die Anzahl der erfüllten Anforderungen sukzessive steigt. [Per17]

III-2.4. Methodik nach de Nicola et al.

Die UPON-Methode ist ein Ansatz, welcher Software-Entwickler in den Vordergrund rückt und aus den Schritten der Anforderungserhebung, Analyse, Design, Implementierung und Test besteht. Software-Entwickler werden im Rahmen der TBox-Entwicklung zur Erstellung von Glossaren sowie UML-Modellen der die Ontologie nutzenden Applikationen eingebunden, sowie letztlich beim Test der Ontologie. Die Software-Entwickler definieren mit Hilfe der in ihren Applikationen genutzten Begriffen ein Glossar, welches die Grundlage für die Entwicklung der TBox liefert. Hierbei wird zunächst ein semantisches Netz basierend auf dem Glossar erstellt und in UML-Diagrammen modelliert, welches mit den Software-Entwicklern diskutiert und damit auf Konsistenz geprüft wird. Das semantische Netz ist dann Ausgangslage für die Entwicklung der eigentlichen TBox. Der Test der erstellten TBox wird durchgeführt, indem einerseits Reasoner genutzt werden, um die semantische Qualität zu überprüfen. Anderseits wird die pragmatische Qualität durch die Software-Experten überprüft, indem sie die erhobenen Anforderungen mit dem erstellten ontologischen Artefakt vergleichen. [NMN09]

Die auf der UPON basierende Methode UPON light verschlankt den Entwicklungsprozess der UPON Methode und bezieht darüber hinaus auch Domänenexperten in den frühen Phasen (insb. Definition des relevanten Lexikons) mit ein. Darüber hinaus basiert die Methodik nicht auf der Nutzung von UML-Diagrammen, sondern tabellarischen Darstellungen. [NiMi16]

III-2.5. Methodik nach Braun et al.

Die Ontology-Maturing-Methode ist ein Domänenexperten-zentrischer Ansatz, der in Abbildung III.5 dargestellt ist. Der Ansatz sieht eine starke werkzeugtechnische Unterstützung bereits in den frühen Phasen der Ontologie-Entwicklung vor. Die Phasen der Methode sind aufgeteilt in die Entstehung von Ideen, die Konsolidierung in einer Gemeinschaft (englisch „Community“), die Formalisierung und die Axiomatisierung. Dabei werden alle Schritte durch Domänenexperten durchgeführt, indem sie in einem Werkzeug, welches einer Vielzahl von Experten gleichzeitig zur Verfügung steht, neue Begriffe eingeben, deren Bedeutung spezifizieren und Verbindungen zu anderen Begriffen herstellen. Durch die standardisierte Eingabemöglichkeit ergibt sich die Möglichkeit, die durch den Nutzer einzugeben Informationen so zu nutzen, dass selbst die Axiomatisierung mit Hilfe dieser Eingaben erfolgen kann. Die Darstellung erfolgt dabei für den Domänenexperten stets als mit Bildern angereicherten semantischen Netzen und verbirgt damit einen großen Teil der dahinterliegenden Semantik. [Bra+07]

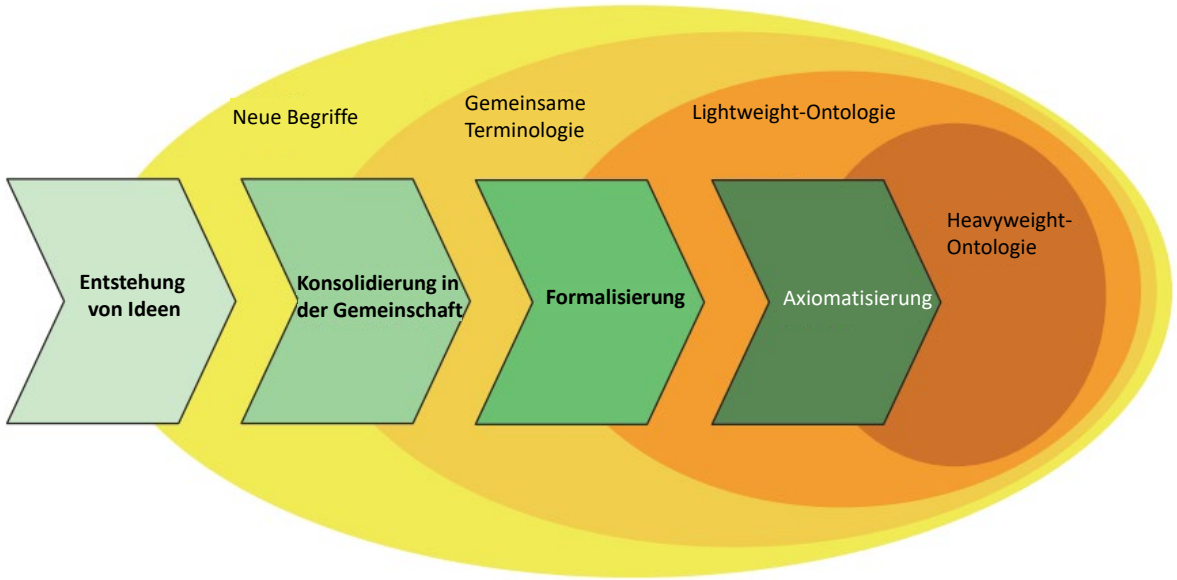

Abbildung III.5.: Methodik nach [Bra+07]

III-2.6. Methodik nach Halilaj

Die VoCol-Methodik basiert auf einem Round-Trip-Modell (siehe Abbildung III.6) mit den Phasen Modellierung, Vervielfältigung und Test, welches sich an Software-Experten und Ontologie-Experten richtet. Dabei beginnt das Vorgehen mit der Definition bzw. Modellierung eines Vokabulars. Hierbei wird eine initiale TBox erstellt, welche iterativ vervollständigt wird. In der Phase der Vervielfältigung werden Mappings genutzt, um aus existierenden Datenquellen (z.B. SQL-Datenbanken) eine ABox zu erzeugen, welche die vorher modellierte TBox nutzt. Vervielfältigt werden somit die Tripel in der ABox. Die Mappings beinhalten

Verknüpfungen zwischen den in der Datenquelle (z.B. SQL-Datenbanken) vorliegenden Daten und den Definition in der TBox, wobei bei Ausführung eine ABox erstellt wird (siehe auch IV-1). Dieses Vorgehen ist notwendig, um im nächsten Schritt eine Verifikation mit Hilfe von SPARQL-Abfragen durchzuführen, da diese sich auf die ABox beziehen können. In der Test-Phase werden Kompetenzfragen aus einem Anwendungsfall generiert und derart formalisiert, dass automatisiert SPARQL-Abfragen abgeleitet werden können. Diese SPARQL-Abfragen werden auf die vorher erstellte ABox ausgeführt, wobei getestet wird, ob die Rückgaben der SPARQL-Abfragen den Antworten der Kompetenzfragen entsprechen. Ist dies nicht der Fall, wird eine weitere Iteration, beginnend mit der Modellierungsphase, durchgeführt. Um die verschiedenen Versionsstände der erstellten TBox zu managen wird ein Versionsmanagementsystem eingesetzt. [Hal+16]

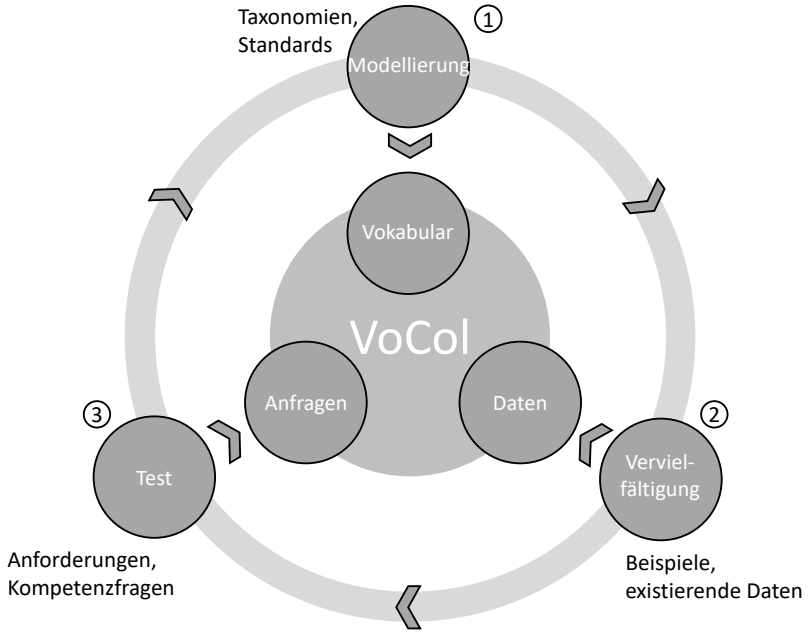

Abbildung III.6.: Methodik in Anlehnung an [Hal+16]

III-2.7. Methodik nach de Almeida Falbo

Bei dem in [Alm14] beschriebenen Ansatz handelt es sich um einen iterativen Ansatz, bestehend aus fünf Phasen. In der ersten Phase werden die Anforderungen in Form von Kompetenzfragen erhoben und der Zweck der Ontologie textuell spezifiziert. Im zweiten Schritt erfolgt eine semi-formale Ontologie-Erstellung. Hierbei werden alle wesentlichen Begriffe und Relationen des im ersten Schritts erstellten Glossars mit Hilfe eines UML-Profils spezifiziert. Darüber hinaus erfolgt parallel eine Formulierung logischer Axiome zur weiteren Spezifikation der Inhalte des UML-Diagramms. Um im Rahmen der Modellierung eine möglichst hohe Wiederverwendung anzustreben, verweist der Ansatz auf die Nutzung von Ontologie-Design-Mustern (ODM) wie in Abschnitt III-2.1 dargestellt. Um die Struktur zwischen den ODMs darzustellen, nutzen die Autoren UML-Paketdiagramme. Im dritten Schritt erfolgt das Design der Ontologie, basierend auf dem vorher erstellten UML-Modell. Hierzu wird

das UML-Modell in ein OWL-Modell überführt und um weitere Semantik, welche in UML nicht darstellbar ist, ergänzt. Hierbei wird auch die vorher definierte Modularisierung der UML-Diagramme berücksichtigt. Im vierten Schritt wird die Ontologie in einer geeigneter Werkzeugumgebung implementiert. Der fünfte Schritt befasst sich mit Tests der erstellten Ontologie. [Alm14]

III-2.8. Bewertung und Auswahl der Methoden zur Ontologie-Entwicklung (TBox)

Die in den bisherigen Abschnitten vorgestellten Ansätze zur Ontologie-Entwicklung (TBox) werden in diesem Abschnitt dahingehend untersucht, inwiefern sie für die Erreichung des Forschungsteilziels II herangezogen werden können. Hierzu werden die Ansätze zunächst mit den Anforderungen abgeglichen, welche für die Erreichung des Forschungsteilziels II erfüllt sein müssen. Basierend auf diesem Abgleich wird anschließend diskutiert, welchen spezifischen Beitrag ein Ansatz für die Erreichung des Forschungsteilziels II im Rahmen dieser Arbeit leistet. Die Gegenüberstellung zwischen Ansätzen und Anforderungen ist in Tabelle III.1 zu finden und wird nachfolgend erläutert.

Tabelle III.1.: Bewertung der TBox Entwicklungsmethoden

Referenz	**Anforderung**					
	A2.FA1	A2.FA2	A2.FA3	A2.FA4	A2.NFA1	A2.NFA2
Presutti et al. [Pre+09]	X	(X)	X	-	-	(X)
Pinto et al. [PST04]	(X)	-	-	-	X	-
Peroni [Per17]	X	X	-	-	-	-
de Nicola et al. [NMN09]	-	-	-	X	X	X
Braun et al. [Bra+07]	-	-	(X)	-	X	X
Halilaj et al. [Hal+16]	-	(X)	(X)	X	-	-
Almeida Falbo [Alm14]	X	-	(X)	-	(X)	(X)

„X“ - erfüllt; „(X)“ - teilweise erfüllt; „-“ - nicht erfüllt

Hinsichtlich der Modularisierung erstellter TBoxen (A2.FA1) liefert insbesondere das Konzept der Ontologie-Design-Muster, welche in den Methoden von [Pre+09] und [Alm14] verwendet werden, einen zentralen Beitrag für die Erreichung des Forschungsziels II, weshalb existierende Forschungsergebnisse im Kontext von Ontologie-Design-Mustern im Rahmen der vorliegenden Arbeit aufgegriffen und auf die Domäne der Automatisierungstechnik angewandt werden. Zentral für den weiteren Verlauf der Arbeit ist dabei insbesondere der konzeptionelle Ansatz bei der Nutzung von Ontologie-Design-Mustern. Im Ansatz von [PST04] sind zwar ebenfalls Ansatzpunkte für die Modularisierung von TBoxen zu finden, allerdings sind diese weniger weitgehend und konsequent verfolgt wie bspw. in [Pre+09], weshalb auf die weitere Betrachtung des Ansatzes von [PST04] verzichtet wird.

Der Ansatz der Test-getriebenen Entwicklung im Rahmen des „eXtreme Design“ entsprechend [Pre+09], aber auch im Rahmen des Ansatzes nach [Per17] sollen im Rahmen dieser Arbeit jedoch nicht weiter verfolgt werden. Ein Test-getriebener Ansatz erfordert den starken Einsatz von Ontologie-Experten in allen Phasen der Ontologieentwicklung, da diese die Tests definieren und ausführen, und erschwert daher die Integration der Domänenexperten (A2.NFA1). Darüber hinaus ist zu erwarten, dass bei einem Test-getriebenen Ansatz die reine Erfüllung der projektspezifischen Anforderungen durch eine Menge von Tests im Vordergrund steht. Dies ist in der Tatsache begründet, dass bei einem Test-getrieben Ansatz direkt aus den Kompetenzfragen eine Menge von Abfragen erstellt wird. Eine Berücksichtigung von Informationsressourcen der Automatisierungstechnik (A2.FA4), oder Normen und Standards im weiteren Sinne, ist dann nicht zwangsläufig Bestandteil der Methodik, da aus Kompetenzfragen erstellte Tests auch ohne Berücksichtigung eines Standards erfolgreich ausgeführt werden können.

Bezüglich der Modularisierung der erstellten ABoxen (A2.FA2) kann von keiner der genannten Methoden ein Bestandteil direkt übernommen werden. Die Ansätze von [Pre+09; Hal+16] führen eine Art Modularisierung der ABox durch, indem SPARQL-Abfragen im Zuge der Erstellung von Tests erzeugt werden. Diese Abfragen sind jedoch, wie für Tests typisch, bezogen auf den Anwendungsfall und nicht auf die Ontologie-Design-Muster bzw. TBox-Module bezogen. Der Ansatz von [Per17] hingegen entwickelt für einzelne „Modelets“ (vergleichbar zu Ontologie-Design-Mustern) Tests. Dies ist insofern von Interesse, als das die Tests sich nicht auf die Erfüllung von Anforderungen in einem spezifischen Anwendungsfall, sondern auf das Vokabular der TBox im Allgemeinen beziehen können. Dies eröffnet grundsätzlich die Möglichkeit die aus dem Vokabular der TBox resultierenden Modellierungsmöglichkeiten der ABox in Form von Tests zu definieren. Da ein Test letztlich eine SPARQL-Abfrage darstellt, in welcher ein gesuchter Teil der ABox modelliert ist, kann dieser Ansatz für die Modellierung von möglichen Modulen der ABox herangezogen werden. Dies soll im Rahmen der vorliegenden Arbeit weiter vertieft werden.

In Bezug auf die Erfüllung der dokumentationsbezogenen Anforderung (A2.FA3) liefert der im Kontext der Methode von [Pre+09] veröffentlichte Ansatz von [Ham17] einen umfangreichen Ansatz, welcher im Rahmen der vorliegenden Methodik aufgegriffen und in Bezug auf die Domäne der Automatisierungstechnik angewandt wird. Zwar berücksichtigen auch die Ansätze von [Hal+16], [Bra+07] und [Alm14] eine Dokumentation der Ergebnisse, allerdings sind diese Ansätze weitaus weniger detailliert und methodisch aufbereitet als in [Ham17].

Die Nutzung von Informationsressourcen der Automatisierungstechnik (A2.FA4), bzw. Standards und Normen im weiteren Sinne, wurde in den Ansätzen von [NMN09] und [Hal+16] erwähnt. Abseits des Hinweises auf Verwendung bzw. Nutzung ist jedoch keinerlei Methodik zur Nutzung dieser Ressourcen erkennbar, weshalb keine Bestandteile übernommen werden.

Hinsichtlich der Domänenexperten-zentrischen Entwicklung (A2.NFA1) und der Nutzung von Beschreibungsmitteln im Rahmen der Entwicklung (A2.NFA2) wird im Rahmen der vorliegenden Arbeit eine Kombination von Aspekten unterschiedlicher Arbeiten vorgenommen. Einerseits wird der Ansatz von [Bra+07] aufgegriffen, in welchem die Domänenexperten eine leichtgewichtige Modellierung der Ontologie vornehmen und somit wesentliche Grundkonzepte definieren. Andererseits wird der Ansatz von [Alm14] bzw. [NMN09] aufgegriffen,

wonach Domänenexperten eine UML-basierte Modellierung im Rahmen der TBox-Erstellung durchführen. Des Weiteren wird der Ansatz von [PST04] als hilfreich erachtet, wonach sich die Domänenexperten in mehreren Iterationen dem Entwicklungsziel nähern und somit ein sukzessiver Projektfortschritt beobachtet werden kann. Dieses iterative Vorgehen liefert vergleichsweise schnell erste Ergebnisse, die in der Gruppe der Stakeholder evaluiert werden können.

III-3. Methodik zur Domänenexperten-zentrischen Entwicklung von Ontologien

Der vorliegende Abschnitt stellt die Artefakte vor, welche zur Erreichung des Forschungsteilziels II notwendig sind. Hierzu stellt Abschnitt III-3.1 zunächst den konzeptionellen Ansatz, die Nutzung von Ontologie-Design-Mustern, in der vorliegenden Arbeit dar. Abschnitt III-3.2 widmet sich dann der Erstellung einer Lightweight-Ontologie bzw. der Suche nach der richtigen Informationsressource als Wissensquelle für die Erstellung der Ontologie. Abschnitt III-3.3 widmet sich der Überführung der Lightweight-Ontologie in eine Heavyweight-Ontologie, sowie der Verifikation und Dokumentation bzw. Dissemination der Entwicklungsergebnisse.

III-3.1. Ontologie-Design-Muster

Die nachfolgenden Abschnitte stellen den konzeptionellen Ansatz der Nutzung von Ontologie-Design-Mustern im Kontext der vorliegenden Arbeit vor. Hierzu widmet sich der Abschnitt III-3.1.1 zunächst der Beschreibung der Charakteristika von Ontologie-Design-Mustern und überträgt das Konzept auf die Domäne der Automatisierungstechnik. Abschnitt III-3.1.2 stellt dann einen methodischen Überblick zur Ontologie-Entwicklung basierend auf Ontologie-Design-Mustern in der Domäne der Automatisierungstechnik vor.

III-3.1.1. Charakteristika von Ontologie-Design-Mustern und Übertragung auf die Automatisierungstechnik

Der Gedanke der Nutzung von Ontologie-Design-Mustern entspringt dem Gedanken der Nutzung von Software-Design-Mustern im Rahmen des Software-Engineerings [Rei00]. Dabei beschreibt ein Software-Design-Muster ein abstraktes, wiederkehrendes Muster in Software, welches Software-Entwicklern die Bearbeitung wiederkehrender Problemkontexte ermöglicht [Cop98]. Im Gegensatz zu Regeln werden Software-Design-Muster nicht durch Computer ausgeführt oder analysiert, stattdessen werden Software-Design-Muster durch Software-Architekten mit einem gewissen Maß an Einblick, Geschmack, Erfahrung und einem Stück weit Ästhetik angewandt [Cop98]. Gleiches gilt für den Entwurf von Ontologie-Design-Mustern [Rei00; Gan05]. Ein Ontologie-Design-Muster ist eine Modellierungslösung, die ein wiederkehrendes Ontologie-Design Problem löst [Rei00]. Grundsätzlich gibt es eine Vielzahl unterschiedlicher Typen von Ontologie-Design-Mustern, bspw. Reasoning Design-Muster,

Namens-Design-Muster, Architektur-Design-Muster und viele mehr (siehe [GaPr09]). Im Rahmen dieser Arbeit sind insbesondere inhaltsbezogene Ontologie-Design-Muster (englisch: „Content Ontology Design Pattern“, kurz „ODM“) von Interesse, welche entsprechend [Jan+16] genutzt werden, um wiederkehrende Muster in komplexen Ontologien im Sinne von wiederverwendbaren Modulen zu modellieren.

Entsprechend [Rei00] können ODMs eine geteilte Sprache für die Kommunikation von Eindrücken und Erfahrungen von Design-Problemen und ihren Lösungen sein, wobei ein ODM das Wissen zur Lösung formal spezifiziert. Entsprechend [GaPr09] ist ein ODM definiert als: „distinguished ontologies. They address a specific set of competency questions, which represent the problem they provide a solution for“. Wie in [Gan05] ersichtlich ist, bezieht sich ein ODM dabei stets auf einen generischen, projektübergeifenden Anwendungsfall (z.B. „Beschreibung von Merkmalen von Montageprozessen“) und einen spezifischen, projektbezogenen Anwendungsfall (z.B. „Beschreibung von Merkmalen eines der Montageprozesse in Werk “Hamburg„“) [GaPr09]. Auf dieser Basis beruht die Definition III.1, welche im Rahmen dieser Arbeit angewandt wird.

Definition III.1 (Ontologie-Design-Muster): *Ein Ontologie-Design-Muster ist eine Menge an Begriffen und Relationen, die innerhalb einer Domäne weitreichend bekannt sind und für die Erfüllung einer generellen Menge ontologischer Anforderungen dienen, welche in einer Vielzahl von speziellen Anwendungsfällen auftauchen.*

Diese Definition trägt grundsätzlich für das gesamte ontologische Spektrum (siehe Abschnitt I-2.2.3) und bedarf einer weiteren Eingrenzung in Form von Charakteristika eines ODM. Eine wesentliche Anforderung an die Methodik dieser Arbeit ist, dass Domänenexperten und Software-Entwickler möglichst stark involviert werden, um dem Mangel an Ontologie-Experten Rechnung zu tragen. Um dies zu erreichen, werden ODMs auf Basis der Informationsressourcen der Domänenexperten und Software-Entwickler erstellt. Durch die Nutzung von Informationsressourcen der Domänenexperten und Software-Entwickler, basieren die erstellten ODMs auf deren Wissen, was das Verständnis dieser Experten von der zu erstellenden Ontologie fördert und eine weitreichende Einbindung in die Modellierung ermöglicht. Dies ist einerseits im Rahmen der Erarbeitung relevanter Begriffe und Relationen auf Lightweight-Ebene möglich als auch im Rahmen der ABox-Entwicklung (siehe Kapitel IV). Um dieser Rahmenbedingung gerecht zu werden, werden die Charakteristika, die ein ODM ausmachen, in Anlehnung an [GaPr09] wie folgt definiert:

1. Ein ODM ist kognitiv relevant: Unterstützt das Verständnis einer Domäne

2. Ein ODM ist linguistisch relevant: Unterstützt die Erklärung von Sachverhalten einer Domäne

3. Ein ODM ist ein „Best practice“: Hat sich in verschiedenen Kontexten als effizient anwendbar erwiesen

4. Ein ODM formalisiert einen „Community“-Konsens: Begriffe und Relationen sind den Experten einer Domäne bekannt und finden breite Zustimmung

5. Ein ODM ist „Computational“: Ist in einer Maschinen-interpretierbaren Sprache im-

plementiert (z.B. OWL)

6. Ein ODM besitzt terminologische Muster: Die Begriffe und Relationen können zur Beschreibung einer Vielzahl von unterschiedlichen Fällen herangezogen werden

7. Ein ODM besitzt assertionale Muster: Die Anwendung der Terminologie auf verschiedene Fälle zeigt wiederkehrende Muster auf

Die ersten vier Kriterien müssen damit im Wesentlichen bereits durch die Informationsressource an sich erfüllt werden, da die Formalisierung einer Informationsressource die ersten vier Kriterien nicht beeinflusst. Wie in [Hil+17*a] dargelegt, eignen sich insbesondere die, teilweise impliziten, ontologischen Definitionen von Industriestandards für die Modellierung unterschiedlicher Informationsbedarfe im industriellen Kontext. Da nicht anzunehmen ist, dass der gesamte Informationsbedarf aller möglichen industriellen Szenarien ausschließlich mit Industriestandards zu decken ist, werden im Rahmen der Methodik dieser Arbeit zusätzlich zu Industriestandards auch wissenschaftliche Veröffentlichungen (z.B. Konferenz- und Zeitschriftenbeiträge) sowie nicht-wissenschaftliche Veröffentlichungen (z.B. unternehmensinterne Standards und Richtlinien) berücksichtigt. Es sei jedoch darauf hingewiesen, dass die ersten vier Kriterien lediglich durch Industriestandards voll erfüllt werden können, da für wissenschaftliche Publikationen Kriterium 3 und 4 nur schwer erfüllbar sind und für nicht-wissenschaftliche Veröffentlichungen Kriterium 4 nur schwer erfüllbar ist. Das fünfte Kriterium definiert, dass ein ODM stets in einer maschineninterpretierbaren Sprache zu implementieren ist, um seine Wiederverwendung zu erleichtern. Im Rahmen dieser Arbeit wird hierzu OWL 2 DL entsprechend [Mcva04@] verwendet. Das sechste Kriterium betrifft die Gestalt der terminologischen Box (TBox). Für diese gilt, dass die verwendeten Begriffe und Relationen von einem spezifischen Fall abstrahieren müssen und für die Anwendung in einer Vielzahl von Fällen herangezogen werden können. Für die assertionale Box (ABox), welche im Kapitel IV dieser Arbeit behandelt wird, gilt, dass auf Basis des terminologischen Musters auch assertionale Muster erkennbar sein sollen. Das bedeutet, dass die Struktur der ABox eines ODM ein wiederkehrendes Muster aufweisen soll. Unter Berücksichtigung der Charakteristika eines ODM, die in [Gan05] genannt sind, soll nachfolgend Definition III.2 gelten, welche die Charakteristika um die Sichten der beteiligten Stakeholder ergänzt.

Definition III.2 (Charakteristika eines Ontologie-Design-Muster): *Ein Ontologie-Design-Muster ist aus Sicht eines Domänenexperten oder Software-Entwickler kognitiv und linguistisch relevant und stellt darüber hinaus eine Art von Community-weitem „Best practice“ für die Beschreibung eines Sachverhaltes dar. Aus Sicht von Ontologie-Experten ist ein ODM in einer formalen Sprache (z.B. OWL) darstellbar und somit maschineninterpretierbar, wobei in verschiedenen Anwendungen sowohl terminologische als auch assertionale Muster erkennbar sind.*

Basierend auf dieser Charakterisierung und unter Bezug auf Abschnitt I-2.2.1 lassen sich die bisherigen Definitionen um den Ausdruck III.1 ergänzen, um die Bestandteile eines ODM auf Modellebene zu beschreiben. Dies bezieht sich auf die in Definition III.2 genannte Sicht der Ontologie-Experten und stellt eine Verbindung zu den in Definition III.1 genannten „Begriffen und Relationen“ her. Ein ODM besteht demnach aus einer Menge an Tripeln T. Die Menge der Tripel teilt sich in die zwei Teilmengen der Tripel der TBox (T_{TBox}) und der

ABox ($T_{ABox}(x)$) auf. Ein Tripel t, welcher Teil der Menge der Tripel T ist, besteht aus drei Entitäten e. Jede Entität e ist Teil der Menge aller Entitäten E und repräsentiert entweder Subjekt s, Prädikat p oder Objekt o. Subjekte und Objekte stellen einen der in Definition III.1 genannten Begriffe dar, während Prädikate die in Definition III.1 genannten Relationen darstellen. Die Tripel der ABox können sich je nach aktuell betrachtetem Fall ändern, weshalb die Menge ($T_{ABox}(x)$) von der Zeit x abhängig ist.

$$\begin{gathered} T = T_{TBox} \cup T_{ABox}(x) \\ e \in E : e = s \vee p \vee o \\ t \in T : t = \{e_1 = s \in S, e_2 = p \in P, e_3 = o \in O, \} \end{gathered} \tag{A.III.1}$$

Um die in Definition III.2 genannte Wiederverwendbarkeit der terminologischen und assertionalen Muster eines ODM auf Modellebene zu beschreiben, wird Ausdruck III.2 herangezogen. Kommt ein ODM im Rahmen zweier Ontologie-Entwicklungsprojekte mit ähnlichen Anforderungen zum Einsatz ($Fall1$ und $Fall2$), so ist die Schnittmenge der Tripel der TBox nicht null und die Schnittmenge der Tripel der ABox ist ebenfalls nicht null (siehe Ausdruck III.2). Daraus folgt, dass Teile der TBox und Teile der ABox in beiden Fällen auftreten. Dieser Ausdruck stellt die Wiederverwendbarkeit von Tripeln der TBox und ABox dar und muss zwingend für ein ODM gelten.

$$\begin{gathered} T_{TBox-Fall1} \cap T_{TBox-Fall2} \neq \emptyset \\ T_{ABox-Fall1}(x_1) \cap T_{ABox-Fall2}(x_2) \neq \emptyset \end{gathered} \tag{A.III.2}$$

Neben einer Charakterisierung eines ODM ist es notwendig, die zugrunde liegende Entwicklungsmethodik so zu strukturieren, dass die Entwicklungsmethodik möglichst geringe Hürden für die Wiederverwendung eines ODM schafft. Hierzu ist es nötig, Qualitätskriterien zu definieren, die als Optimierungsziel im Rahmen der Ontologie-Entwicklung herangezogen werden können. Als Zielgrößen für die Gestaltung eines ODM werden in [Ham17] folgende Qualitätskriterien herangezogen:

- Funktionale Eignung: Grad, zu dem ein ODM die in einem Projekt definierten Anforderungen erfüllt.
- Nutzbarkeit: Grad, zu dem ein ODM in einem Projekt zur Zielerreichung eingesetzt werden kann, unter Berücksichtigung von Effektivität und Effizienz.
- Wartbarkeit: Grad der Effektivität und Effizienz, mit der ein ODM (und aus ODMs resultierende Ontologien) angepasst und modifiziert werden, nachdem sie in einem Projekt eingesetzt wurden.
- Kompatibilität: Grad, zu dem ein ODM erfolgreich wiederverwendet und mit anderen ODMs in eine neue Ontologie integriert werden kann.
- Resultierende Performanceeffizienz: Kann in Form von durchschnittlichen Zeiten für Abfragen, Inferenzen, oder Konsistenzprüfungen bestimmt werden.

In [Ham17] werden darüber hinaus Sub-Merkmale (siehe Tabelle A.1 im Anhang) definiert, sowie Indikatoren (siehe Tabelle A.2 und Tabelle A.3), welche auf die Erfüllung von Qualitätsmerkmalen hinweisen. Dies dient als Basis für Definition III.3.

Definition III.3 (Designziel bei der Verwendung von Ontologie-Design-Mustern): *Das Designziel im Entwurf von Ontologie-Design-Mustern ist eine Minimierung der Hürden der Wiederverwendung, durch Optimierung der Qualitätsmerkmale: Funktionale Eignung, Nutzbarkeit, Wartbarkeit und Kompatibilität. Je höher der Erfüllungsgrad aller Qualitätsmerkmale und der zugehörigen Sub-Merkmale (siehe Tabelle A.1), desto höher ist das Wiederverwendungspotential eines ODM einzustufen [Ham17]. Die genannten Merkmale werden daher als Design-Richtlinien an den entsprechenden Stellen der Entwicklungsmethodik in der vorliegenden Arbeit herangezogen.*

III-3.1.2. Methodischer Überblick zur TBox-Entwicklung basierend auf Ontologie-Design-Mustern

Die für die Nutzung von Ontologie-Design-Mustern empfohlene Entwicklungsmethodik ist das „eXtreme Design“, welches auf den Prinzipien der „starken Einbindung von Kunden und ihrem Feedback“, „anforderungsgetriebenen Entwicklung“, „iterative und inkrementelle Entwicklung“, sowie „Test-getriebene Entwicklung“ beruht. Dabei ist der Ablauf dadurch charakterisiert, dass Anforderungen in enger Abstimmung mit dem Kunden in Form von Kompetenzfragen aufgenommen und in SPARQL-Abfragen übersetzt werden („test-driven“). Mit Hilfe der SPARQL-Abfragen werden passende ODMs gesucht oder entwickelt. Dabei ist essenziell, dass keine Wasserfallstrategie in der Entwicklung verfolgt wird, sondern eine Modulweise (ODM-weise) Entwicklung erfolgt („iterative Entwicklung“), was zu einem „divide-and-conquer“ Paradigma führt. In diesem Paradigma werden stets Teile der Anforderungen gruppiert und mit Hilfe eines ODM abgearbeitet, bevor eine neue Anforderungsgruppe fokussiert wird. [BHP16]

Während im Rahmen der Methodik dieser Arbeit die ersten drei Prinzipien übernommen werden können, ist dies für die Test-getriebene Entwicklung nicht möglich, was zu einer essentiellen Anpassung der Methodik führt. Der Grund hierfür liegt darin, dass die Methodik in dieser Arbeit eine starke Einbindung von Domänenexperten und Software-Entwicklern vorsieht und ODMs entwickelt werden, welche auf Basis von Informationsressourcen dieser Experten beruhen (z.B. Industriestandards). Eine Test-getriebene Entwicklung ist nicht zielführend, da diese statt Domänenexperten und Software-Entwicklern die Ontologie-Experten fokussiert und diesen das Wissen fehlt um Tests basierend auf dem Vokabular von Industriestandards zu definieren. Die in dieser Arbeit definierte Methodik zur Entwicklung der TBox basiert daher auf den Prinzipien der anforderungsgetriebenen Entwicklung, des iterativen und inkrementellen Vorgehens, sowie der Domänenexperten-zentrischen Entwicklung. Dabei sind die in Abbildung III.7 dargestellten Aufgabenbereiche und Aufgaben durchzuführen. Es sei darauf hingewiesen, dass Ontologie-Experten lediglich im Rahmen der Heavyweight-Modellierung und Dissemination eine entwickelnde Rolle einnehmen und somit lediglich mit der Formalisierung und Dokumentation befasst sind.

Basierend auf den erhobenen Anforderungen mit Hilfe der Methodik aus Kapitel II, welche

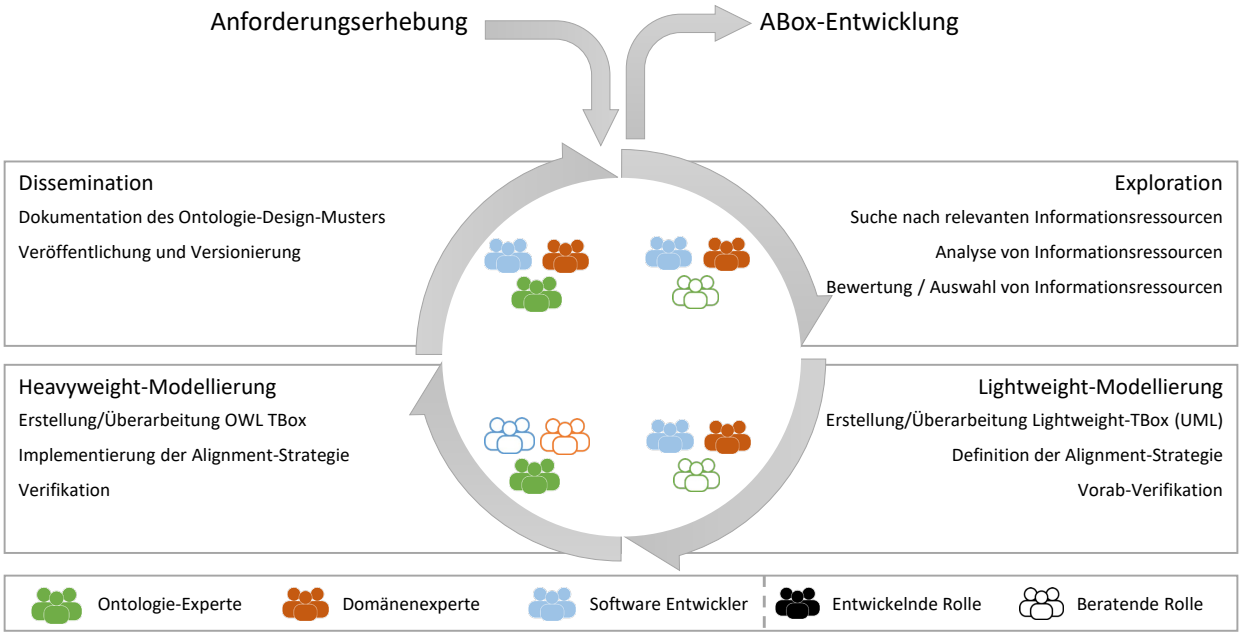

Abbildung III.7.: Iterativer TBox-Entwicklungsprozess basierend auf Ontologie-Design-Mustern

dem Prinzip der starken Einbindung von Domänenexperten und Software-Entwicklern durch integrative Anforderungserhebung von System und Ontologie folgt, wird zunächst explorativ nach geeigneten Informationsressourcen gesucht, welche die Anforderungen erfüllen können. Anschließend werden diese Informationsressourcen analysiert und bewertet, sowie bei entsprechender Eignung für die Entwicklung ausgewählt. Im Anschluss erfolgt eine Lightweight-Modellierung der Begriffe und Relationen aus den ausgewählten Informationsressourcen. Für den Fall, dass im Rahmen der Exploration bereits eine Lightweight-Ontologie als geeignet identifiziert wurde, wird diese etwaig überarbeitet. Dabei sei darauf hingewiesen, dass dies stets unter Wahrung der Konsistenz der zugrundeliegenden Informationsressource erfolgt. Das heißt, dass lediglich Präzisierungen und Erweiterungen basierend auf der Informationsressource erfolgen, nicht basierend auf dem Projektkontext. Für die Lightweight-Modellierung werden die gefundenen Informationsressourcen einzeln mit Hilfe von UML-Klassendiagrammen modelliert. Nachdem eine entsprechende Anzahl an Iterationen durchgeführt wurde und mehr als eine Informationsressource modelliert wurde, wird eine Alignment-Strategie definiert. Dabei wird zunächst angenommen, dass jede gefundene Informationsressource ein potenzielles ODM darstellt, da zu diesem Zeitpunkt noch nicht bewertet werden kann, inwiefern die Qualitätsmerkmale (siehe Abschnitt III-3.1.1) zutreffen. Die Alignment-Strategie stellt dar, wie die für die Erfüllung der Anforderungen notwendigen ODMs miteinander kombiniert werden sollen. Hierzu werden die von [BHP16] angeführten Mechanismen zur Kombination von ODMs auf Lightweight-Ebene angewandt. Abschließend wird eine Vorab-Verifikation durchgeführt, sodass die durch einen oder mehrere ODMs angesprochenen Anforderungen identifiziert und ihre Erfüllung verifiziert werden kann.

Im Rahmen der Heavyweight-Modellierung werden die in der vorherigen Phase identifizierten Informationsressourcen einzeln entsprechend ihrer UML-Klassendiagrammmodelle in eine Heavyweight-Variante überführt und mit ergänzenden Axiomen versehen, welche durch

die Modellierung mit Klassendiagrammen nicht darstellbar sind. Nun kann eine erste Betrachtung der Qualitätsmerkmale erfolgen. Sollten Domänenexperten, Software-Entwickler und Ontologie-Experten ein ODM entsprechend der Definition III.1 und Definition III.2 ausgemacht haben und eine entsprechende Eignung als ODM erkennbar sein, wird dieses für die spätere Dissemination vorgemerkt. An dieser Stelle wird die vorher definierte Alignment-Strategie implementiert, sofern bereits mehr als ein Heavyweight-ODM existiert, und auf diese Weise die zu entwickelnde Ontologie erstellt. Zur Überprüfung der Anforderungserfüllung wird eine Verifikation durchgeführt. Im Rahmen der letzten Phase werden die erkannten ODMs dokumentiert und veröffentlicht. Dies hat zum Vorteil, dass die TBox auch außerhalb des vorliegenden Projektkontextes einer Weiterentwicklung unterworfen ist, sofern das ODM die in Definition III.3 genannten Qualitätskriterien ausreichend erfüllt.

Betrachtet man den Artefaktfluss zwischen den einzelnen Methodenbausteinen, kann Abbildung III.8 herangezogen werden. Demnach besteht die Methode aus zwei wesentlichen Methodenbausteinen, welche bestimmte In- und Output-Artefakte besitzen. Basierend auf dem Ergebnis der Ontologie-Anforderungserhebung, dem Ontologie-Anforderungsmodell, befasst sich der erste Methodenbaustein mit der Entwicklung einer Lightweight-Ontologie basierend auf Ontologie-Design-Mustern. Für diesen Methodenbaustein dienen einerseits existierende Informationsressourcen mit textuellen Spezifikationen als Input, welche im Rahmen des Methodenbausteins in eine Lightweight-Ontologie überführt werden. Andererseits auch existierende Lightweight-Ontologie-Design-Muster, welche im Rahmen der Durchführung dieser Methode in früheren Interationszyklen oder anderen Projekten entstanden sind. Outputs der Methode sind projektunabhängige, semi-formale Ontologie-Design-Muster sowie eine semi-formale Alignment-Ontologie. Letztere stellt die Verbindungen zwischen verschiedenen Lightweight-Ontologie-Design-Mustern dar und dient der Anpassung dieser auf einen projektspezifischen Anwendungsfall. Der zweite Methodenbaustein befasst sich mit der Formalisierung der Lightweight-Ontologien, um die TBox der Heavyweight-Ontologie zu definieren. Für diesen Methodenbaustein dient neben den bisher erstellten Lightweight-Ontologien auch das Ontologie-Anforderungsmodell als Input, sowie existierende Heavyweight-Ontologie-Design-Muster. Resultat der Durchführung dieses Methodenbausteins sind sowohl projektunabhängige, formale Ontologie-Design-Muster, als auch eine formale Alignment-Ontologie. Darüber hinaus umfasst der Methodenbaustein auch eine Dissemination der erstellten Ontologie-Design-Muster in Form einer entsprechenden Dokumentation sowie Veröffentlichung zum Zwecke der Wiederverwendung.

III-3.2. Ontologie-Erstellung (Lightweight-TBox)

Die nachfolgenden Abschnitte beschreiben den Methodenbaustein „Ontologie-Erstellung (Lightweight-TBox)“, welcher sich mit der Erstellung der zu entwickelnden Ontologie in einer leichtgewichtigen (englisch: „Lightweight“) Repräsentation befasst. Hierzu stellt Abschnitt III-3.2.1 zunächst dar, wie geeignete Informationsressourcen für die Entwicklung der zu entwickelnden Ontologie identifiziert werden können. Abschnitt III-3.2.2 stellt dann dar, wie auf deren Basis eine Lightweight-Ontologie erstellt wird.

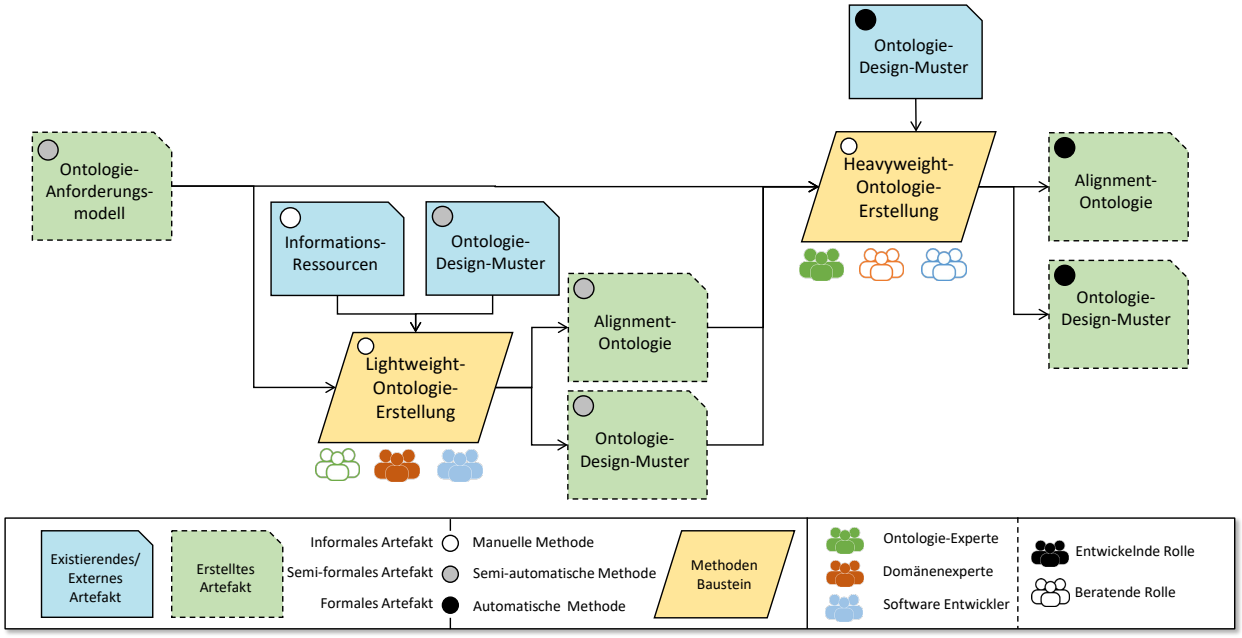

Abbildung III.8.: Artefaktflussorientierter Überblick zur TBox-Entwicklung der Ontologie

III-3.2.1. Suche und Bewertung relevanter Informationsressourcen

Das Ziel dieser Aktivität ist die Identifikation relevanter Informationsressourcen als Wissensquellen für die Erstellung der Ontologie, wobei für die Informationsressourcen die Funktionale Eignung und Nutzbarkeit (siehe Abschnitt III-3.1.1) bestimmt werden soll. Da die Anforderungen informell in Form von Kompetenzfragen definiert sind und im Falle von nicht als Ontologien formalisierten Informationsressourcen die Begriffe und Relationen hierin ebenfalls informell spezifiziert sind, bietet es sich an, mit einem Glossar relevanter Begriffe der Kompetenzfragen und Antworten die Basis für die Bestimmung der Funktionalen Eignung und Nutzbarkeit zu definieren. Aus diesem Glossar sollte hervorgehen, welche semantische Bedeutung den Begriffen und Relationen zugesprochen wird. Diese Aufgabe ist durch diejenigen Stakeholder durchzuführen, welche die Anforderungen initial definiert haben. Im Anschluss ist es optional möglich, den Begriffen eine Nennungshäufigkeit zuzuordnen, indem die Kompetenzfragen und Antworten per Textsuche analysiert werden. Dies kann bei einer hohen Anzahl von Begriffen und Relationen eine Priorisierung ermöglichen. Das Resultat der Glossarerstellung sollte dann die verschiedenen Begriffe, eine textuelle Erläuterung, sowie eventuell eine Nennungshäufigkeit besitzen.

III-3.2.1.1. Suche nach Informationsressourcen

Bezüglich der Art der Informationsressourcen wird im Rahmen dieser Arbeit in drei Kategorien (siehe Abschnitt III-3.1.1) unterschieden. Dies sind:

- Industriestandards
- Wissenschaftliche Publikationen

- Nicht-wissenschaftliche Publikationen

Nach der Erstellung des Glossars kann auf zweierlei Arten verfahren werden, um relevante Informationsressourcen zu identifizieren, sowie zu diesen gehörige Lightweight-Ontologie-Design-Muster. Die erste Möglichkeit ist ein semi-strukturiertes Experteninterview, in welchem Domänenexperten und Software-Entwickler, auch aus anderen Projekten, interviewt werden. Hierzu wird aus einem Glossareintrag eine Frage generiert, die dem Experten gestellt wird. Dazu kann einerseits die Bedeutung eines Begriffs abgefragt werden, um bei einer etwaigen Deckung mit der gewünschten Erläuterung die Herkunft dieser Definition zu erfragen. Andererseits können auch aus den Erläuterungen direkt Fragen erstellt werden, die auf die Kenntnis von Informationsressourcen zu der gewünschten Erläuterung abzielen.

Die zweite Möglichkeit ist die Durchführung einer strukturierten Suche in Datenbanken, basierend auf Schlüsselbegriffen, welche dem Glossar entnommen werden. Hierzu wird mit Hilfe der im Glossar genannten Begriffe eine Stichwortsuche in entsprechenden Datenbanken durchgeführt, wodurch insbesondere Industriestandards und wissenschaftliche Veröffentlichungen gefunden werden können.

Unabhängig davon, welche der beiden Möglichkeiten zur Suche nach Informationsressourcen gewählt wird, müssen funktionale Eignung und Nutzbarkeit der Informationsressourcen bestimmt werden. Dies ist in den nachfolgenden Abschnitten III-3.2.1.2 und III-3.2.1.3 dargestellt. Es sei darauf hingewiesen, dass eine exakte Berechnung der folgenden Formeln für jede gefundene Informationsressource nicht nicht zwingend notwendig sein muss. Vielmehr dient die Darstellung der Formeln dazu, bei vorliegenden von unterschiedlichen Informationsressourcen eine fundierte Entscheidung treffen zu können. Im Zweifel kann eine Berechnung erfolgen, dennoch sollte eine Einschätzung der Kriterien durch Domänenexperten der Regelfall sein, um Aufwand und Nutzen in ein angemessenes Verhältnis zu setzen.

III-3.2.1.2. Bewertung der Funktionalen Eignung

Für die Bestimmung der Funktionalen Eignung auf Basis von Glossaren sind einige Kennzahlen notwendig, sodass unterschiedliche Informationsressourcen hinsichtlich dieses Kriteriums bewertet werden können. Die Informationsressourcen werden hierzu zunächst hinsichtlich der semantischen Bedeutung der verwendeten Begriffe analysiert und deren inhaltliche Eignung (semantische Eignung SE_{IR_i} der Informationsressource) für die Erfüllung der ontologischen Anforderungen aus Sicht der Domänenexperten und Software-Entwickler überprüft. Dies entspricht einer qualitativen Überprüfung der Funktionalen Eignung durch die Domänenexperten und Software-Entwickler und soll insbesondere verhindern, dass offensichtlich ungeeignete Informationsressourcen möglichst früh erkannt werden. Dabei wird die Bestimmung der semantischen Eignung in Anlehnung an die semantische Qualität entsprechend [Bur+05] durchgeführt, wobei die Informationsressourcen einzeln hinsichtlich vier Sub-Kriterien bewertet werden:

- Eignung des Begriffsmodells: Einschätzung der Eignung des enthaltenen Begriffsmodells der Informationsressource für die Beantwortung der Kompetenzfragen der zu entwickelnden Ontologie.

- Verständlichkeit des Begriffsmodells: Für die spätere Verwendung der zu entwickelnden Ontologie ist es von zentraler Bedeutung, dass die verwendeten Begriffe durch Anwender (Domänenexperten und Software-Entwickler) grundsätzlich verständlich sind.

- Konsistenz des Begriffsmodells: Sollte eine Informationsressource eine inkonsistente Verwendung von Begriffen aufweisen, muss diese Inkonsistenz im Rahmen der Ontologie-Entwicklung mitigiert werden. Da dies ein zusätzlicher Aufwand ist, sollte bereits bei der Auswahl der Informationsressourcen auf ein konsistentes Begriffsmodell geachtet werden.

- Eindeutigkeit einzelner Begriffe: Sollte eine Informationsressourcen Mehrdeutigkeiten aufweisen, so sind diese Mehrdeutigkeiten im Rahmen der Ontologie-Entwicklung zu mitigieren, auch dies ist analog zu einem inkonsistenten Begriffsmodell ein zusätzlicher Aufwand.

Um einen Wert für das Kriterium der semantischen Eignung SE_{IR_i} zu bestimmen, bietet sich die Ermittlung des arithmetischen Mittelwerts der vier oben genannten Sub-Kriterien an. Als mögliche Skala kann bspw. eine dreigeteilte Kategorisierung vorgenommen werden, i.S.v. „zutreffend“, „teilweise-zutreffend“ und „nicht zutreffend“.

Um anschließend eine quantitative Bewertung hinsichtlich der Funktionalen Eignung vorzunehmen, werden die in den Informationsressourcen genannten Begriffe und die in den ontologischen Anforderungen genannten Begriffe entsprechend dem Vorgehen von [ViGó12] syntaktisch analysiert. Hierzu wird zunächst die begriffliche Abdeckung $B_A(B_{IR_i})$ bestimmt, wobei die Schnittmenge aus verwendeten Begriffen B_{IR_i} der Informationsressource i und verwendeten Begriffen B_G des Anforderungsglossars G gebildet wird. Diese Schnittmenge stellt die relevanten Begriffe einer Informationsressource in Bezug auf die gestellten Anforderungen dar und wird dann mit Hilfe der Anzahl verwendeter Begriffe im Anforderungsglossar B_G normalisiert. Formel III.3 stellt dies dar. Besitzt ein Anforderungsglossar B_G bspw. vier Begriffe und eine Informationsressource definiert davon zwei Begriffe, so ist der Betrag der Schnittmenge von B_{IR_i} und B_G zwei, wobei die begriffliche Abdeckung $B_A(B_{IR_i})$ dann 0,5 beträgt. Anhand der begrifflichen Abdeckung einer Informationsressource ist erkennbar, wie viele relevante Begriffe eine Informationsressource im Verhältnis zur Anzahl der Begriffe in den Anforderungen enthält. Eine hohe Abdeckung spricht für eine hohe Relevanz für die folgende Modellierung.

$$B_A(B_{IR_i}) = \frac{|(B_{IR_i}) \cap (B_G)|}{|B_G|} \tag{F.III.3}$$

Anschließend kann die Präzision der Informationsressource berechnet werden. Bei der Berechnung der Präzision $B_P(B_{IR_i})$ einer Informationsressource (siehe Formel III.4) wird das Verhältnis aus relevanten Begriffen einer Informationsressource ($|(B_{IR_i}) \cap (B_G)|$) zu allen Begriffen einer Informationsressource betrachtet. Nimmt man in Bezug auf obiges Beispiel an, die Informationsressource B_{IR_i} definiert 10 Begriffe, so wäre bei einer gleichbleibenden Schnittmenge von 2 Begriffen die Präzision $B_P(B_{IR_i})$ gleich 0,2.

$$B_P(B_{IR_i}) = \frac{|(B_{IR_i}) \cap (B_G)|}{|B_{IR_i}|} \tag{F.III.4}$$

Dabei ist bei einer hohen Präzision anzunehmen, dass die analysierte Informationsressource einen starken Fokus auf den in den ontologischen Anforderungen enthaltenen Anwendungsfall besitzt. Da es vorkommen kann, dass eine Informationsressource allein nicht alle Begriffe enthält, welche für die Erfüllung der ontologischen Anforderungen relevant sind, können auch Gruppen von Informationsressourcen gebildet und die Präzision und Abdeckung für eine Gruppe berechnet werden. Dabei wird die Schnittmenge der Begriffe der Informationsressourcen einer Gruppe gebildet und die Präzision und Abdeckung analog zu einzelnen Informationsressourcen berechnet.

III-3.2.1.3. Bewertung der Nutzbarkeit

Für die Bewertung der Nutzbarkeit einer Informationsressource für die Erstellung der zu entwickelnden Ontologie wird nachfolgend ebenfalls eine Reihe von Kriterien definiert, welche einen Vergleich der Nutzbarkeit unterschiedlicher Informationsressourcen erlauben soll. Für die Berechnung werden die im vorherigen Schritt als geeignet erkannten Informationsressourcen herangezogen.

Die erste Kennzahl wird durch den Formalismus F_{IR_i} der Informationsressource dargestellt. Für die Bewertung des Formalismus wird auf das ontologische Spektrum zurückgegriffen (siehe Abbildung I.11), wobei vier Kategorien gebildet werden:

- Existenz eines Glossars
- Existenz einer maschineninterpretierbaren Spezifikation (z.B. XML)
- Existenz einer Lightweight-Ontologie (z.B. UML-Klassendiagramm)
- Existenz einer Heavyweight Ontologie (z.B. OWL)

Je höher der Formalisierungsgrad einer Informationsressource F_{IR_i}, desto vorteilhafter für den folgenden Entwicklungsprozess, da weniger Begriffe und Relationen formalisiert werden müssen. Demnach ist die Nutzbarkeit umso höher, je höher der Formalismus ist. Es sei jedoch darauf hingewiesen, dass die oben genannten Kategorien als Ergänzung zu einer textuellen, menscheninterpretierbaren Spezifikation zu verstehen sind, da eine textuelle, menscheninterpretierbare Spezifikation für das Verständnis im Engineering der zu entwickelnden Ontologie unerlässlich ist.

Abschließend werden die Informationsressourcen hinsichtlich der Implementierungskomplexität im Rahmen des Projektes bewertet, wobei alle beteiligten Stakeholder bei dieser Betrachtung involviert werden. Um die Implementierungskomplexität zu quantifizieren, werden die in [Ham17] vorgeschlagenen Punkte zur Bewertung der Wiederverwendbarkeit von ODMs herangezogen und mit den im bisherigen Verlauf definierten Merkmalen einer Informationsressource kombiniert. Die nachfolgenden Kriterien (Overhead, Modifizierbarkeit, ziel-

bezogene Anpassungsfähigkeit) stammen demnach aus [Ham17] und werden im Rahmen der vorliegenden Methode über Formeln miteinander in Beziehung gesetzt, um den Einfluss auf die Implementierungskomplexität als zusammenfassendes Bewertungskriterium darzustellen.

Der inhaltliche Overhead B_O bringt entsprechend Formel III.5 zum Ausdruck, welcher Anteil der Begriffe und Relationen einer Informationsressource bei deren Nutzung als Overhead mitberücksichtigt werden müssen. Der Overhead entspricht der in Formel III.4 dargestellten Präzision $B_P(B_{IR_i})$ in umgekehrter Logik. Eine wenig präzise Informationsressource besitzt somit einen hohen Overhead an Begriffen, welche im Rahmen der Modellierung u.U. mit berücksichtigt werden müssen, aber für die Erfüllung der Anforderungen nicht von Relevanz sind [Ham17]. Beträgt die Präzision $B_P(B_{IR_i})$ bspw. 0,2, so beträgt der Overhead B_O 0,8. Demnach wären 80% der Begriffe einer Informationsressource Overhead. Der Overhead ist demnach ebenso wie die Präzision eine normierte Kennzahl.

$$B_O(B_{IR_i}) = 1 - B_P(B_{IR_i}) \tag{F.III.5}$$

Für den Fall, dass die Nutzung einer Informationsressource eine andere Informationsressource als Voraussetzung bedingt und sie daher referenziert, senkt dies die Modifizierbarkeit der Informationsressource, bzw. Ontologie oder ODM, da eine enge Verflechtung zwischen Informationsressourcen aufgebrochen werden müsste, um in einem anderen Einsatzkontext eingebunden zu werden [Ham17]. Um diesem Umstand gerecht zu werden, soll eine normierte Kennzahl die Unabhängigkeit IRU_i einer Informationsressource von für ein Projekt nicht notwendigen, anderen Informationsressourcen darstellen. Diese sinkt mit jeder Referenz Ref zu einer anderen Informationsressource, was in Formel III.6 dargestellt ist. Der genaue Funktionsverlauf ist hierbei nicht entscheidend, sondern lediglich die Grenzwerte der Funktion. Der Grenzwert 1 tritt bei keiner Referenz zu anderen, für das Projekt nicht benötigten, Informationsressourcen auf. Bei steigender Anzahl an Referenzen geht die Unabhängigkeit der Informationsressource IRU_i gegen null.

$$IRU_i = 1 - \frac{Ref}{1 + Ref} \tag{F.III.6}$$

Die Unabhängigkeit der Informationsressource hat Einfluss auf die zielbezogene Anpassungsfähigkeit der Informationsressource $IRAF_i$. Letztere bringt zum Ausdruck inwiefern projektspezifische Erweiterungen möglich sind, unter Wahrung der Konsistenz des gesamten Begriffsmodells [Ham17]. Besitzt eine Informationsressource einen hohen Overhead B_O, so ist dies insofern hinderlich, als das bei Ergänzung eines projektspezifischen Begriffs die Konsistenz zu allen existierenden, für das Projekt nicht relevanten, Begriffen überprüft werden muss. Dieses Problem vervielfacht sich mit einer geringen Unabhängigkeit der Informationsressource IRU_i, da auch die Konsistenz mit den Begriffen in den referenzierten Informationsressourcen gewahrt werden muss. Dies tritt bspw. auf, wenn im Rahmen eines Projektes der Begriff „Geschäftsprozess“ zu einer Informationsressource über „Prozesse“ hinzugefügt werden soll, da dieser nicht in der Informationsressource enthalten ist, durch die Anforderungen jedoch gefordert wird. Bei der Definition des Begriffs ist dann auf die Konsistenz mit allen existierenden Begriffen zu achten, auch denen die nicht zwingend für das Projekt von Interesse

sind (siehe Overhead B_O) und denen die sich in für das Projekt nicht notwendigen, referenzierten Informationsressourcen befinden. Dieser Zusammenhang ist Allgemein in Formel III.7 dargestellt.

$$IRAF_i = (1 - B_O) \cdot IRU_i \tag{F.III.7}$$

Als Merkmal für den Vergleich von Aufwänden, welche sich aus der Implementierung der Informationsressource ergeben können, kann die Implementierungskomplexität IRK_i herangezogen werden [Ham17]. Die Implementierungskomplexität fasst im Rahmen der vorliegenden Methode alle bisher dargestellten Kennzahlen zusammen. Entsprechend Formel III.8 liegt eine optimal niedrige Implementierungskomplexität IRK_i vor, wenn die Berechnung den Wert 0 ergibt. Eine hohe Implementierungskomplexität liegt vor, wenn der Wert gegen 1 geht. Je höher der Formalismus F und die semantische Eignung SE_{IR_i}, desto einfacher ist die Implementierung der Informationsressource in eine Ontologie. Im Idealfall liegt bereit eine OWL DL Ontologie vor, der Formalismus ist somit maximal bewertet und die semantische Eignung wurde ebenfalls optimal bewertet. Letzteres deutet auf eine verständliches und konsistentes Begriffsmodell mit eindeutigen Begriffen und einer schätzungsweise hohen Eignung für das Projekt hin. Beide Kriterien werden mit der Summe der zu erreichenden Punkte N normiert. Die Implementierungskomplexität wird durch die in der zielbezogenen Anpassungsfähigkeit $IRAF_i$ enthaltenen Faktoren negativ beeinflusst. Ein hoher Overhead sowie eine Vielzahl von Referenzen zu nicht benötigten Informationsressourcen erhöhen die Implementierungskomplexität. Da $IRAF_i$ bereits normiert ist, wird es mit dem normierten Wert für Formalismus F und die semantische Eignung SE_{IR_i} multipliziert. Eine Gewichtung ist demnach nicht vorgesehen.

$$IRK_i = 1 - \frac{F_{IR_i} + SE_{IR_i}}{N} \cdot IRAF_i; \quad \text{mit} \quad N = F_{max} + SE_{max} \tag{F.III.8}$$

Anhand der hergeleiteten Merkmale kann nun für eine Menge an Informationsressourcen entschieden werden, welche sich eher für eine weitere Verwendung eignen als andere. Es sei an dieser Stelle nochmals darauf hingewiesen, dass diese Darstellung lediglich der Verdeutlichung der Zusammenhänge von beeinflussenden positiven und negativen Faktoren bei der Auswahl von Informationsressourcen für ein spezifisches Projekt dient. Eine exakte Berechnung ist nur dann notwendig, wenn eine entsprechende Einschätzung der Kriterien durch Domänenexperten zu keinem eindeutigen Ergebnis führt.

III-3.2.2. Modellierung von Lightweight-Ontologie-Design-Mustern

Der vorliegende Abschnitt beschreibt das Vorgehen zur Entwicklung von Lightweight-Ontologie-Design-Mustern basierend auf den entsprechend des vorherigen Abschnitts identifizierten Informationsressourcen. Hierzu stellt der Abschnitt III-3.2.2.1 zunächst dar, wie relevante Begriffe und Relationen der Informationsressourcen in eine Lightweight-Ontologie überführt werden können. Abschnitt III-3.2.2.2 stellt anschließend dar, wie mehre-

re Lightweight-Ontologie-Design-Muster miteinander verbunden werden können, bevor Abschnitt III-3.2.2.3 auf eine Vorab-Verifikation eingeht.

III-3.2.2.1. UML-basierte Modellierung von Lightweight-Ontologien

Im Rahmen der Lightweight-Modellierung werden einzelne Informationsressourcen mit Hilfe von UML-Klassendiagrammen als Lightweight-Ontologien modelliert, nachdem die für die Ontologie-Entwicklung relevanten Informationsressourcen entsprechend Abschnitt III-3.2.1 identifiziert wurden. Hierbei ist zunächst zu definieren, welche Modellelemente hierfür genutzt werden sollen, wobei sich für eine Definition der relevanten Modellelemente erneut ein UML-Profil anbietet. Das in [Obj14@] definierte und auf der Arbeit von Brockmanns in [Bro+06] basierende UML-Profil bietet sich für diesen Zweck an. Allerdings richtet sich das in [Obj14@] beschriebene Profil an Ontologen und soll im Rahmen der vorliegenden Methodik durch Domänenexperten und Software-Entwickler anwendbar sein. Daher stellt das nachfolgend vorgestellte Profil lediglich eine leichtgewichtige Abwandlung des Profils aus [Obj14@] dar. Diese leichtgewichtige Abwandlung ist in Abbildung III.9 zu finden. Demnach sind die einzigen Modellelemente „normale“ UML-Klassen, UML-Assiziationen und UML-Generalisierungen. Darüber hinaus wird ein Stereotyp „Equivalence“ eingeführt, welcher zur Modellierung semantischer Äquivalenz genutzt werden kann, im Fall das Begriffe eine identische Semantik aufweisen („positive“) oder sich gegenseitig ergänzen aber nicht äquivalent sind („negative“).

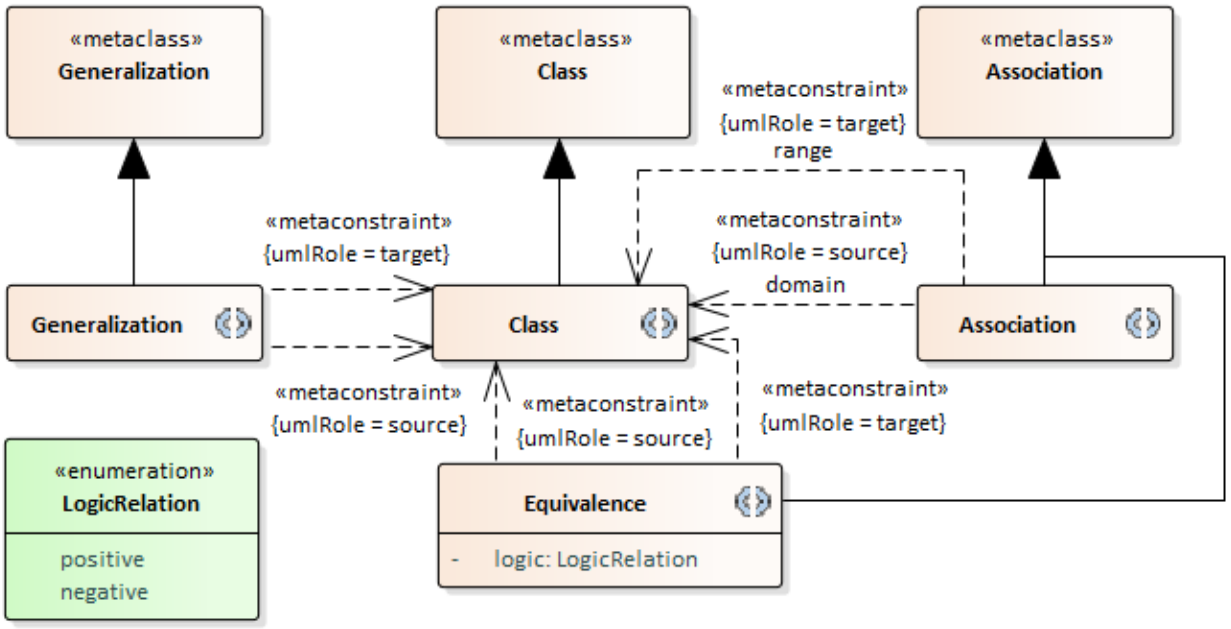

Abbildung III.9.: UML-Profil zur Lightweight-Modellierung von Ontologien

Im Wesentlichen sollen mit Hilfe des Profils Begriffe und Relationen der Informationsressourcen in Form von Klassen, Object- und DatatypeProperties aus OWL 2 DL modelliert werden, sowie die für das Alignment relevanten Operationen. Darüber hinaus soll mit Hilfe

des Profils ausschließlich die TBox modellierbar sein. Um dies zu realisieren, werden für die Beschreibung von Klassen lediglich die „AtomicClasses“, und „ObjectProperties“ [Obj14@] benötigt. Im Gegensatz zu [Obj14@] werden ObjectProperties nicht als Stereotyp von UML-Klassen, sondern UML-Assoziationen, definiert, da dies erheblich schlankere Modelle erzeugt und die Wartbarkeit (siehe Abschnitt III-3.1) erhöht. Im Falle mehrerer Domains und Ranges eines ObjectProperties werden die Verwendungen eines ObjectProperties stets als logisches „ODER“ verstanden, da dies die Mehrzahl der Fälle betrifft. Falls das nicht zutrifft, ist dies über das Notizfeld eines Properties zu spezifizieren und in der Erstellung der Heavyweight-Ontologie entsprechend zu berücksichtigen. DatatypeProperties werden als klassische UML-Attribute des Stereotyps „Class“ definiert.

Das Ziel der Anwendung des gezeigten Profils ist es, die in der Informationsressource i enthaltenen Begriffe und Relationen in einen Zusammenhang zu setzen und auf diese Weise die TBox der Informationsressource T_{TBox_i} zu modellieren. Im Rahmen der Lightweight-Modellierung ist die Menge der in der Informationsressource enthaltenen Begriffe B_{IR_i} von Interesse. Diese Begriffe werden im Rahmen der Modellierung zunächst einer UML-Klasse, einer UML-Assoziation, oder einem UML-Attribut zugewiesen, wobei Mehrfachzuweisungen nicht möglich sind. Dabei sollte darauf geachtet werden, die Informationsressourcen einzeln zu modellieren und erst im Rahmen der Modellierung der Alignment-Ontologie miteinander zu verbinden, sodass einzeln wartbare und möglichst kompatible Lightweight-Ontologien entstehen. Ausdruck III.9 stellt die Logik der Überführung einer Informationsressource i in eine Lightweight-Ontologie dar. Demnach ist ein einzelner Begriff b_{IR_i} einer Informationsressource Teil der Menge aller Begriffe B_{IR_i} der Informationsressource i. Für jeden Begriff b_{IR_i} gilt, dass er in eine UML-Klasse ($Class(b_{IR_i})$), eine UML-Assoziation ($Association(b_{IR_i})$) oder ein UML-Attribut ($Attribute(b_{IR_i})$) überführt wird.

$$\begin{gathered} b_{IR_i} \in B_{IR_i} \\ (\forall b_{IR_i})(Class(b_{IR_i}) \dot{\vee} Association(b_{IR_i}) \dot{\vee} Attribute(b_{IR_i})) \end{gathered} \tag{A.III.9}$$

Um diesen Schritt zu verbildlichen und ein Beispiel für nachfolgende Ausdrücke zu liefern, wird Abbildung III.10 herangezogen. Für eine fiktive Informationsressource i wurden in der Abbildung alle Begriffe modelliert und miteinander in Beziehung gesetzt. Die Begriffe Produkt, Prozess, Technische Ressource sowie Halbzeug und Maschine wurden dabei als Klasse modelliert. Die Relationen „ist Input für“ und „nutzt“ wurden als Assoziationen dargestellt. Darüber hinaus wurde der in Abbildung III.9 gezeigte Stereotyp „Equivalence“ genutzt, um eine Komplementärbeziehung in orange anzuzeigen und Äquivalenzbeziehung in blau. Letztlich wurde der Begriff „Länge“ als Attribut der Klassen Halbzeug und Maschine verwendet.

Da die Modellierung als Lightweight-Ontologie nur einen Zwischenschritt zur Heavyweight-Modellierung darstellt, müssen die einzelnen Modellelemente einer Lightweight-Ontologie logisch mit Modellelementen von OWL, RDFS und RDF verbunden werden können. Dies wird nachfolgend am Beispiel von Abbildung III.10 erläutert. Wird ein Begriff einer UML-Klasse zugewiesen, so entspricht dies dem Äquivalent einer OWL-Klasse, was in Ausdruck III.10 dargestellt ist. Da in RDF die Nutzung von Namensräumen vorgesehen ist (siehe Abschnitt III-1.1), definiert die Informationsressource i daher den Namensraum aller Begriffe und Relationen j der Informationsressource i. Dies ist in Ausdruck III.10 durch die Nutzung des

Doppelpunkts angezeigt. In Bezug auf das Beispiel in Abbildung III.10 sind die Klassen Produkt, Prozess, Technische Ressource sowie Halbzeug und Maschine demnach äquivalent zu einer OWL-Klasse. Die Informationsressource des Beispiels (z.B. „VDI1234“) würde dann den Namensraum der OWL-Klassen definieren. Der gesamte IRI der Klasse „Produkt“ wäre dann bspw. `VDI1234:Produkt`.

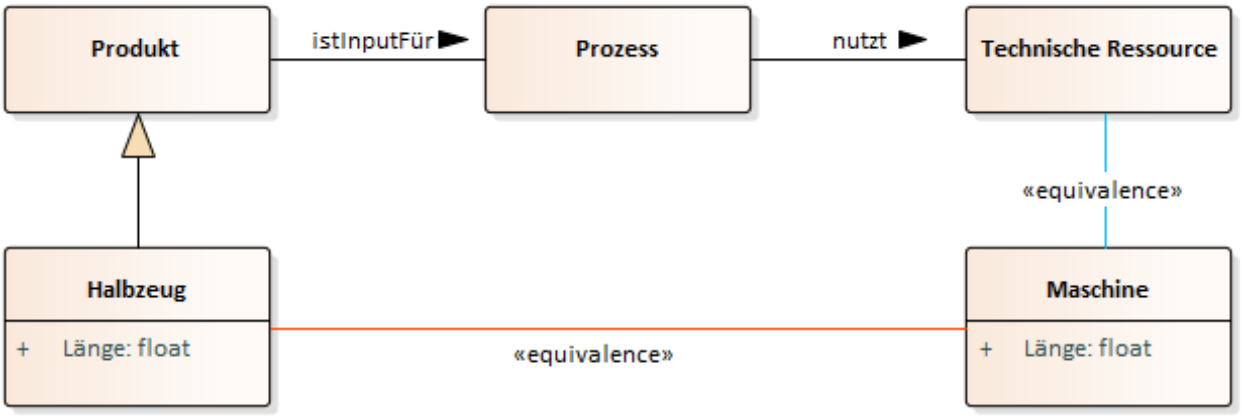

Abbildung III.10.: Beispiel einer Lightweight-Ontologie

$$Class(b_{IR_i})_j \rightarrow i{:}b_{IR_i,j};\ mit\ i{:}b_{IR_i,j}\ \ als\ \ owl{:}Class \tag{A.III.10}$$

Im Falle von zwei Begriffen $b_{IR_i,n}$ und $b_{IR_i,m}$, welche als UML-Klassen $Class(b_{IR_i,n})$ und $Class(b_{IR_i,m})$ modelliert wurden und einer vorliegenden Generalisierungsbeziehung als Modellelement j zwischen $Class(b_{IR_i,n})$ und $Class(b_{IR_i,m})$, resultiert dies in einer Subklassenbeziehung entsprechend dem RDFS-Modellement `rdfs:SubClassOf`. Hierbei werden Quellbegriff und Zielbegriff übernommen. Dies ist in in Ausdruck III.11 dargestellt. Für das Beispiel in Abbildung III.10 würde somit eine RDFS-Subklassenbeziehung zwischen Produkt und Halbzeug modelliert (`rdfs:SubClassOf(Halbzeug, Produkt)`).

$$\begin{aligned} &Generalization(Class(b_{IR_i,n}), Class(b_{IR_i,m}))_j \rightarrow \\ &rdfs{:}SubClassOf(Class(b_{IR_i,n}), Class(b_{IR_i,m})) \end{aligned} \tag{A.III.11}$$

Attribute werden im Sinne der UML-Klassendiagramme einer Klasse zugewiesen, wobei auch ein Datentyp vergeben wird. Attribute können dann mehrfach unterschiedlichen Klassen zugewiesen werden, es ist jedoch keine geschlossene Definition der Verwendbarkeit eines UML-Attributs vorgesehen [Obj17@]. Dies ist ein Unterschied zur Definition von `owl:DatatypeProperties`, da für letztere eine geschlossene Definition der Verwendbarkeit mit Hilfe des `rdfs:domain` und `rdfs:range` möglich ist [W3C12@a]. Die `rdfs:domain` verweist in diesem Falle auf Klassen, welche mit diesem `owl:DatatypeProperty` verbunden werden, wohingegen die `rdfs:range` den Datentyp des `owl:DatatypeProperties` definiert. Um diesen Unterschied logisch zu überbrücken, wird folgende Beziehung in Ausdruck III.12 eingeführt. Demnach wird jedes Attribut *Attribute* als Modellelement j, das einer UML-Klasse $Class(b_{IR_i,n})$ zugewiesen ist und den Datentyp $Datentyp_m$ trägt, zu einem

`owl:DatatypeProperty`. Dabei gilt, dass die *Domain* für alle UML-Klassen $Class(b_{IR_i,n})$ im Sinne eines logischen „ODER“ definiert wird. Da bei einem `owl:DatatypeProperty` nur eine `rdfs:range` definiert werden kann, muss dies bei der Lightweight-Modellierung berücksichtigt werden, indem stets derselbe Datentyp für $Datentyp_m$ verwendet wird. Für das Attribut „Länge“ im Beispiel der Abbildung III.10 würde somit ein `owl:DatatypeProperty` modelliert werden, welches als `rdfs:domain` die Klassen „Halbzeug“ oder „Maschine“ haben kann und als Datentyp eine Gleitkommazahl („float“) besitzt. Auch hier wird der Namensraum der Informationsressource übernommen, was im Beispiel in einem `owl:DatatypeProperty` mit dem IRI `VDI1234:Länge` resultiert.

$$\begin{gathered} Attribute_j(Class(b_{IR_i,n}), Datentyp_m) \rightarrow i\colon Attribute_j(Domain, Range); \\ mit\ i\colon Attribute_j(Domain, Range)\ als\ owl{:}DatatypeProperty \\ Domain = \bigvee_{n=0}^{N} Class(b_{IR_i,n}) \\ Range = Datentyp_1 = Datentyp_2 = Datentyp_m \end{gathered} \tag{A.III.12}$$

Der gleiche Unterschied besteht in der Semantik einer UML-Assoziation und einem `owl:ObjectProperty`. Für eine UML-Assoziation wird stets eine Quelle (englisch: „source“) und ein Ziel (englisch: „target“) definiert, allerdings ist dies keine geschlossene Definition [Obj17@]. Im Rahmen der Definition eines `owl:ObjectProperty` hingegen wird eine `rdfs:domain` für die möglichen „Quellen“ und ein `rdfs:range` für die möglichen „Ziele“ dieses `owl:ObjectProperty` definiert [W3C12@a]. Um diesen Unterschied logisch zu überbrücken, wird folgende Beziehung in Ausdruck III.13 eingeführt. Liegt eine Assoziation *Association* als Modellelement j vor, welche zwischen den Klassen $Class(b_{IR_i,n})$ und $Class(b_{IR_i,m})$ angelegt ist, so wird diese in ein `owl:ObjectProperty` überführt. Die *Domain* des `owl:ObjectProperty` wird dabei als logisches „ODER“ aller Klassen $Class(b_{IR_i,n})$ modelliert, welche als Quelle der Assoziation auftreten. Gleiches gilt für alle Klassen $Class(b_{IR_i,m})$ die als Ziel der Assoziation auftreten. In Bezug auf das Beispiel in Abbildung III.10 würde die Assoziation „nutzt“ zu einem `owl:ObjectProperty` werden, dessen `rdfs:domain` die Klasse „Prozess“ darstellt und dessen `rdfs:range` die Klasse „Technische Ressource“ darstellt. Die Nutzung des Namensraums der Informationsressource erfolgt hier analog zu vorherigen Beschreibungen, was im Beispiel in einem `owl:ObjectProperty` mit dem IRI `VDI1234:nutzt` resultiert.

$$\begin{gathered} Association_j(Class(b_{IR_i,n}), Class(b_{IR_i,m})) \rightarrow i\colon Association_j(Domain, Range); \\ mit\ i\colon Association_j(Domain, Range)\ als\ owl{:}ObjectProperty \\ Domain = \bigvee_{n=0}^{N} Class(b_{IR_i,n}) \\ Range = \bigvee_{m=0}^{M} Class(b_{IR_i,m}) \end{gathered} \tag{A.III.13}$$

Sollte für zwei unterschiedliche Klassen semantische Äquivalenz bestehen, zwei unterschiedliche Begriffe also die exakt gleiche Bedeutung in der Informationsressource aufweisen, dann kann dies mit dem Stereotypen „Equivalence“ angezeigt werden. Dabei kann der Verbinder auch für die Modellierung einer Komplementärbeziehung genutzt werden. Dies bedeutet, dass keine semantische Äquivalenz vorliegt, es sich also um Begriffe mit unterschiedlicher Bedeutung handelt, diese sich aber in einem Nutzungskontext ergänzen. Dabei gilt die gleiche Semantik wie für semantische Äquivalenz in OWL, was in Ausdruck III.14 dargestellt ist. Ist zwischen zwei Klassen $Class(b_{IR_i,n})$ und $Class(b_{IR_i,m})$ eine Äquivalenzbeziehung $Equivalence$ als Modellelement j angelegt, so wird dies bei Vorliegen der Eigenschaft „positive“ in eine `owl:equivalentClass` und bei Vorliegen der Eigenschaft „negative“ in ein `owl:complementOf` überführt. Für das Beispiel in Abbildung III.10 wurde eine orangefarbene Assoziation mit dem Stereotyp „Equivalence“ zwischen den Klassen „Halbzeug“ und „Maschine“ angelegt, was eine Komplementärbeziehung („negative“) darstellen sollte. Diese resultiert in einem (`owl:complementOf(Halbzeug, Maschine)`). Zwischen der Klasse „Maschine“ und „Technische Ressource“ wurde eine blaue Assoziation mit dem Stereotyp „Equivalence“ angelegt, welche eine Äquivalenzbeziehung („positive“) anzeigen sollte. Letztere resultiert in einem (`owl:equivalentClass(Halbzeug, Maschine)`).

$$\begin{gathered} Equivalence(Class(b_{IR_i,n}), Class(b_{IR_i,m}), positive)_j \rightarrow \\ owl:equivalentClass(Class(b_{IR_i,n}), Class(b_{IR_i,m})); \\ Equivalence(Class(b_{IR_i,n}), Class(b_{IR_i,m}), negative)_j \rightarrow \\ owl:complementOf(Class(b_{IR_i,n}), Class(b_{IR_i,m})) \end{gathered} \tag{A.III.14}$$

III-3.2.2.2. UML-basiertes Alignment von Lightweight-Ontologie-Design-Mustern

Im Rahmen der Erstellung der Alignment-Ontologie ist die Erstellung von zwei Diagrammen von Vorteil. Das erste Diagramm stellt die eigentliche Alignment-Ontologie im gleichen Stil wie Abschnitt III-3.2.2.1 dar und ist in jedem Fall zu erstellen. Das zweite Diagramm dokumentiert in Anlehnung an [Alm14] die vorgenommenen Operationen, welche auf den ODMs im Rahmen des Alignments durchgeführt wurden, und ist insbesondere bei einer Vielzahl an ODMs und vorgenommenen Operationen von Vorteil, da auch im Nachinein die Alignment-Strategie erkennbar ist und somit die Wartbarkeit der entwickelnden Ontologie als Ganzes steigt.

Wichtige Operationen, die auf ODMs ausgeführt werden, sind entsprechend [GaPr09] die Komposition, Spezialisierung bzw. Generalisierung, der Import und die Erweiterung. Im Rahmen der Komposition werden ODMs zu einer neuen Ontologie verbunden. Die entstandene Ontologie ist dabei als Summe der Konzepte der einzelnen ODMs zu verstehen, wobei lokal weitere Konzepte (bspw. Klassen, Object- und DatatypeProperties) hinzugefügt werden können, um mehrere ODMs miteinander zu kombinieren. Im Falle einer Spezialisierung wird eine neue Ontologie durch hierarchische Anordnung im Sinne einer Spezialisierung bzw. Generalisierung zwischen Konzepten zweier ODMs realisiert. Hierbei werden `rdfs:subClassOf` Relationen zwischen Klassen, bzw. `owl:subPropertyOf` Relationen zwischen Properties der ODMs angelegt. Der Import ist der simpelste Fall, der Kombination mehrere ODMs und der einzige Fall der im OWL-Vokabular (`owl:import`) vorkommt. Hierbei werden unterschiedli-

che ODMs in eine neue Ontologie importiert, ohne dass weitere Maßnahmen wie Komposition oder Spezialisierung notwendig sind. Der letzte Fall ist die Erweiterung, bei welcher einem ODM weitere Axiome, Klassen oder Properties hinzugefügt werden. [GaPr09]

Übertragen auf die Lightweight-Modellierung mit Hilfe von UML-Klassendiagrammen ergeben sich daher folgende Modellierungsmöglichkeiten, wobei auf die Erläuterung der Modellsemantik in Abschnitt III-3.2.2.1 hingewiesen sei. Zunächst ist ein neues Diagramm zu erstellen, welches die Alignment-Ontologie repräsentiert. In dieses Diagramm sind alle Klassen und Assoziationen der anderen Diagramme zu importieren, was damit dem `owl:import` entspricht. Für den Fall, dass ein einfacher import nicht ausreichend ist, um die an die Ontologie gestellten Anforderungen zu erfüllen, können im Rahmen der Komposition weitere UML-Klassen im Namensraum der Alignment-Ontologie eingeführt werden, sowie weitere UML-Assoziationen. Sollte lediglich ein ODM importiert worden sein, so handelt es sich bei diesem Vorgehen um eine Erweiterung statt einer Komposition. Um eine Spezialisierung durchzuführen, kann die UML-Spezialisierung genutzt werden, was zu einer hierarchischen Anordnung von Klassen zweier ODMs führt.

Für den Fall, dass für die Erstellung der Alignment-Ontologie keine Veränderung der importierten ODMs notwendig ist, handelt es sich um ein Zusammenfügen (englisch: „merging“) der einzelnen ODMs [EuSh13]. Sollten Änderungen an Klassen, Assoiziationen oder Attributen einzelner ODMs notwendig sein, führt dies zu einer Versöhnung (englisch: „reconciliation“) [EuSh13]. Im letzteren Fall ist eine neue Version des entsprechenden ODM zu erstellen, in welcher die Änderungen vorgenommen werden, und das Alignment dann erneut durchzuführen. Für die Darstellung der vorgenommenen Alignment-Operationen kann ein UML-Paketdiagramm erstellt werden, wobei jedes Paket einen ODM repräsentiert bzw. ein Paket die Alignment-Ontologie. Die entsprechenden Elemente des Profils sind in Abbildung III.11 zu finden. Demnach werden vier Modellelemente genutzt: ein Stereotyp „Generalization“ als Erweiterung eines UML-Nestings, ein Stereotyp „Ontology“ als Erweiterung eines UML-Package, ein Stereotyp „owl:import“ als Erweiterung eines UML-PackageImport und ein Stereotyp „Composition“ als Erweiterung eines UML-PackageMerge. Die Stereotypen „Generalization“, „Composition“ und „owl:import“ können mit dem Stereotypen „Ontology“ verbunden werden.

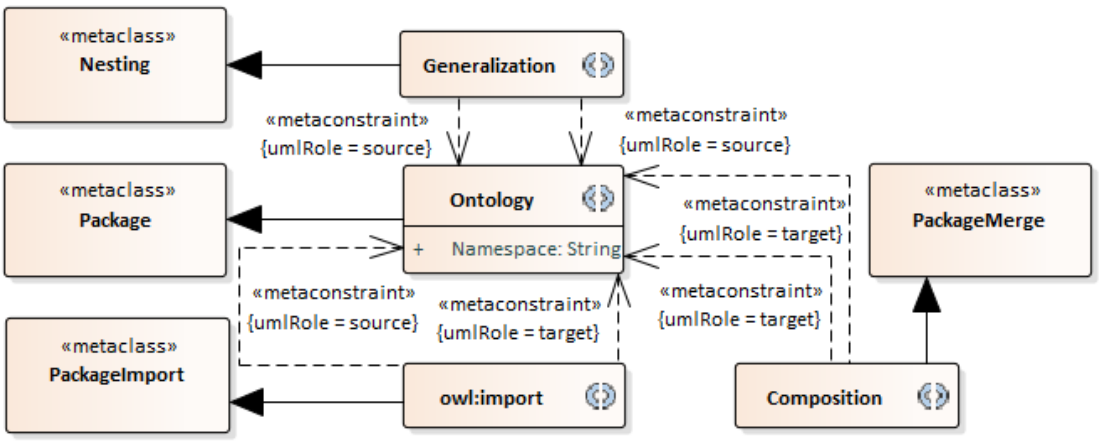

Abbildung III.11.: UML-Profil zur Modellierung der Alignment-Operationen

Ein Beispieldiagramm welches unter Nutzung des Profils in Abbildung III.11 erstellt wurde, ist in Abbildung III.12 dargestellt. Die Abbildung weist vier Ontologien auf, sowie alle als UML-Klasse dargestellten Begriffe und den Namensraum der Ontologien. Die „Alignment-Ontologie“ importiert die drei anderen Ontologien. Darüber hinaus ist eine „Composition“ Relation zwischen den Ontologien „Ontologie-A“ und „Beispiel VDI1234“ angelegt. Dies bedeutet, dass mindestens neue Relationen zwischen den beiden Ontologien angelegt wurden. Zwischen „Ontologie-A“ und „Ontologie-B“ wurde darüber hinaus ein UML-Nesting angelegt, was bedeutet, dass mindestens Teile von „Ontologie-B“ eine Spezialisierung von „Ontologie-A“ darstellen.

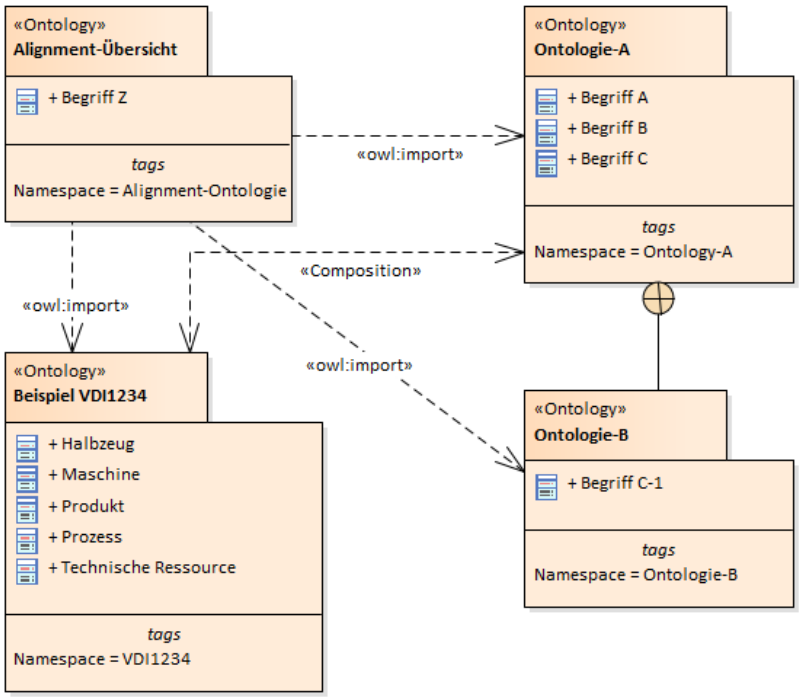

Abbildung III.12.: Beispieldiagramm zur Übersicht über Alignment-Operationen

III-3.2.2.3. Vorab-Verifikation von Lightweight-Ontologien

Nachdem eine Lightweight-Variante eines ODM oder einer Alignment-Ontologie erzeugt wurde, kann bereits eine Vorab-Verifikation durchgeführt werden. Die Vorab-Verifikation wird durch die Stakeholder durchgeführt, welche die vorherigen Modellierungsschritte unternommen haben (Software-Entwickler und Domänenexperten) und dient lediglich der Überprüfung, ob alle notwendigen Begriffe und Relationen in der Lightweight-Variante der ODMs enthalten sind, um mit einer Modellierung der Heavyweight-Variante beginnen zu können. Hierzu wird zunächst die Menge an Begriffen und Relationen B_i bestimmt, welche bisher in Lightweight-Ontologien i modelliert wurden. Dabei steht i für eine einzelne Ontologie, welche wiederum eine einzelne Informationsressource bzw. die Alignment-Ontologie repräsentiert. Für die Bestimmung der Gesamtmenge B werden die Mengen der Begriffe B_i der einzelnen Lightweight-Ontologien i vereint. Dies ist in Ausdruck III.15 dargestellt.

$$B = \bigcup_{i=0}^{I} B_i \tag{A.III.15}$$

Für die Begriffsabdeckung kann die Vollständigkeit δ bestimmt werden, welche ausdrückt, ob alle Begriffe des Anforderungsglossars modelliert worden sind. Die Vollständigkeit δ ist eins wenn dies zutrifft und null wenn nicht. Der Abgleich erfolgt über die vorher bestimmte Gesamtmenge an Begriffen B in modellierten Lightweight-Ontologien und die Menge an Begriffen im Anforderungsglossar B_G und ist in Ausdruck III.16 dargestellt. Sofern die Menge an Begriffen im Anforderungsglossar eine Teilmenge der Begriffe in den Lightweight-Ontologien, also alle Begriffe des Glossars in Lightweight-Ontologien zu finden sind, dann ist die Vollständigkeit δ eins.

$$\delta = 1; \; wenn \; B_G \subseteq B \tag{A.III.16}$$

III-3.3. Ontologie-Erstellung (Heavyweight-TBox)

Im vorherigen Abschnitt wurden Lightweight-Ontologie-Design-Muster erstellt sowie eine Lightweight-Alignment-Ontologie. Das Ziel dieses Abschnitts ist es, die einzelnen ODMs in Heavyweight-Ontologie-Design-Muster zu überführen und in einer Heavyweight-Alignment-Ontologie miteinander zu verbinden und anschließend eine Verifikation und Dokumentation durchzuführen. Abschnitt III-3.3.1 widmet sich der Überführung von Lightweight-Ontologien nach OWL 2 DL (nachfolgend kurz: OWL). Abschnitt III-3.3.2 geht dann auf die Verifikation der erstellten TBox ein. Letztlich stellt III-3.3.3 zu berücksichtigende Aspekte bei der Dissemination, also der Dokumentation und Veröffentlichung, auf.

III-3.3.1. Modellierung der Heavyweight-Ontologie-Design-Muster und der Alignment-Ontologie

Der vorliegende Abschnitt beschreibt das Vorgehen zur Erstellung der Heavyweight-Ontologie-Design-Muster. Hierzu stellt Abschnitt III-3.3.1.1 zunächst das Vorgehen zur Überführung der Lightweight- in eine Heavyweight-Ontologie vor. Abschnitt III-3.3.1.2 geht dann auf die Erstellung der Alignment-Ontologie ein, bevor Abschnitt III-3.3.1.3 assertionale Muster als wesentliches Element zur Steigerung der Wiederverwendbarkeit der entwickelten Artefakte einführt.

Für den gesamten Abschnitt III-3.3.1 sollen nachfolgende Definitionen in Ausdruck III.17 gelten. Eine Informationsressource i definiert einen gleichnamigen oder abgeleiteten, eindeutigen Namensraum i. Die einzelnen Namensräume i sind Teil der Menge aller Namensräume I in einem betrachteten Projekt. Die Alignment-Ontologie erhält ebenfalls einen Namensraum $\jmath$, welcher Teil der Menge aller Namensräume I ist. Die Anzahl der nach OWL zu überführenden UML-Modellelemente sei L, wobei ein einzelnes UML-Modellelement als l

gekennzeichnet ist. Die Überführung von UML-Modellementen nach OWL wird durch eine Bedingung c dargestellt, welche Teil der Menge aller Bedingungen C für die Überführung ist. Trifft eine Bedingung c auf ein UML-Modellelement zu, dann resultiert dies in einer Anzahl n von Tripeln $t_{i,l,n}$ für ein UML-Modellelement j des Namensraums i. Die Tripel $t_{i,l,n}$ sind Bestandteil der Menge der Tripel der TBox T_{TBox_i} eines Namensraums i.

$$\begin{aligned} &\text{Namensräume der Ontologien: } i \in I \\ &\text{Namensraum der Alignment-Ontologie: } \jmath \in I \\ &\text{Laufindex der UML-Modellelemente: } l \dots L \\ &\text{Bedingungen für die Überführung von UML in OWL: } c \in C \\ &\text{Tripel für ein UML-Modellelement der TBox: } t_{i,l,n} \in T_{TBox_i} \end{aligned} \tag{A.III.17}$$

III-3.3.1.1. Definition der Ontologie-Design-Muster

Im ersten Schritt wird ein eindeutiger Namensraum i für jedes ODM und die Alignment-Ontologie erstellt. Dies wird mit Hilfe eines „Internationalized Resource Identifiers“ (IRI) vorgenommen [W3C12@a]. Danach werden die UML-Modellelemente der Lightweight-Ontologien nach OWL 2 DL (nachfolgend kurz: OWL) überführt. Hierbei wird auf die Verknüpfung der Modellsemantik der Lightweight- und Heavyweight-Ontologien in Abschnitt III-3.2.2 zurückgegriffen. Für die nachfolgend beschriebenen Ausdrücke wird stets das Beispiel in Abbildung III.10 herangezogen um eine beispielhafte Anwendung der Ausdrücke aufzuzeigen. Zur Software-technischen Unterstützung kann beispielsweise das Werkzeug Protégé [Sta19@] genutzt werden.

Zunächst können die UML-Klassen nach OWL überführt werden, wobei vorerst keine Hierarchisierung vorgenommen wird. Dabei wird für jede erstellte UML-Klasse $class_{i,l} \in Classes_i$ eine OWL-Klasse erstellt. Ausdruck III.18 stellt dies dar. Die Bedingung definiert, das bei Vorliegen einer UML-Klasse $class_{i,l}$ ein Tripel entsteht. Der Begriff den die Klasse repräsentiert wird als Subjekt verwendet.

$$\begin{aligned} &c\colon class_{i,l} \\ &t_{i,l}\colon i\colon class_{i,l}\ \textit{rdf:type}\ \textit{owl:Class}\ . \end{aligned} \tag{A.III.18}$$

Als Beispiel kann Listing III.5 basierend auf Abbildung III.10 erstellt werden. Demnach sind alle Klassen als OWL-Klassen modelliert. Der Namensraum wurde als Prefix definiert und allen Klassen zugewiesen.

Listing III.5: Beispielhaftes Resultat der Anwendung von Ausdruck III.18

```
PREFIX i: <example://VDI1234#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
```

```
%TBox
i:Produkt rdf:type owl:Class.
i:Halbzeug rdf:type owl:Class.
i:Prozess rdf:type owl:Class.
i:TechnischeRessource rdf:type owl:Class.
i:Maschine rdf:type owl:Class.
```

Aus der Ausführung von Ausdruck III.18 ergibt sich zunächst eine flache Struktur von Klassen. Da im Rahmen der Lightweight-Modellierung auch eine Modellierung von Generalisierungs- bzw. Spezialisierungsbeziehungen möglich ist, kann Ausdruck III.19 herangezogen werden, um eine Hierarchie von OWL-Klassen basierend auf UML-Generalisierungsbeziehungen einzuführen. Die Bedingung besagt, dass eine UML-Generalisierungsbeziehungen zwischen zwei UML-Klassen $class_{i,o}$ und $class_{i,p}$ in einem Tripel resultiert, wobei o und p nicht gleich sein sollen. Der Tripel weist dann $class_{i,o}$ als Subjekt, `rdfs:SubClassOf` als Prädikat und $class_{i,p}$ als Objekt auf.

$$\begin{aligned} c\colon & \mathit{Generalization}(class_{i,o}, class_{i,p})_{i,l};\ \mathit{mit}\ o \neq p \\ t_{i,l}\colon & i\colon class_{i,o}\ \mathit{rdfs{:}subClassOf}\ i\colon class_{i,p}. \end{aligned} \tag{A.III.19}$$

Als Beispiel für eine Anwendung der Logik in Ausdruck III.19 kann Listing III.6 erstellt werden. Da im Beispiel der Abbildung III.10 lediglich eine Generalisierungsbeziehung modelliert wurde, wird auch nur ein Tripel erstellt.

Listing III.6: Beispielhaftes Resultat der Anwendung von Ausdruck III.19

```
PREFIX i: <example://VDI1234#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

%TBox
i:Halbzeug rdfs:subClassOf i:Produkt.
```

Des Weiteren besteht die Möglichkeit, dass im Rahmen der Lightweight-Modellierung eine Äquivalenz („positive“ oder „negative“) zwischen zwei Klassen definiert wurde. Dabei können zwei Fälle auftreten, je nach Nutzung der Logik-Variable, welche im Beispiel in Abbildung III.10 als farbliche Markierung angezeigt wurde. Der positive Fall wird in Ausdruck III.20 dargestellt. Demnach wird für ein UML-Modellelement l, welches eine Äquivalenzbeziehung mit positiver Logik zwischen den Klassen $class_{i,o}$ und $class_{i,p}$ darstellt, ein Tripel erstellt. Dieser Tripel definiert die Klasse $class_{i,o}$ als Subjekt, `owl:equivalentClass` als Prädikat und die Klasse $class_{i,p}$ als Objekt des Tripels.

$$\begin{aligned} c\colon & \mathit{Equivalence}(class_{i,o}, class_{i,p}, positive)_{i,l};\ \mathit{mit}\ o \neq p \\ t_{i,l}\colon & i\colon class_{i,o}\ \mathit{owl{:}equivalentClass}\ i\colon class_{i,p}. \end{aligned} \tag{A.III.20}$$

Den Fall einer negativen Logik, also einer Komplementärbeziehung zwischen zwei Klassen $class_{i,o}$ und $class_{i,p}$ stellt Ausdruck III.21 dar. Demnach resultiert ein Tripel aus einem UML-Modellelement l, welches eine Äquivalenzbeziehung mit negativer Logik zwischen den Klassen $class_{i,o}$ und $class_{i,p}$ darstellt. Subjekt des Tripels ist die $class_{i,o}$, Prädikat `owl:complementOf` und Objekt die Klasse $class_{i,p}$. Als Beispiel für eine Anwendung der Logik in Ausdruck III.20 und III.21 kann analog zu bisherigen Beispielen Listing III.7 erstellt werden.

$$\begin{aligned} c\colon & \mathit{Equivalence}(class_{i,o}, class_{i,p}, negative)_{i,l};\ mit\ o \neq p \\ t_{i,l}\colon & i\colon class_{i,o}\ \mathit{owl{:}complementOf}\ i\colon class_{i,p}. \end{aligned} \tag{A.III.21}$$

Listing III.7: Beispielhaftes Resultat der Anwendung von Ausdruck III.20 und III.21

```
PREFIX i: <example://VDI1234#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

%TBox
i:Maschine owl:equivalentClass i:TechnischeRessource.
i:Halbzeug owl:complementOf i:Maschine.
```

Für UML-Assoziationen gilt, dass sie in OWL-ObjectProperties überführt werden können. Hierbei muss unterschieden werden, ob die UML-Assoziationen mehrere UML-Klassen als Quelle oder Ziel aufweist, also mehrfach verwendet wurde. Im einfachen Fall wurde eine UML-Assoziationen l lediglich einmal zwischen einer Klasse *source* und einer Klasse *target* verwendet, darauf geht Ausdruck III.22 ein. Der Betrag der Mengen der Quellen *Sources* und der Menge der Ziele $Targets$ ist dann also gleich eins. In diesem Falle werden drei Tripel definiert. Der erste Tripel definiert die Assoziation $association_{i,l}$ als Subjekt, `rdf:type` als Prädikat und `owl:ObjectProperty` als Objekt. Die beiden anderen Tripel definieren die Assoziation $association_{i,l}$ ebenfalls als Subjekt, jedoch `rdfs:domain` bzw. `rdfs:range` als Prädikat und die Quelle der UML-Assoziation *source* bzw. das Ziel der UML-Assoziation *target* als Objekt. Als Beispiel für eine Anwendung der Logik in Ausdruck III.22 kann Listing III.8 erstellt werden.

$$\begin{aligned} c_1\colon & \mathit{association}(source, target)_{i,l} \\ c_2\colon & source \in Sources; |Sources| = 1 \\ c_3\colon & target \in Targets; |Targets| = 1 \\ t_{i,l,1}\colon & i\colon association_{i,l}\ \mathit{rdf{:}type\ owl{:}ObjectProperty}. \\ t_{i,l,2}\colon & i\colon association_{i,l}\ \mathit{rdfs{:}domain\ i{:}source}. \\ t_{i,l,3}\colon & i\colon association_{i,l}\ \mathit{rdfs{:}range\ i{:}target}. \end{aligned} \tag{A.III.22}$$

Listing III.8: Beispielhaftes Resultat der Anwendung von Ausdruck III.22

```
PREFIX i: <example://VDI1234#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

%TBox
i:nutzt rdf:type owl:ObjectProperty.
i:nutzt rdfs:domain i:Prozess.
i:nutzt rdfs:range i:TechnischeRessource.
i:istInputFuer rdf:type owl:ObjectProperty.
i:istInputFuer rdfs:domain i:Produkt.
i:istInputFuer rdfs:range i:Prozess.
```

Im Falle einer mehrfachen Verwendung der UML-Assoziation müssen die mehrfach auftauchenden Quellen bzw. Ziele als logisches „ODER“ in OWL modelliert werden. Ausdruck III.23 geht darauf ein. Liegt auch hier eine UML-Assoziationen l vor, weißt jedoch der Betrag der Menge der verwendeten Quellen $Sources$ und Ziele $Targets$ einen Wert größer eins auf, so entstehen sieben Tripel. Der erste Tripel definiert analog zu Ausdruck III.22 das `owl:ObjectProperty`. Der zweite Tripel $t_{i,l,2}$ definiert als Subjekt Assoziation $association_{i,l}$, als Prädikat `rdfs:domain` und als Objekt einen BlankNode. Dieser stellt einen leeren Knoten dar, welcher lediglich für die Verknüpfung weiterer Tripel dient (siehe [BrGu14[@]]). Tripel $t_{i,l,3}$ definiert den BlankNode als OWL-Klasse. Tripel $t_{i,l,4}$ führt dann die Verknüpfung als logisches „ODER“ mit Hilfe des `owl:unionOf` ein. Subjekt ist hier erneut der BlankNode, Prädikat ist `owl:unionOf` und Objekt ist die Menge von Klassen $i : Sources$. Die Tripel fünf bis sieben stellen dies analog für die Range des `owl:ObjectProperty` dar. In Ausdruck III.23 wurde lediglich der Fall betrachtet, dass sowohl Quellen als auch Ziele der UML-Assoziationen l mehrfach auftreten. Treten nur Quellen oder Ziele mehrfach auf, ist eine Kombination von Ausdruck III.22 und III.23 vorzunehmen.

$$
\begin{aligned}
c_1 &: association(source, target)_{i,l} \\
c_2 &: source \in Sources; |Sources| > 1 \\
c_3 &: target \in Targets; |Targets| > 1 \\
t_{i,l,1} &: i\,{:}\,association_{i,l}\ \textit{rdf:type owl:ObjectProperty.} \\
t_{i,l,2} &: i\,{:}\,association_{i,l}\ \textit{rdfs:domain _:BlankNodeSource}_l. \\
t_{i,l,3} &: \textit{_:BlankNodeSource}_l\ \textit{rdf:type owl:Class.} \\
t_{i,l,4} &: \textit{_:BlankNodeSource}_l\ \textit{owl:unionOf (i:Sources).} \\
t_{i,l,5} &: i\,{:}\,association_{i,l}\ \textit{rdfs:range _:BlankNodeTargets}_l. \\
t_{i,l,6} &: \textit{_:BlankNodeTargets}_l\ \textit{rdf:type owl:Class.} \\
t_{i,l,7} &: \textit{_:BlankNodeTargets}_l\ \textit{owl:unionOf (i:Targets).}
\end{aligned}
\tag{A.III.23}
$$

Als Beispiel für eine Anwendung der Logik in Ausdruck III.23 kann Listing III.9 erstellt

werden. Da das Beispiel in Abbildung III.10 jedoch keine Mehrfachverwendung von UML-Assoziationen darstellt, sei angenommen, dass die UML-Assoziation „nutzt“ auch zwischen den Klassen „Maschine“ und „Prozess“ angelegt wurde. Es seien somit mehrere Ziele *target* der UML-Assoziation modelliert worden.

Listing III.9: Beispielhaftes Resultat der Anwendung von Ausdruck III.23

```
PREFIX i: <example://VDI1234#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

%TBox
i:nutzt rdf:type owl:ObjectProperty.
i:nutzt rdfs:domain i:Prozess.
i:nutzt rdfs:range _:BlankNodeTarget.
_:BlankNodeTarget rdf:type owl:Class.
_:BlankNodeTarget owl:unionOf (i:TechnischeRessource i:Maschine).
```

Für die in der Lightweight-Ontologie definierten UML-Attribute gilt, dass sie die OWL-DatatypeProperties zugeordnet werden können. Analog zur Überführung von UML-Assoziationen können hier zwei Fälle auftreten. Im einfachen Fall wurde ein UML-Attribut l lediglich einer einzigen Klasse zugewiesen. Darauf geht Ausdruck III.24 ein. Ist ein UML-Attribute l exakt einer Quelle *source* zugewiesen und weist den Datentyp *target* auf, so resultiert dies in drei Tripeln. Der erste Tripel definiert das UML-Attribut $attribute_{i,l}$ als Subjekt, `rdf:type` als Prädikat und `owl:DatatypeProperty` als Objekt. Tripel zwei und drei definieren analog zum Ausdruck III.22 die `rdfs:domain` bzw. die `rdfs:range`. Jedoch stellt das Ziel *target* hier einen Datentypen dar und verweist nicht auf eine Klasse, weshalb auch der Namensraum nicht dargestellt ist.

$$
\begin{aligned}
&c_1 \colon attribute(source, target)_{i,l} \\
&c_2 \colon source \in Sources; |Sources| = 1 \\
t_{i,l,1} &\colon i{:}attribute_{i,l}\ \textit{rdf:type owl:DatatypeProperty.} \\
t_{i,l,2} &\colon i{:}attribute_{i,l}\ \textit{rdfs:domain i:source.} \\
t_{i,l,3} &\colon i{:}attribute_{i,l}\ \textit{rdfs:range target.}
\end{aligned}
\tag{A.III.24}
$$

Als Beispiel für eine Anwendung der Logik in Ausdruck III.24 kann Listing III.10 erstellt werden. Da das Beispiel in Abbildung III.10 eine Mehrfachverwendung von UML-Attributen darstellt, sei angenommen, dass das UML-Attribut „Länge“ lediglich der Klasse „Produkt“ zugewiesen sein.

Listing III.10: Beispielhaftes Resultat der Anwendung von Ausdruck III.24

```
PREFIX i: <example://VDI1234#>
```

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

%TBox
i:Laenge rdf:type owl:DataProperty.
i:Laenge rdfs:domain i:Produkt.
i:Laenge rdfs:range xsd:float.
```

Im Falle einer mehrfachen Verwendung eines UML-Attributs müssen die mehrfach auftauchenden Quellen als logisches „ODER“ in OWL modelliert werden. Ausdruck III.25 geht darauf ein. Liegt auch hier ein UML-Attribut l vor, weißt jedoch der Betrag der Menge der verwendeten Quellen *Sources* einen Wert größer eins auf, so entstehen fünf Tripel. Tripel $t_{i,l,1}$ wird analog zum Ausdruck III.24 definiert. Tripel zwei, drei und vier werden analog zu mehrfach auftreten UML-Assoziationen modelliert. Wie in Abschnitt III-3.2.2.1 beschrieben, ist bereits bei der Leightweight-Modellierung darauf zu achten, dass stets der gleiche Datentyp für mehrfach auftretende UML-Attribute genutzt wird. Daher ist Tripel fünf der gleiche wie Tripel drei in Ausdruck III.24. Als Beispiel für eine Anwendung der Logik in Ausdruck III.25 kann Listing III.11 erstellt werden.

$$
\begin{aligned}
& c_1 : attribute(source, target)_{i,l} \\
& c_2 : source \in Sources;\ mit\ |Sources| > 1 \\
t_{i,l,1} &: attribut_{i,l}\ rdf{:}type\ owl{:}DatatypeProperty. \\
t_{i,l,2} &: i{:}attribut_{i,l}\ rdfs{:}domain\ _{:}BlankNodeSource_l. \\
t_{i,l,3} &: _{:}BlankNodeSource_l\ rdf{:}type\ owl{:}Class. \\
t_{i,l,4} &: _{:}BlankNodeSource_l\ owl{:}unionOf\ (i{:}Sources). \\
t_{i,l,5} &: i{:}attribute_{i,l}\ rdfs{:}range\ target.
\end{aligned}
\tag{A.III.25}
$$

Listing III.11: Beispielhaftes Resultat der Anwendung von Ausdruck III.25

```
PREFIX i: <example://VDI1234#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

%TBox
i:Laenge rdf:type owl:DataProperty.
i:Laenge rdfs:domain _:BlankNodeSource.
_:BlankNodeSource rdf:type owl:Class.
_:BlankNodeSource owl:unionOf (i:Produkt i:Maschine).
i:Laenge rdfs:range xsd:float.
```

Da im Rahmen der Lightweight-Modellierung keine Hierarchisierung von ObjectProperties durchgeführt wurde, kann dies erfolgen, nachdem die UML-Assoziationen überführt wurden. Des Weiteren können die ObjectProperties um Charakteristika (z.B. funktional, invers funktional, relfexiv etc.) ergänzt werden, siehe [Hit+12@]. Dies erfolgt im strukturierten Interview, indem der Ontologe jedes Charakteristikum in eine Frage verwandelt, wie beispielhaft in Tabelle III.2 dargestellt.

Tabelle III.2.: Strukturierte Fragen zur Bestimmung der Charakteristika von Properties in Anlehnung an [Hit+12@]

Eigenschaft	Exemplarische Frage	Logikbeispiel
Inverse	Gibt es für $association_{i,l}(a,b)$, eine $association_{i,l+1}(a,b)$ die komplementär, aber mit entgegengesetzter Richtung zu interpretieren ist?	$hatElternteil(a,b) \rightarrow hatKind(b,a)$
Symmetrisch	Gibt es eine $association_{i,l}(a,b)$, für die immer gilt, dass sie auch zwischen Ziel und Quelle ($association_{i,l}(b,a)$) angelegt werden muss?	$hatEhepartner(a,b) \rightarrow hatEhepartner(b,a)$
Asymmetrisch	Gibt es eine $association_{i,l}(a,b)$, für die niemals gilt, dass sie zwischen Ziel und Quelle $association_{i,l}(b,a)$ angelegt werden darf?	$hatKind(a,b) \rightarrow \neg hatKind(b,a)$
Disjunkt	Gibt es eine $association_{i,l}(a,b)$ und eine $association_{i,l+1}(a,b)$ die niemals Ziel und Quelle teilen dürfen?	$hatElternteil(a,b) \rightarrow \neg hatEhepartner(a,b)$
Reflexiv	Gibt es eine $association_{i,l}(a,b)$ für die bei Verwendung gilt, dass sie die Quelle stets mit sich selbst $association_{i,l}(a,a)$ verbindet?	$hatVerwandte(a,b) \rightarrow hatVerwandte(a,a)$
Irreflexiv	Gibt es eine $association_{i,l}(a,b)$ für die bei Verwendung gilt, dass sie die Quelle niemals mit sich selbst $association_{i,l}(a,a)$ verbinden darf?	$hatElternteil(a,b) \rightarrow \neg hatElternteil(a,a)$
Transitiv	Gibt es eine $association_{i,l}(a,b)$ für die bei einer verketteten Verwendung $association_{i,l}(b,c)$ gilt, dass sie auch zwischen der ersten Quelle und letztem Ziel ,i.S.v. $association_{i,l}(a,c)$ angelegt werden soll?	$hatNachkommen(a,b) \wedge hatNachkommen(b,c) \rightarrow hatNachkommen(a,c)$
Funktional	Gibt es eine $association_{i,l}(a,b)$ für die bei Verwendung gilt, dass sie die Quelle mit nur einem Ziel verbinden darf?	$(\forall b)(b \in B, \lvert B\rvert = 1) \wedge (hatEhemann(a,b))$
Invers-Funktional	Gibt es eine $association_{i,l}(a,b)$ für die bei Verwendung gilt, dass sie ihr Ziel mit nur einer Quelle verbunden werden darf?	$(\forall a)(a \in A, \lvert A\rvert = 1) \wedge (hatEhemann(a,b))$
Property-Chain	Gibt es eine $association_{i,l}(a,b)$ für die bei weiterer Verwendung in der Gestalt $association_{i,l}(b,c)$ gilt, dass eine neue Assoziation $association_{i,l+1}(a,c)$ angelegt werden darf?	$hatKind(a,b) \wedge hatKind(b,c) \rightarrow hatEnkelkind(a,c)$

Auch für UML-Attribute wurde im Rahmen der UML-Modellierung keine Hierarchisierung vorgenommen. Dies kann an dieser Stelle analog zu ObjectProperties erfolgen, ebenso die Definition von Charakteristika, siehe [Hit+12@], wobei lediglich Funktionalität und Disjunktheit von Relevanz sind. Des Weiteren sind die jeweiligen begrifflichen Definition der einzelnen UML-Modellelemente als `rdfs:comment` zu übernehmen, sodass auch einen menschenlesbare Definition der Bedeutung der OWL-Modellelemente vorliegt.

III-3.3.1.2. Definition der Alignment-Ontologie

Im Rahmen der Definition der Heavyweight-Alignment-Ontologie werden die vorher definierten Mengen der Tripel einzelner Heavyweight-ODMs in einer Ontologie vereint. Zur Steigerung der Wartbarkeit und Kompatibilität behalten die einzeln definierten ODMs ihre eigenen Namensräume und auch der Alignment-Ontologie wird ein eigener Namensraum $\jmath$ zugewiesen. Dies führt dazu, dass die einzelnen ODMs wie Module aufzufassen sind, welche lediglich an ihren Schnittstellen (der Alignment-Ontologie) aufeinander abzustimmen sind. Der einfachste Fall, wie in Abschnitt III-3.2.2.2 dargestellt, ist dabei der `owl:import`. Dies ist in Ausdruck III.26 dargestellt. Demnach resultiert die Menge der Tripel der TBox T_{TBox} aus der Vereinigung der einzelnen Mengen von Tripeln T_{TBox_i} der Informationsressourcen i. Diese wiederum resultieren aus der Vereinigung aller einzelnen Tripel $t_{i,l,n}$ über alle Informationsressourcen i und alle UML-Modellelemente l sowie etwaig mehrfache Tripel n pro UML-Modellelement l. Die Herleitung der Tripel $t_{i,l,n}$ wurde in Abschnitt III-3.3.1.1 beschrieben.

$$T_{TBox} = \bigcup_{i=0}^{I} T_{TBox_i} = \bigcup_{i=0}^{I} \bigcup_{l=0}^{L} \bigcup_{n=0}^{N} t_{i,l,n} \tag{A.III.26}$$

Für den Fall, dass eine Komposition, Generalisierung/Spezialisierung oder Erweiterung durchgeführt werden soll, gilt das Vorgehen analog zu Abschnitt III-3.3.1.1. Die sich dabei ergebende Menge der Tripel T_{TBox} ist in Ausdruck III.27 dargestellt. Demnach ist die Menge der Tripel der TBox aus Ausdruck III.26 zu vereinigen mit der Menge der Tripel der Alignment-Ontologie $T_{TBox_\jmath}$.

$$T_{TBox} = \bigcup_{i=0}^{I} T_{TBox_i} \cup T_{TBox_\jmath} \tag{A.III.27}$$

III-3.3.1.3. Definition assertionaler Muster

Wie in Abschnitt III-3.1.1 beschrieben, ist ein Charakteristikum eines ODM das Vorhandensein eines assertionalen Musters. Letzteres wird nachfolgend hergeleitet. Wie in Formel III.1 beschrieben, besteht eine Ontologie aus einer Menge an Tripeln T. Ein Tripel besteht aus drei Entitäten e, welche entweder Subjekt, Prädikat oder Objekt darstellen. Die Menge der Tripel einer TBox T_{TBox} setzt sich aus Entitäten zusammen, welche aus Begriffen und Relationen einer Informationsressource hergeleitet sind, sowie Entitäten der W3C Standards für RDF, RDFS und OWL. Die Menge der Tripel der ABox T_{ABox} hingegen setzt sich aus einer Menge an faktenbezogenen Entitäten zusammen, sowie aus der Menge der Entitäten, welche in der TBox definiert sind. Dabei ist ein einzelner Tripel der ABox dadurch definiert, dass entweder Subjekt, Prädikat, oder Objekt eine faktenbezogene Entität darstellen. Des Weiteren ist die ABox von der Zeit x abhängig, das heißt, dass die Fakten der ABox nur für einen bestimmten Zeitpunkt oder Zeitraum definiert werden. Es ist jedoch möglich, die von der Zeit abhängigen Bestandteile der ABox zu bestimmen und als Variablen zu deklarieren. Auf diese Weise ergibt sich eine wiederkehrende Struktur der ABox mit variablen

Bestandteilen. Dies soll in Ausdruck III.28 zusammengefasst werden. Die Menge der Tripel T_i einer Ontologie i ist definiert als die Vereinigung der Tripelmengen von TBox T_{TBox_i} und ABox $T_{ABox_i}(x)$. Wie oben beschrieben, ist die Menge der Tripel der ABox von der Zeit x abhängig. Die Menge der Tripel der ABox $T_{ABox_i}(x)$ sei definiert durch eine Menge von assertionalen Mustern $\Lambda_i(e(x))$ die ebenfalls spezifisch für die Ontologie i definiert werden. Ein einzelnen assertionales Muster $\lambda_{i,h}(e(x))$ mit dem Laufindex h ist Teil der Menge aller assertionaler Muster $\Lambda_i(e(x))$ einer Ontologie i und besteht selbst Tripeln $t_{i,h,j}(x)$. Ein einzelner Tripel $t_{i,h,j}(x)$ mit dem Laufindex j bis J eines assertionalen Musters $\lambda_{i,h}(e(x))$ besitzt drei Entitäten ($e_{i,j,1}$, ($e_{i,j,2}$ und ($e_{i,j,3}$ wobei mindestens eine der drei Entitäten von der Zeit x abhängig ist. Letztere wird als variable Entität $e(x)$ bezeichnet.

$$
\begin{gathered}
T_i = T_{TBox_i} \cup T_{ABox_i}(x) \\
T_{ABox_i}(x) = \Lambda_i(e(x)) = \bigcup_{h=0}^{H} \lambda_{i,h}(e(x)) = \bigcup_{h=0}^{H} \bigcup_{j=0}^{J} t_{i,h,j}(x) \\
t_{i,h,j}(x);\ mit\ (e_{i,j,1} \vee e_{i,j,2} \vee e_{i,j,3} = e(x))
\end{gathered}
\tag{A.III.28}
$$

Ein ODM enthält nun mindestens ein assertionales Muster $\lambda_{i,h}(e(x))$, welches die Struktur der ABox dahingehend charakterisiert, dass die intendierte Nutzung der TBox-Entitäten für den Anwendungsfall der Informationsressource ersichtlich wird. Dabei wird das assertionale Muster nicht auf einen bestimmen Fall bezogen, sondern zeigt lediglich auf, wie der Fallbezug hergestellt werden kann, indem die variablen Entitäten $e(x)$ separat deklariert werden. Ein assertionales Muster lässt sich mit Hilfe von SPARQL-Abfragen bzw. SPARQL-Updates darstellen. Dies hat den Vorteil, dass die erstellten assertionalen Muster auch im weiteren Verlauf der Entwicklungsmethodik produktiv eingesetzt werden können, sei es zur Entwicklung der ABox oder zum Entwurf von Abfragen. Listing III.12 stellt die Struktur dar, wie ein entsprechendes assertionales Muster als SPARQL-Update implementiert werden kann. Dabei werden die Tripel der ABox des assertionalen Musters in der INSERT-Klausel aufgelistet. Die variablen Entitäten hingegen, welche in der INSERT-Klausel genutzt werden, werden in der WHERE-Klausel deklariert. Demnach beinhaltet ein SPARQL-Update in der INSERT-Klausel alle Tripel (j bis J) $t_{i,h,j}(x)$ eines assertionalen Musters $\lambda_{i,h}(e(x))$. Hierbei kann für jeden Tripel gelten, dass eine der drei Entitäten eine variable Entität $e(x)$ ist. In der WHERE-Klausel hingegen müssen die variablen Entitäten $e(x)$, welche in der INSERT-Klasuel genutzt wurden, vor Ausführung des SPARQL-Update gebunden werden.

Listing III.12: SPARQL-INSERT-WHERE Struktur für die Erstellung assertionaler Muster

```
PREFIX i: <http://...#>
      INSERT {

            t_{i,h,j}(x) ∈ λ_{i,h}(e(x)), mit (e_{i,j,1} ∨ e_{i,j,2} ∨ e_{i,j,3} = e(x)) für j...J

      } WHERE {

            e(x)_{i,h,j,m} ∈ t_{i,h,j}(x) für j...J und m...M

}
```

Auf diese Weise beinhaltet die INSERT-Klausel eine wiederkehrende Struktur mit Variablen, welche in der WHERE-Klausel deklariert werden. Dies kann unter Nutzung des Beispiels aus Abbildung III.10 verbildlicht werden. Angenommen, der dem Beispiel zugrunde liegende Standard nutzt lediglich die Klassen „Produkt“, „Prozess“, „Technische Ressource“ sowie die Relationen „nutzt“ und „ist Input für“ als eine wiederkehrende Beschreibungen von Fällen, so kann Listing III.13 als ein assertionales Muster für diese Ontologie definiert werden. Dabei wird auf die in Abschnitt III-3.3.1.1 definierte TBox zurückgegriffen.

Listing III.13: SPARQL-INSERT-WHERE Struktur für die Erstellung eines Beispielmusters

```
PREFIX i: <example://VDI1234#>
    INSERT {

    ?Produkt rdf:type i:Produkt.
    ?Prozess rdf:type i:Prozess.
    ?TechnischeRessource rdf:type i:TechnischeRessource.

    ?Produkt i:istInputFuer ?Prozess.
    ?Prozess i:nutzt ?TechnischeRessource.

    } WHERE {

    BIND(IRI(STR("http://ExampleProdukt")) AS ?Produkt).
    BIND(IRI(STR("http://ExampleProzess")) AS ?Prozess).
    BIND(IRI(STR("http://ExampleRessource")) AS ?TechnischeRessource).

}
```

Die gleiche Struktur ist überführbar in eine SPARQL-Abfrage, wobei hier nicht die ABox zum Zeitpunkt x modelliert, sondern abfragt wird. Listing III.14 stellt die Struktur einer derartigen Abfrage dar. Dabei ist erkennbar, dass der Teil der INSERT-Klausel aus Listing III.12 hier unverändert in der WHERE-Klausel auftaucht. Als Beispiel hierfür kann das gleiche Beispiel wie in Listing III.13 herangezogen werden. Wie in den Listings ersichtlich, wird eine Abfrage mit dem Sternoperator (*) durchgeführt, daher werden alle als Variablen deklarierten Entitäten in der Rückgabe berücksichtigt.

Listing III.14: SPARQL-SELECT-WHERE Struktur für die Abfrage eines assertionalen Muster

```
PREFIX i: <http://...#>
    SELECT *
    WHERE {
     t_{i,h,j}(x) ∈ λ_{i,h}(e(x)), mit (e_{0,i,j} ∨ e_{1,i,j} ∨ e_{2,i,j} = e(x)) für j...J
}
```

Listing III.15: SPARQL-SELECT-WHERE Struktur für die Abfrage des Beispielmusters

```
PREFIX i: <example://VDI1234#>
    SELECT *
    WHERE {

        ?Produkt rdf:type i:Produkt.
        ?Prozess rdf:type i:Prozess.
        ?TechnischeRessource rdf:type i:TechnischeRessource.

        ?Produkt i:istInputFuer ?Prozess.
        ?Prozess i:nutzt ?TechnischeRessource.

}
```

Insbesondere bei einer zunehmenden Kardinalität der Menge der Tripel einer ABox erleichtern die assertionalen Muster die Wiederverwendung eines ODM. Einerseits, da die Erstellung von Tripeln der ABox durch Wiederverwendung der SPARQL-Updates erleichtert wird. Andererseits, da die Erstellung von Abfragen durch Bereitstellung von Vorlagen erleichtert ist. Lediglich bei Abwandlung der assertionalen Muster im Rahmen der ABox-Erstellung ist davon auszugehen, dass die Abfrage-Muster nicht direkt wiederverwendet werden können.

Die Menge der assertionalen Muster eines ODM ist stark von der zugrundeliegenden Informationsressource abhängig. So ist es einerseits möglich, dass lediglich ein Muster vorliegt, mit dem sich eine Vielzahl von Fällen beschreiben lässt. Andererseits können Minimal- und Maximalmuster oder Varianten auftreten. Im Rahmen der TBox-Entwicklung müssen aufgrund der iterativen Struktur des Entwicklungsprozesses nicht alle denkbaren assertionalen Muster auf einmal modelliert werden. Vielmehr sollten die im Rahmen des Projektes relevanten assertionalen Muster erarbeitet und dokumentiert werden, sodass die Nutzbarkeit eines ODM für andere Projekte erhöht und somit die Wiederverwendbarkeit gesteigert wird.

Jedes assertionale Muster kann als INSERT und DELETE definiert werden. Hierzu wird für eine existierende SPARQL-Operation das Schlüsselwort von INSERT auf DELETE geändert. Die Logik dieser Operation ist dann äquivalent zur INSERT-Operation, mit Ausnahme, dass der Mustergraph in der DELETE-Klausel entfernt und nicht hinzugefügt wird. Nachdem die assertionalen Muster modelliert sind, können diese testweise ausgeführt werden, um zu verifizieren, dass die erstellten SPARQL-Operationen funktional sind. Zu diesem Zweck wird eine Graph-Datenbank genutzt, beispielsweise [Ont19@], welche die SPARQL-INSERT-Operationen ausführen kann. Zur Überprüfung der korrekten Modellierung der assertionalen Muster werden entsprechende SPARQL-SELECT Abfragen definiert, um das Ergebnis der SPARQL-INSERTs zu überprüfen.

III-3.3.2. Verifikation der erstellten TBox

Im Rahmen der Verifikation wird die Funktionale Eignung der Heavyweight-Ontologien basierend auf [Ham17] überprüft. Hierzu wird auf die Kriterien der Vollständigkeit und Konsistenz entsprechend [Ham17] zurückgegriffen, wobei nachfolgende Erläuterungen sich stets auf die gesamte entwickelte Ontologie beziehen.

Zunächst kann die Überprüfung der Vollständigkeit Δ vorgenommen werden, was nachfolgend anhand von Formel III.29 beschrieben wird. Für die Überprüfung der Vollständigkeit können SPARQL-SELECT Abfragen erstellt werden [NMN09; SPH12]. Hierbei wird für eine Kompetenzfrage r, die im Rahmen der Anforderungserhebung definiert wurde und Teil der Menge aller Kompetenzfragen R ist, mindestens eine Abfrage q_r erstellt. Eine Abfrage q_r ist dann Teil der Menge aller erfolgreichen Abfragen Q_r, wenn die Rückgabe der Abfrage q_r der zur Kompetenzfrage gehörigen Antwort entspricht. Entsprechend Formel III.29 wird dann der Betrag der Menge der erfolgreichen SPARQL-Abfragen $|Q_r|$ durch den Betrag der Menge der Kompetenzfragen $|R|$ geteilt. Das resultierende Verhältnis entspricht der Vollständigkeit Δ.

$$\Delta = \frac{|Q_r|}{|R|},\ mit\ r \in R\ und\ q_r \in Q_r \tag{F.III.29}$$

Für die Erstellung der SPARQL-Abfragen können die in Abschnitt III-3.3.1.3 definierten assertionalen Muster in zwei Schritten genutzt werden. Um die Rückgaben mit den gewünschten Antworten vergleichen zu können, muss zunächst eine Beispiel-ABox erzeugt werden. Dies erfolgt unter Nutzung der assertionalen Muster in Form von SPARQL-INSERTs, wobei ein von Domänenexperten definiertes Beispiel modelliert wird. Anschließend können die assertionalen Muster in Form von SPARQL-SELECT-Abfragen durch Anpassung genutzt werden, um die anforderungserfüllenden SPARQL-SELECT-Abfragen daraus zu komponieren.

Die Überprüfung der Konsistenz $\Im$ der Tripel der TBox T_{TBox_i} erfolgt durch die Nutzung eines Reasoners, welcher automatisiert konsistente Verwendung der in Abschnitt III-1.1 bis III-1.3 dargestellten Semantiken überprüft. Auf diese Weise kann beispielsweise eine inkonsistente Verwendung von Disjunktheit und Äquivalenz überprüft werden. Ein möglicher Reasoner ist beispielsweise Pellet entsprechend [Sir+07]. Eine Übersicht über verfügbare OWL-Reasoner ist in [Mat+15] zu finden. Abschnitt III-1.3 nennt Beispiele für derartige Inkonsistenzen. Abseits von Inkonsistenzen in der Modellierung der Klassenstruktur (z.B. eine zyklische Subklassenstruktur) soll in diesem Schritt auch überprüft werden, inwiefern Inferenzen auch korrekt auf Fakten der ABox ausgeführt werden. Hierzu werden ein positiver und negativer Test entsprechend [Pre+12] herangezogen, wobei die zuvor erstellte Beispiel-ABox die für die Tests notwendigen Fakten T_{ABox^+} liefert. Bei einem positiven Test ist nun zu überprüfen, inwiefern die erwünschten Inferenzen (Subklasssen, transitive Charakteristika etc.) nach dem Start einer Inferenzmaschine in den Fakten erkennbar sind. Bei einem negativen Test sind die Fakten der Beispiel-ABox dahingehend zu kompromittieren, als dass sie vorsätzlich Inkonsistenzen T_{ABox^-} enthalten sollen, welche durch die Inferenzmaschine erkannt werden sollen. Sind alle Konsistenztests erfolgreich bestanden, so ist die erstellte Ontologie als konsistent anzunehmen, was zur Erfüllung der Formel III.30 führt. Demnach

ist das Modellierungsergebnis konsistent, wenn einerseits die Tripelmenge der TBox T_{TBox_i} aller Namensräume i konsistent ist. Für diesen Fall ist der Teil $\Im(T_{TBox_i})$ über alle i gleich eins. Andererseits soll für alle positiven Testfälle $T_{ABox_i^+}$ aller Namensräume i gelten, dass sie ebenfalls konsistent sind. Für diesen Fall ist der Teil $\Im(T_{ABox_i^+})$ über alle i gleich eins. Letztlich soll für alle negativen Testfälle $T_{ABox_i^-}$ aller Namensräume i gelten, dass sie nicht konsistent sind. Für diesen Fall ist der Teil $\neg\Im(T_{ABox_i^+})$ über alle i gleich eins.

$$\bigwedge_{i=0}^{I} \Im(T_{TBox_i}) \wedge \Im(T_{ABox_i^+}) \wedge \neg\Im(T_{ABox_i^-}) = 1 \tag{F.III.30}$$

III-3.3.3. Dissemination der Ontologie-Design-Muster

Wie in Abschnitt III-3.1.2 dargestellt, ist ein wesentlicher Aspekt der Methodik dieser Arbeit die Wiederverwendung von Ontologie-Design-Mustern. Hierzu ist es notwendig, die erstellten ODMs entsprechend zu veröffentlichen, sei es Community-weit oder im Kontext einer Organisation, was über eine reine Ermöglichung des Zugriffs auf die TBox durch Dritte hinausgeht. Im Rahmen der Dokumentation ist einerseits die Struktur der Dokumentation der ODMs von Relevanz [KHH17], wie auch eine kontrollierte Möglichkeit zur Versionierung im Rahmen zukünftiger Entwicklungsiterationen, sodass eine Weiterentwicklung unter Verfügbarkeit vorheriger Versionen möglich ist [Ham17]. Die Dokumentation dient dabei den Stakeholdern eines anderen bzw. neuen Ontologie-Entwicklungsprojektes, die Nutzbarkeit (siehe Abschnitt III-3.2.1.3) eines ODM für die Projektziele und Anforderungen zu bewerten, wobei in Anlehnung an [Ham17] folgende Kriterien für die Bewertung der Nutzbarkeit förderlich sind:

- Erkennbarkeit der Angemessenheit: Grad, zu dem potentielle Nutzer die Eignung eines ODM für ein Projekt erkennen können.
- Lernbarkeit: Grad, zu dem die Struktur und die vorgesehene Nutzung eines ODM durch die Stakeholder erlernt werden kann.
- Bedienbarkeit: Grad, zu dem die Dokumentation die Anwendung und Nutzung des ODM erleichtert.
- Zugänglichkeit: Grad, zu dem die Dokumentation des ODM für Nutzer unterschiedlicher Fähigkeitsstände zugänglich ist.

Aus diesen Aspekten kann unter Nutzung der tabellarischen Dokumentationsstruktur entsprechend [Gan05] und [KHH17] die Tabelle III.3 abgeleitet werden, welche nachfolgend erläutert wird. Zunächst wird eine Beschreibung des „generischen Anwendungsfalls“ entsprechend [Gan05] vorgenommen. Dies umfasst eine textuelle Beschreibung des generellen Problems (bspw. „Beschreibung von Prozessen“) und dessen Rahmenbedingungen (bspw. „Fertigungs- und Verfahrenstechnik“), aus denen der Einsatzzweck eines ODM ersichtlich wird. Der generelle Anwendungsfall wird einmalig definiert und kann i.d.R. aus dem Anwendungsfall der zugrundeliegenden Informationsressource abgeleitet werden. Die Informationsressource, welche explizit genannt wird, stellt die zugrundeliegende Quelle dar, aus denen

die notwendigen Informationen bzw. das Wissen extrahiert wurde, um das ODM zu erstellen (siehe Abschnitt III-3.2.1). Dabei kann es sich um einen Industriestandard, eine wissenschaftliche Publikation oder eine nicht-wissenschaftliche Publikation handeln, welche als weiterführende Dokumentation der Inhalte des ODM dient. Auf diese Weise erlangt die eigentliche Dokumentation des ODM eine minimalistische Struktur, ohne Verlust der Vollständigkeit, was die Lernbarkeit des ODM positiv beeinflusst [Ham17].

Tabelle III.3.: Beschreibung von Ontologie-Design-Mustern

Dokumentationsaspekt	**Dokumentationsform**	**Geförderter Aspekt der Bewertung der Nutzbarkeit**
Generischer Anwendungsfall	Textuelle Spezifikation basierend auf zugrundeliegender Informationsressource.	Erkennbarkeit der Angemessenheit
Informationsressource	Verweis auf die zugrundeliegende Informationsressource	Erkennbarkeit der Angemessenheit, Lernbarkeit
Spezifischer Anwendungsfall	Textuelle Spezifikation basierend auf generellem Anwendungsfall und Bezug zu komplementären ODMs.	Erkennbarkeit der Angemessenheit
Exemplarische Kompetenzfragen und Antworten	Tabellarische Darstellung exemplarischer Kompetenzfragen mittels Archetypen von Kompetenzfragen	Zugänglichkeit (insb. Software-Entwickler), Erkennbarkeit der Angemessenheit, Lernbarkeit
Komplementäre ODMs	Verweis auf komplementäre ODMs unter Bezugsnahme auf spezifische Anwendungsfälle, oder exemplarische Kompetenzfragen, die die Nutzung des komplementären ODMs notwendig machten.	Lernbarkeit
Lightweight Ontologie	Grafische Darstellung der Struktur der TBox des ODM als UML-Klassendiagramm	Zugänglichkeit (Domänenexperte), Lernbarkeit
Terminologisches Muster	Verweis auf Datei in der die TBox des ODM enthalten ist, unter Verweis auf die Wissensrepräsentationsprache des ODM (z.B. OWL DL).	Zugänglichkeit (insb. Ontologe), Lernbarkeit
Assertionale Muster	Darstellung von assertionalen Mustern des ODM in Form von SPARQL-SELECT Abfragen und SPARQL-INSERT Updates	Zugänglichkeit (insb. Ontologe), Bedienbarkeit, Lernbarkeit

Des Weiteren wird eine Beschreibung spezifischer Probleme (bspw. „Beschreibung von Montageprozessen“), sowie deren Rahmenbedingungen (bspw. „in der Automobilfertigung“) vorgenommen, welche in bisherigen Anwendungen auftraten. Der spezifische Anwendungsfall stellt mindestens einen realen Anwendungsfall des ODM dar. Dabei dienen die spezifischen Anwendungsfälle als eine Illustration von Nutzungsbeispielen des ODM in Teilgebieten des generellen Anwendungsfalls, was entsprechend [Ham17] die Lernbarkeit des ODM positiv beeinflusst. Darüber hinaus gilt, dass eine höhere Anzahl spezifischer Anwendungsfälle die Erkennbarkeit der Angemessenheit positiv beeinflusst [Ham17]. Aus den im Rahmen der Anforderungerhebung definierten Kompetenzfragen werden im nächsten Schritt für den ODM als exemplarisch anzusehende Kompetenzfragen und Antworten extrahiert. Diese dienen ebenfalls einer Illustration der Nutzungsmöglichkeiten des ODM [Ham17]. Da sich die

exemplarischen Kompetenzfragen aus spezifischen Anwendungsfällen ergeben, welche über die Anzahl der Anwendungen des ODM zunehmen, ist auch die Menge der exemplarischen Kompetenzfragen nicht abgeschlossen. Ein ODM wird nur in seltenen Fällen allein verwendet und stattdessen mit anderen ODMs für einen spezifischen Anwendungsfall kombiniert. Von daher ist es für die Lernbarkeit von Vorteil, aufzuzeigen, welche komplementären ODMs existieren. Die Lightweight-Ontologie, welche die wesentlichen Begriffe und Relationen der Informationsressource in Form von UML-Klassen, Relationen und Attributen darstellt, wird dazu genutzt, um die Struktur des ODM zu illustrieren. Dies wirkt sich positiv auf die Erkennbarkeit der Angemessenheit und Lernbarkeit aus [Ham17].

Die für einen ODM erstellten assertionalen Muster dienen im Rahmen der Dokumentation der Illustration von detaillierten Nutzungsbeispielen des ODM. Mit deren Hilfe ist es einem Ontologen möglich, in nur wenigen Schritten erste Beispiele in der ABox zu erstellen und auf diese Weise die Nutzbarkeit für einen bestimmten Anwendungsfall zu erkennen. Des Weiteren unterstützt die Verfügbarkeit von assertionalen Mustern die Bedienbarkeit eines ODM, da auch ohne Detailwissen über jede Entität der TBox eine erste Anwendung möglich ist. Da es sich bei der Menge der assertionalen Muster eines ODM nicht um eine abgeschlossene Menge handelt, kann die Menge an assertionalen Mustern im Laufe der Zeit durch Anwendung in unterschiedlichen Projekten anwachsen.

III-4. Wiederverwendbare Ontologie-Design-Muster

In diesem Abschnitt werden wiederverwendbare Ontologie-Design-Muster vorgestellt, welche auf Industriestandards beruhen. Die Auswahl der Standards basiert dabei auf der im Projekt „Semantische Allianz für Industrie 4.0“ getroffenen Auswahl von Industriestandards, welche für die semantische Beschreibung in Industrie 4.0 Anwendungsszenarien von Relevanz sind [Hil+17*a]. Mit Hilfe der auf diesen Industriestandards basierenden Ontologie-Design-Mustern ist eine Modellierung unterschiedlicher Informationsbedarfe in Industrie 4.0 Anwendungsszenarien möglich. Die Abschnitte III-4.1 bis III-4.4 stellen nachfolgend die Ontologie-Design-Muster (ODM) dieser Informationsressourcen vor. Des Weiteren wird ein Alignment der genannten Ontologie-Design-Muster vorgestellt (siehe Abschnitt III-4.5), welches die in [Hil+20*] vorgestellte Alignment-Ontologie verwendet. Für die Vorstellung der einzelnen Ontologie-Design-Muster werden die in Abschnitt III-3.3.3 erläuterten Dokumentationsaspekte herangezogen. Die jeweiligen assertionalen Muster sind aus Platzgründen im Anhang zu finden. Die zu den vorgestellten ODMs gehörenden Heavyweight-Varianten sind online veröffentlicht [HiKö19@a]. Die vorgestellten Kompetenzfragen werden als eine exemplarische Auswahl dargestellt. Darüber hinaus wird auf die Nennung komplementärer ODMs verzichtet, da alle vorgestellten ODMs komplementär zueinander sind. Eine Nennung spezifischer Anwendungsfälle ist in Abschnitt V-2 zu finden, sowie eine detaillierte Beschreibung der spezifischen Anwendungsfälle.

III-4.1. Modellierung von Systemfunktionen basierend auf der VDI 3682

Die Formalisierte Prozessbeschreibung (FPB) entsprechend [VDI/VDE 3682-1] ermöglicht eine standardisierte Beschreibung von technischen und nicht-technischen Prozessen, wobei bei technischen Prozessen sowohl kontinuierliche als auch Batch- und diskrete Prozesse modelliert werden können. Der zweite Teil der Richtlinie [VDI/VDE 3682-2] beinhaltet bereits eine Lightweight-Ontologie, welche für die Erstellung des Lightweight-Ontologie-Design-Musters in Abbildung III.13 herangezogen wurde. Ein Prozess in der FPB beschreibt ein gewünschtes Verhalten eines Systems im Sinne einer lösungsneutralen Beschreibung von In- und Outputs. Inputs (Produkte, Energien und Informationen) werden durch einen Prozessoperator zu Outputs transformiert, wobei dieser Prozess durch eine Technische Ressource durchgeführt wird. Eine Technische Ressource ist eine abstrakte Beschreibung eines Systems, welches sowohl Hardware als auch Software beinhalten kann. Ein Prozess besteht weiterhin aus Prozessoperatoren, Produkten, Energien und Informationen in verschiedenen Zuständen, sowie einer Systemgrenze.

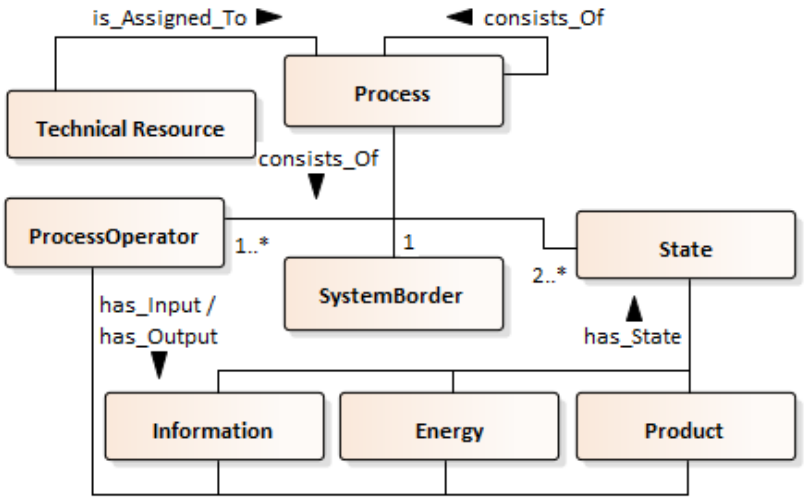

Abbildung III.13.: Lightweight-Ontologie-Design-Muster der [VDI/VDE 3682-1], Abbildung aus [Hil+20*]

Exemplarische Kompetenzfragen, welche mit Hilfe des ODM beantwortet werden können, sind in Tabelle III.4 zu finden. Listing C.1 und Listing C.2 stellen assertionale Muster dieses ODM dar, wobei einerseits ein minimales und ein maximales Muster dargestellt sind.

III-4.2. Modellierung des Systemverhaltens basierend auf der ISA 88

Der technische Report [ANSI/ISA 88] basiert auf den Konzepten der „Packaging Machine Language“ (PackML) und wird zur standardisierten Beschreibung von Maschinenzuständen genutzt. Die Begriffe in diesem Standard beschreiben zwar die Zustände und Transitionen von Verpackungsmaschinen, sind aber grundsätzlich auch auf andere Anwendungsfel-

Tabelle III.4.: Exemplarische Kompetenzfragen für das ODM der VDI 3682

Kompetenzfrage	Antwort
Welche Technischen Ressourcen werden für die Herstellung von Produkt „P“ benötigt?	Liste der Technischen Ressourcen
Welche Prozesse werden für die Herstellung von Produkt „P“ benötigt?	Liste von Prozessen
Welche Informationen sind für die Durchführung von Prozess „X“ notwendig?	Liste von Informationen
Welche Produkte können mit Hilfe der Technischen Ressource „T“ hergestellt werden?	Liste von Produkten
Welche Technische Ressource kann den Prozess „X“ durchführen?	Liste Technischer Ressourcen
Welche Energien werden für den Prozess „X“ benötigt?	Liste von Energien

der übertragbar, wie zum Beispiel modulare verfahrenstechnische Anlagen [Blo+18]. Dies ist darin begründet, dass die genutzten Begriffe vergleichsweise generisch sind und keinerlei direkten Bezug zu Verpackungsmaschinen besitzen. Der Standard beschreibt 17 verschiedene Zustände, die sich in zwei Kategorien aufteilen lassen:

- Aktive Zustände: Repräsentieren die aktuelle Durchführung einer Menge von Aktivitäten. Dabei können einfache oder wiederholte Ausführungen von Prozessen in einer logischen Reihenfolge, bestimmten Zeit, oder bis zu einem definierten Abbruchkriterium erfolgen. Dies umfasst beispielsweise Zustände wie „Execute“ oder „Holding“.

- Wartende Zustände: Stellen dar, dass die Maschine eine bestimmte Menge von Bedingungen erfüllt hat und auf die Erfüllung weiterer Bedingungen wartet. Dies umfasst beispielsweise Zustände wie „Idle“, oder „Stopped“.

Auf Basis der in [ANSI/ISA 88] enthaltenen Begriffe, wurde das Lightweight-Ontologie-Design-Muster in Abbildung III.14 entworfen. Demnach besitzen Zustände eingehende und ausgehende Transitionen. Zustände und Transitionen besitzen hingegen Subklassen, die weitere Semantik ergänzen. Wie der Abbildung zu entnehmen ist, wurden aus Gründen der Übersichtlichkeit nicht alle Zustände und Transitionen der letzten Hierarchiestufe modelliert. Stattdessen ist nur angedeutet, dass alle aktiven Zustände („{ActingStates}“) als Spezialisierung der Klasse „Acting“ anzulegen sind. Dies gilt analog für die wartenden Zustände („{WaitingStates}“), die Kommandos („{Commands}“) so wie für die abschließenden Transitionen („{Completions}“). Darüber hinaus sind in der Lightweight-Variante nicht alle logisch möglichen Verknüpfungen von Zuständen und Transitionen enthalten, was in der Ausdrucksstärke von UML-Klassendiagrammen begründet ist. Beispielsweise kann nicht ausgedrückt werden, dass ein bestimmtes Kommando wie „Start“ nur mit dem Zustand „Idle“ verbunden werden darf. Die Heavyweight-Variante hingegen beinhaltet die zulässigen Verbindungen zwischen Zuständen und Transitionen entsprechend der Matrixdarstellung der Zustandsmaschine in [ANSI/ISA 88].

Exemplarische Kompetenzfragen, welche mit Hilfe des ODM beantwortet werden können, sind in Tabelle III.5 zu finden. Listing C.3 stellt ein assertionales Muster zur Erstellung eines Zustandsautomaten dar.

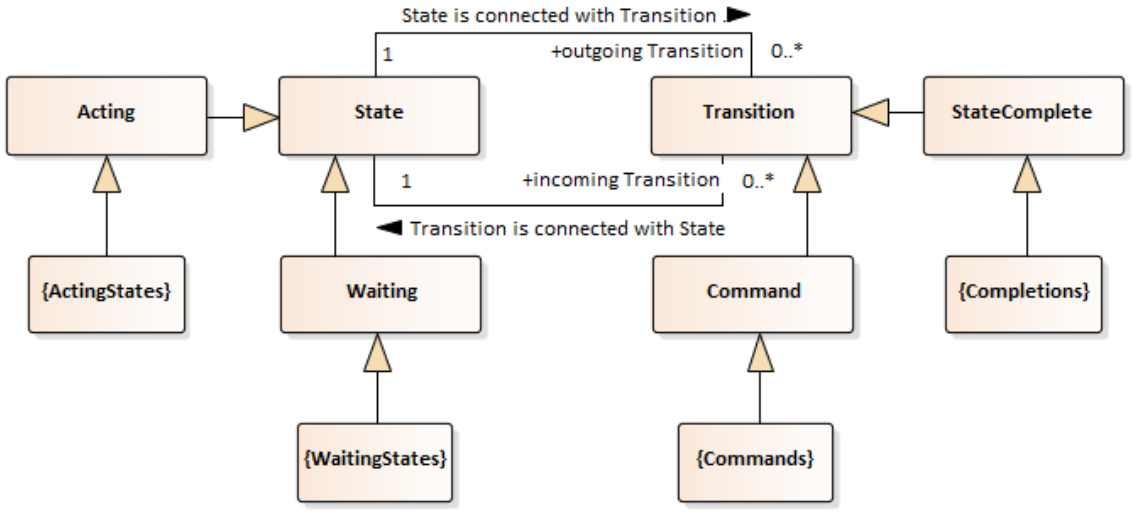

Abbildung III.14.: Lightweight-Ontologie-Design-Muster der Zustandsmaschine der [ANSI/ISA 88], Abbildung in Anlehnung an [Hil+20*]

Tabelle III.5.: Exemplarische Kompetenzfragen für das ISA 88 ODM

Kompetenzfrage	Antwort	Restriktion
Was ist der aktuelle Zustand des Systems „S“?	Aktueller Zustand	ODM für Systemstrukturen
Warum ist System „S“ nicht in Zustand „Z“?	Aktueller Zustand und relevante Transitionen	
Welche Systeme „S“ sind aktuell im Zustand Idle?	Liste der Systeme	ODM für Systemstrukturen
Was sind mögliche Zustände des Systems „S“?	Liste der Zustände	
Wielange war System „S“ im Zustand Idle?	Zeit in Stunden	ODM für Zeitbeschreibung

III-4.3. Modellierung von Systemstrukturen basierend auf der VDI 2206

Die Richtlinie [VDI 2206] beschreibt die Entwicklung mechatronischer Systeme und definiert notwendige Begriffe und Relationen zu deren struktureller Beschreibung. Diese Begriffe und Relationen wurden in das Lightweight-Ontologie-Design-Muster der Informationsressource in Abbildung III.15 überführt. Den Beschreibungen der [VDI 2206] folgend besteht ein mechatronisches System aus einem Basissystem, Sensoren, Aktoren und einer Informationsverarbeitung. Während das Basissystem das physische System repräsentiert, welches durch Sensoren erfasst und durch Aktoren beeinflusst wird, beinhaltet die Informationsverarbeitung die notwendige Logik zur Interpretation der von den Sensoren erzeugten Daten, was zu einer der Logik entsprechenden Steuerung bzw. Regelung der Aktoren führt. Des Weiteren definiert die Richtlinie drei Arten von Flüssen. Produktflüsse, Energieflüsse und Informationsflüsse, welche zwischen Komponenten des mechatronischen Systems fließen können. Letztlich erlaubt die [VDI 2206] die Definition einer hierarchischen Struktur, in welcher ein mechatronisches System aus verschiedenen Modulen besteht, welcher miteinander verbunden und komponiert bzw. dekomponiert werden können. Die unterste Ebene eines Moduls besteht jedoch immer aus den Eingangs erwähnten Komponenten.

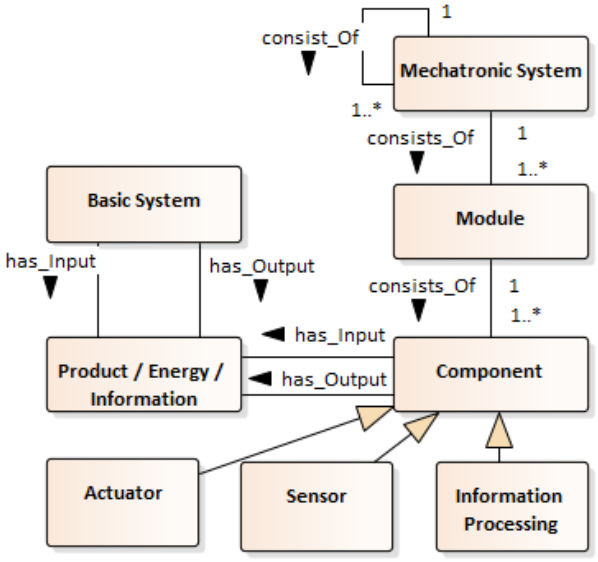

Abbildung III.15.: Lightweight-Ontologie-Design-Muster der Strukturbeschreibung mechatronsicher Systeme entsprechend [VDI 2206], Abbildung aus [Hil+20*]

Exemplarische Kompetenzfragen, welche mit Hilfe des ODM beantwortet werden können, sind in Tabelle III.6 zu finden. Listing C.4 stellt ein assertionales Muster zur Beschreibung von hierarchischen Systemstrukturen und Listing C.5 ein assertionales Muster zur Beschreibung eines mechatronischen Systems dar.

Tabelle III.6.: Exemplarische Kompetenzfragen für das VDI 2206 ODM

Kompetenzfrage	Antwort
Woraus besteht System „S“	Liste aller Entitäten in „S“
Aus welchen Modulen besteht das System „S“?	Liste der Module
Aus welchen Komponenten besteht das System „S“?	Liste der Komponenten
Was detektiert der Sensor „SE“?	Detektierte Entität

III-4.4. Modellierung von Datenelementen basierend auf der DIN EN 61360

Wie in [Hil+17*b] vorgestellt, können mit Hilfe von Datenelementen die Eigenschaften von Entitäten (z.B. Prozesse, Transitionsbedingungen etc.) beschrieben werden, wobei es nötig ist, sowohl den Typ des Datenelements (Datenelementtyp, bzw. nachfolgend kurz „Typbeschreibung“) zu beschreiben als auch die Instanz einer Typbeschreibung. Entsprechend [Hil+17*b] kann, wie nachfolgend beschrieben, für die Typbeschreibungen die [DIN EN 61360-1] und für die Beschreibung der Instanzen eine auf [Epp11] beruhende Ergänzung der [DIN EN 61360-1] genutzt werden. Die Typbeschreibung stellt eine Eigenschaft eines physischen oder logischen Objektes dar, die den Träger eines Datenelementes charakterisiert. Typbeschreibungen werden durch Attribute eindeutig charakterisiert, wobei Attribute der Typbeschreibung folgendermaßen klassifiziert werden können:

- Identifizierende Attribute, z.B..: ID, bevorzugter Name, Versionsnummer
- Semantische Attribute, z.B.: Definition, Quellendokument der Definition des Datenelementtyps, Formel
- Werteattribute, z.B..: Datentyp, Maßeinheit, Werteliste
- Relationale Attribute, z.B.: Übergeordnete Klasse eines Datenelementtyps

Bei der Instanziierung einer Typbeschreibung erhält das Datenelement zusätzlich eine Instanzbeschreibung, um die Ausprägung der Eigenschaft, bspw. durch Angabe eines konkreten Wertes, abzubilden. Hierzu werden die ergänzenden Attribute aus [Epp11] herangezogen, die die Instanzbeschreibung des Datenelementes näher charakterisieren. Dabei enthält die Instanzbeschreibung drei wesentliche Attribute: das Aussageziel, die Aussagenlogik und den eigentlichen Wert. Die Aussagenlogik definiert, ob der eigentliche Wert als kleiner, größer oder gleich zu interpretieren ist. Das Aussageziel hingegen gibt an, auf welche Art der Wert zu interpretieren ist, d.h. welche Aussage von dem Bereitsteller des Datenelements beabsichtigt wird. Es ist zwischen den folgenden Aussagezielen zu unterscheiden:

- Anforderungsausprägung (as required)
- Zusicherungsausprägung (as designed)
- Istausprägung (as built bzw. after use)

Das auf den Ausführungen von [Hil+17*b] und den Informationsressourcen [DIN EN 61360-1] und [Epp11] beruhende Lightweight-Ontologie-Design-Muster ist in Abbildung III.16 dargestellt. Um die Abbildung zu vereinfachen, wurden die UML-Attribute der Typbeschreibung („Type Description“) entsprechend obiger Aufzählung zusammengefasst.

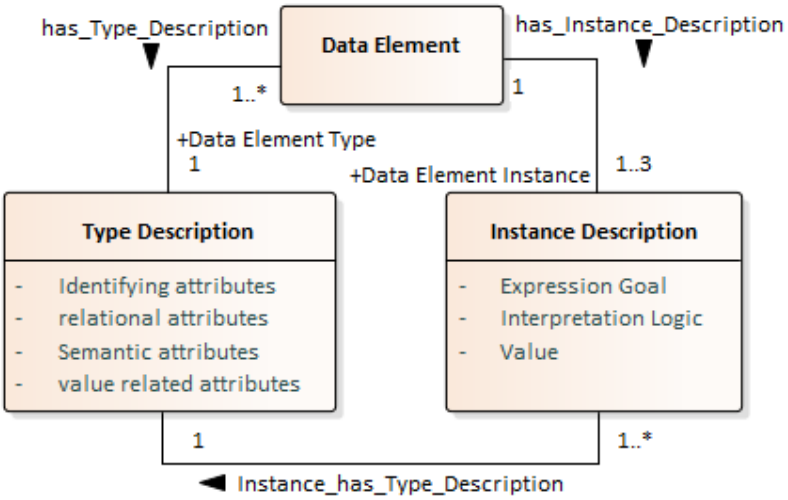

Abbildung III.16.: Lightweight-Ontologie-Design-Muster basierend auf der DIN EN 61360, Abbildung aus [Hil+20*]

Exemplarische Kompetenzfragen, welche mit Hilfe des ODM beantwortet werden können, sind in Tabelle III.7 zu finden. Listing C.6 stellt ein assertionales Muster zur Erzeugung einer Typbeschreibung und Listing C.7 assertionales Muster zur Erzeugung einer Instanzbeschreibung entsprechend der [DIN EN 61360-1] dar.

Tabelle III.7.: Exemplarische Kompetenzfragen für das DIN EN 61360 ODM

Kompetenzfrage	Antwort	Restriktion
Welche Merkmale hat das System „S“?	Liste der Typbeschreibungen	ODM für Systemstrukturen
Was ist der Wert des Merkmals „M“ von System „S“?	Wert der Instanzbeschreibung	ODM für Systemstrukturen
Was ist die Definition des Merkmals „M“?	Definition	
Welche Entitäten nutzen das Merkmal „M“?	Liste der Entitäten	
Handelt es sich bei Wert „I“ um eine Anforderung?	Aussageziel von „I“	
Welche Typen von Aktualwerten besitzt die Entität „E“?	Liste der Typbeschreibungen	

III-4.5. Alignment-Ontologie

Die in diesem Abschnitt dargestellte Alignment-Ontologie für die Ontologie-Design-Muster der Abschnitte III-4.1 bis III-4.4 basiert auf der Darstellung aus [Hil+19*], welche die vorgestellten Ontologie-Design-Muster durch Komposition entsprechend den Vorgaben aus [Hil+17*b] zusammenfügt. Für einen ersten Überblick über die vorgenommenen Alignment-Operationen wird Abbildung III.17 herangezogen. Demnach wurde die Alignment-Ontologie durch Import der einzelnen Ontologie-Design-Muster erstellt und verfügt über keinerlei zusätzliche Begriffe. Jedoch wurden zwischen allen Ontologie-Design-Mustern Kompositionsoperationen vorgenommen.

Die Kompositionsoperationen werden nachfolgend entsprechend den Ausführungen aus [Hil+19*] vorgestellt, wobei die im Rahmen der Komposition hinzugefügten Relationen in in Abbildung III.18 grün hervorgehoben sind. Drei Relationen wurden zwischen den Begriffen der [VDI/VDE 3682-1] und [VDI 2206] angelegt, wobei zwei davon eine Äquivalenz-Beziehung ausdrücken. Die erste Äquivalenz-Beziehung wird zwischen Produkt, Energie und Information der beiden Standards angelegt, da in beiden Standards die Begriffe für Produkt, Information und Energie mit identischer Bedeutung versehen sind. Das Basissystem der [VDI 2206] hingegen beschreibt das physikalische System, welches durch das mechatronische System zu steuern bzw. zu regeln ist, was als äquivalent zum Prozess der [VDI/VDE 3682-1] zu verstehen ist, welcher durch eine Technische Ressource realisiert wird. Die Technische Ressource und das mechatronische System hingegen sind nicht als äquivalent aufzufassen, da die [VDI 2206] ein konkretes System beschreibt, während die Technische Ressource der [VDI/VDE 3682-1] auch zur Beschreibung abstrakter Anforderungen an ein System genutzt werden kann [SHF17*]. Aus diesem Grund wird die Beziehung der `structural realization` zwischen der Technischen Ressource und dem mechatronischen System angelegt. Zwischen

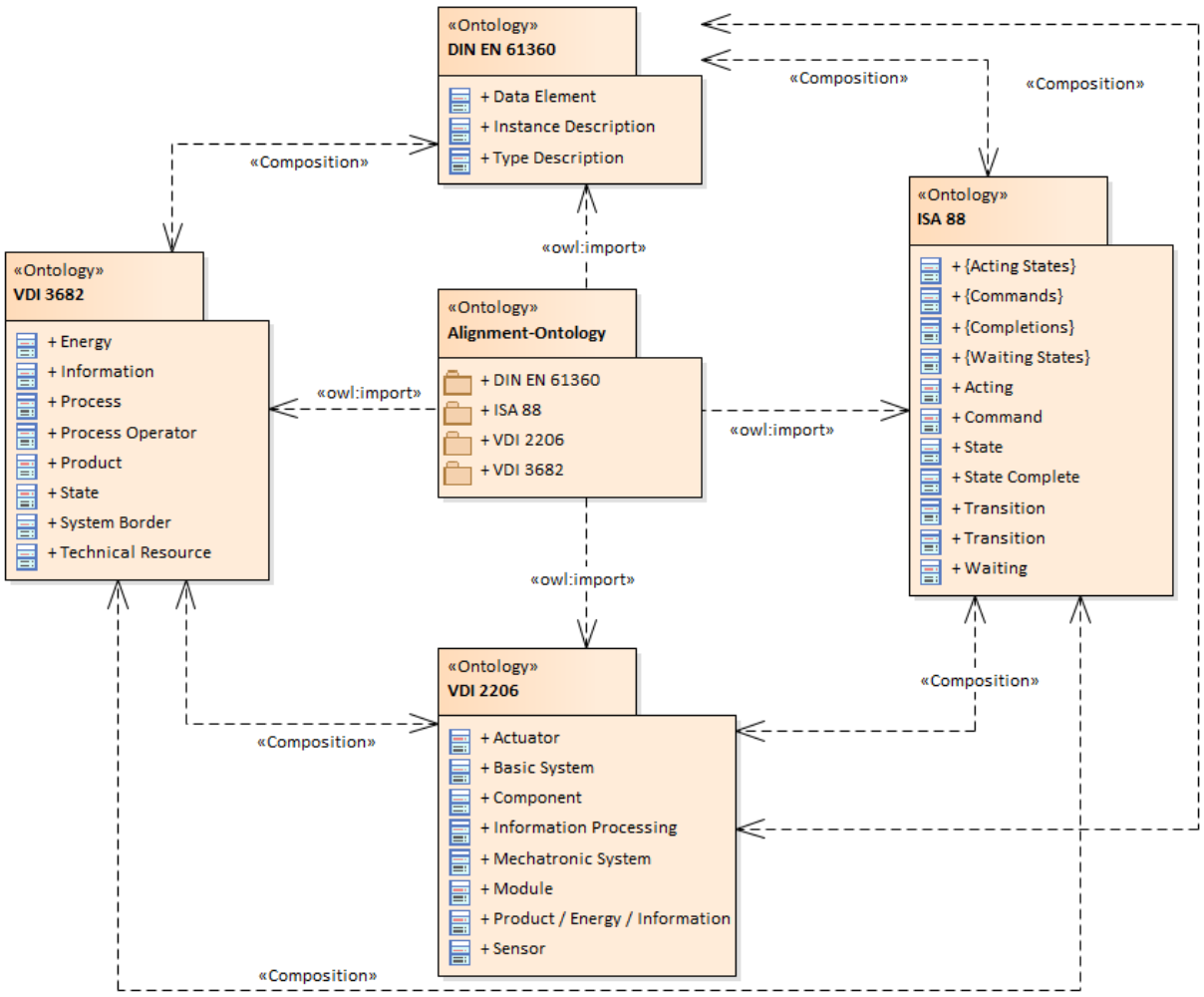

Abbildung III.17.: Übersicht über die Lightweight-Alignment-Ontologie für die Ontologie-Design-Muster der VDI 3682, VDI 2206, ISA88 und DIN EN 61360

der Technischen Ressource der [VDI/VDE 3682-1] und den Zuständen und Transitionen der [ANSI/ISA 88] wird die Beziehung `has behavioral realization` angelegt. In diesem Sinne können die Zustände und Transitionen der [ANSI/ISA 88] dann zur Beschreibung des Verhaltens einer Technischen Ressource genutzt werden.

Die Beziehungen zwischen der [VDI 2206] und der [ANSI/ISA 88] sind zweierlei. Zunächst existiert eine Verbindung von Komponenten der [VDI 2206] zu Transitionen der [ANSI/ISA 88] um anzuzeigen, dass diese Komponenten bereitstellen, welche in der Beschreibung von Transitionsbedingungen vorkommen. Des Weiteren kann jedes mechatronische System, Modul oder jede Komponente sich in einem Zustand befinden, welcher im Modell der [ANSI/ISA 88] zu finden ist. Hinsichtlich der Beschreibung mit Datenelementen basierend auf der [DIN EN 61360-1] wird der Empfehlung in [Hil+17*b] gefolgt, wonach ein Datenelement grundsätzlich jede Art von Entität beschreiben kann. Daher kann ein Datenelement des Ontologie-Design-Musters der [DIN EN 61360-1] mit Individuen jeder Klasse der anderen Ontologie-Design-Muster verbunden werden.

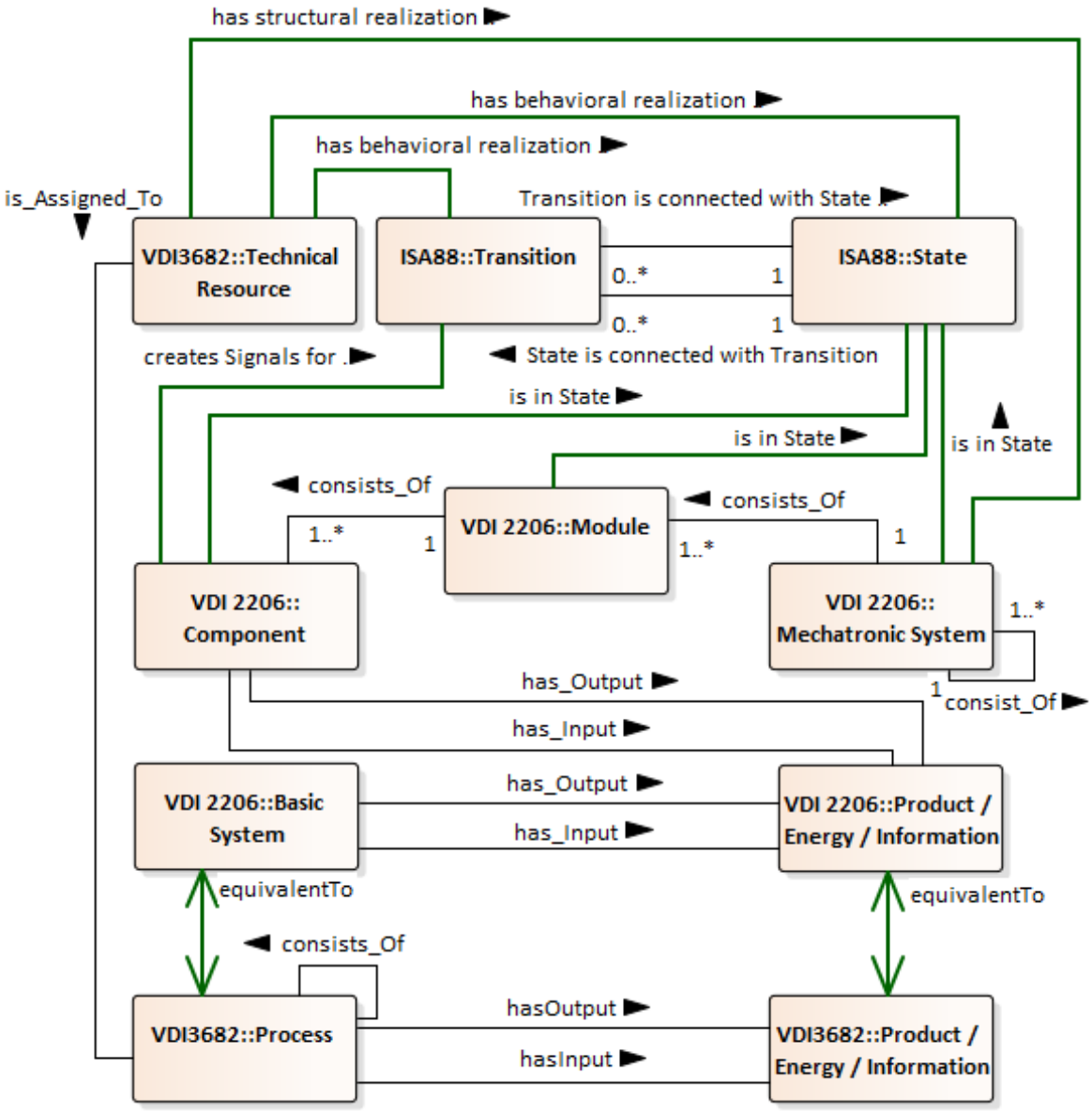

Abbildung III.18.: Lightweight-Alignment-Ontologie für die Ontologie-Design-Muster der VDI 3682, VDI 2206, ISA88 und DIN EN 61360, Abbildung aus [Hil+20*]

IV. Ontologie-Entwicklung ABox

Das vorliegende Kapitel widmet sich der Entwicklung einer Menge von Artefakten zur Erfüllung von Forschungsteilziel III. Hierzu stellt Abschnitt IV-1 zunächst die Grundlagen zur ABox-Modellierung vor, bevor Abschnitt IV-2 den relevanten Stand der Wissenschaft darstellt. Im Anschluss widmet sich der Abschnitt IV-3 der Darstellung der Methodik, welche zur Erreichung von Forschungsteilziel III entwickelt wurde.

IV-1. Grundlagen zur ABox-Modellierung

Der vorliegende Abschnitt definiert relevante Grundlagen zur ABox-Modellierung im Kontext der vorliegenden Arbeit. Zur Strukturierung der nachfolgenden Grundlagenabschnitte werden folgende Wissensfragen genutzt:

1. Wie kann eine ABox manuell modelliert werden?
2. Wie kann eine ABox automatisiert erstellt werden?
3. Wie kann zwischen zeitlich veränderlichen Bestandteilen und zeitlich unveränderlichen Bestandteilen der ABox methodisch unterschieden werden?
4. Welche Arten von Artefakten kommen für die automatisierte Erstellung der ABox in Frage?
5. Im Falle einer automatisierten Erstellung der ABox: hinsichtlich welcher Eignungskriterien sind die Artefakte zu prüfen?

Die erste Frage wird in Abschnitt IV-1.1.1 beantwortet, wobei das grundlegende Vorgehen zur manuellen Modellierung vorgestellt wird. Die zweite Frage wird in Abschnitt IV-1.1.2 beantwortet, wobei hier zwei Verfahren konzeptuell vorgestellt werden, welche durch die Ansätze in Abschnitt IV-2 auf unterschiedliche Art implementiert sein können. Die dritte und vierte Frage werden in Abschnitt IV-1.2.1 beantwortet, wobei hier auf unterschiedliche Arten von Artefakten eingegangen wird, welche für die Erstellung der ABox herangezogen werden können. Die letzte Frage wird in Abschnitt IV-1.2.2 beantwortet, wobei hier auf den Einfluss der Qualität der Artefakte auf die automatisierte Erstellung der ABox eingegangen wird.

IV-1.1. Verfahren zur Erstellung der ABox

Der vorliegende Abschnitt stellt relevante Grundlagen zur Erstellung der ABox dar. Hierzu geht Abschnitt IV-1.1.1 zunächst auf die manuelle Modellierung der ABox ein. Abschnitt IV-1.1.2 zeigt dann zwei unterschiedliche Konzepte zur automatisierten ABox-Erstellung auf.

IV-1.1.1. Manuelle Modellierung der ABox

Bei der manuellen Modellierung der ABox werden die Tripel der ABox manuell erstellt. Hierbei kommt meist ein Software-Werkzeug zum Einsatz, wie z.B. Protégé [Sta19@] oder TopBraid Composer [Top19@]. In [NoMc11@] wird die Nutzung der genannten Werkzeuge zur ABox-Entwicklung dargestellt. Analog zur Modellierung der TBox werden hierbei für die ABox einzelne Individuen manuell definiert und mit Object- und DatatypeProperties versehen. Die Automatisierung der Erstellung geht dabei nicht über die Erstellung eines Tripels pro Nutzereingabe hinaus, was insbesondere die Erstellung großer ABoxen erschwert (siehe auch Abschnitt I-2.2.2). Dass es sich hierbei auch nicht um den ursprünglichen Einsatzweck der Werkzeuge handelt, wird in [Mus15] herausgestellt. Ursprünglich zielte die Entwicklung von Protégé darauf ab, im medizinischen und biologischen Bereich Wissensbasen aufzubauen und hierbei entsprechende Klassifikationen durch Nutzereingaben zu erstellen. Das Haupteinsatzgebiet der genannten Werkzeuge ist demnach die Erstellung der TBox, auch wenn eine Erstellung der ABox grundsätzlich möglich ist [Mus15].

IV-1.1.2. Automatisierte Erstellung der ABox

Im Rahmen der automatisierten Erstellung der ABox werden sogenannte Mappings definiert, welche die Inhalte einer Datenquelle in Inhalte der ABox übersetzen [Dim+14]. Mappings stellen demnach eine Art Übersetzung dar, welche entsprechend [Cal+16] generell auf zwei unterschiedliche Arten genutzt werden können: zur Materialisierung oder zur Virtualisierung. Im Rahmen eines Materialisierungsansatzes wird ein Mapping ausgeführt und die resultierende ABox in einer entsprechenden Datenbank oder als Datei persistent gespeichert. Dies ist in Abbildung IV.1 als „1“ dargestellt. Diese Datenbank oder Datei kann dann von einer Applikation mit Hilfe von SPARQL abgefragt werden, was in Abbildung IV.1 als „2“ dargestellt wird. Ändern sich die Daten in der Datenquelle, so muss für eine Synchronisation der Daten das Mapping erneut ausgeführt und die ABox erneut materialisiert werden. Als Verfahren kann hierbei ein „Extract Transform Load“ (ETL) Verfahren verwendet werden [Aue+13].

Im Rahmen eines Virtualisierungsansatz erfolgt eine Übersetzung einer eingehenden SPARQL-Abfrage durch einen Mediator, welcher die SPARQL-Anfrage mittels des Mappings in die Abfragesprache der Datenquelle (z.B. SQL) übersetzt [Cal+16]. Dies ist in Abbildung IV.1 als „3“ gekennzeichnet. Demnach benötigt der Virtualisierungsansatz keine zusätzliche Datenbank und die Daten der Datenquelle werden nicht dupliziert [Aue+13]. Die ABox existiert zu keinem Zeitpunkt persistent, sondern liegt lediglich zum Zeitpunkt der Anfrage des Clients als transiente Menge an Tripeln vor, welche durch den Mediator bereitgestellt wird.

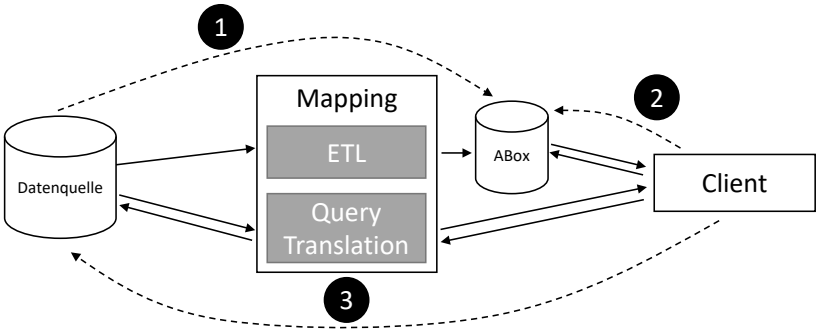

Abbildung IV.1.: Arten der automatisierten ABox-Erstellung in Anlehnung an [Aue+13]

Als Verfahren kann hierbei ein „Query Translation“ Verfahren verwendet werden [Cal+16]. Unabhängig vom gewählten Ansatz muss ein Mapping vorliegen, welches entweder in einem ETL-Verfahren oder in einer „Query Translation“-Lösung genutzt wird [Civ+13].

IV-1.1.2.1. Extract Transform Load

Ein „Extract Transform Load“ Prozess besteht aus drei Schritten: dem Extrahieren von Daten aus einer Datenbank oder einer Datei, einer Transformation dieser Daten entsprechend den Anforderungen eines Ziel-Systems und letztlich dem Laden der Daten in eine Ziel-Datenbank oder eine Ziel-Datei [Vil+17]. Im Rahmen der Extraktion können hierbei Daten aus unterschiedlichen Datenformaten (z.B. CSV, XLS etc.) herangezogen werden oder bei einem Restful-Client angefragt werden [BaKa15]. Im Rahmen der Transformation werden Aufgaben durchgeführt, welche sich insbesondere auf die Bereinigung, Normalisierung, das Entfernen von Duplikaten sowie dem Anwenden von bestimmten Filtern beziehen [BaKa15]. Im Rahmen der ABox-Erstellung kann ein ETL-Verfahren genutzt werden, um die Daten aus verschiedenen Formaten und Quellsystemen zu extrahieren und auf diese Weise bspw. verschiedene Engineering-Artefakte in einer ABox zu vereinen [Lie+19]. Ein auf [Lie+19] basierendes Beispiel hierfür ist in Abbildung IV.2 dargestellt und wird nachfolgend erläutert. Hierbei werden Produktinformationen aus unterschiedlichen Datenbanken („SAP“, „Entwicklungsdatenbank“) durch Abfrage in der Quellsprache (z.B: SQL) extrahiert und deren Eindeutigkeit in einer gemeinsamen Datenbank sichergestellt. Die Eindeutigkeit wird hierbei durch Vergabe von eindeutigen Bezeichnern in der „Import-DB“ realisiert. Basierend auf dieser Datenbank wird mittels der Mappingsprache „RDB to RDF Mapping Language“ (R2RML), welche in Abschnitt IV-2.2.1.3 näher erläutert wird, ein Mapping definiert. Dieses Mapping definiert, welche Daten der Datenbank welchen Entitäten der genutzten Ontologie entsprechen. Bei Ausführung des Mappings entsteht die ABox, welche im weiteren Verlauf mittels SPARQL durch einen Client abgefragt werden kann.

IV-1.1.2.2. Query Translation und Ontology-based Data Access

Im Rahmen der „Query Translation“ wird eine Umwandlung der semantischen Abfragesprache (SPARQL) in eine andere Sprache (z.B. SQL) vorgenommen [Civ+13]. Die Übersetzung findet in einer Middleware statt, welche auch als OBDA-System (englisch:

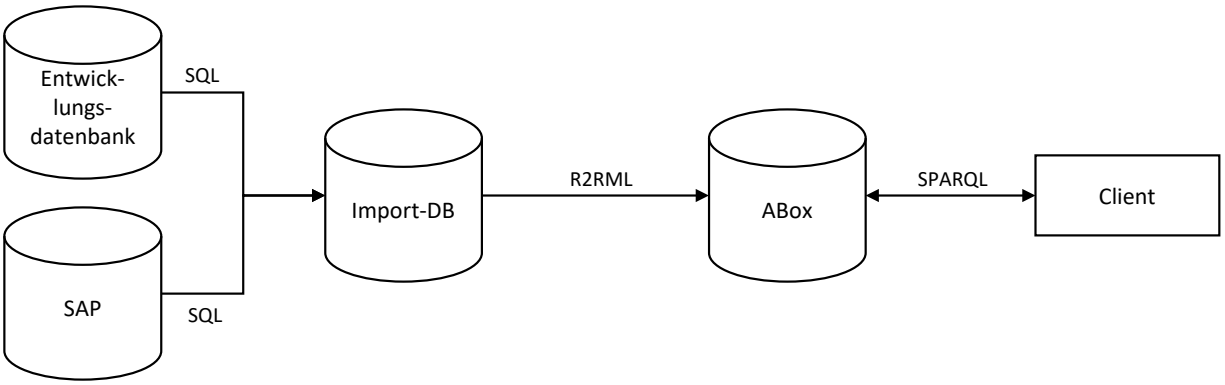

Abbildung IV.2.: Vereinfachter ETL-Prozess zur Überführung von Engineering-Artefakten in Anlehnung an [Lie+19]

„Ontology-based Data Accesss“) bezeichnet wird. Es bedarf somit keines zusätzlichen Speichermediums, in welchem die Daten repliziert und abgelegt werden [Cal+16].

Ein OBDA-System basiert auf drei Layern, wobei es sich um einen konzeptionellen Layer, einen Mappinglayer und einen Datenquellenlayer handelt [Civ+13]. Der Datenquellenlayer beinhaltet die tatsächlichen Daten, wobei eine Vielzahl an Datenquellen mit unterschiedlichen Formaten und Semantiken betrachtet werden können. Der konzeptionelle Layer beinhaltet die TBox einer Ontologie, welche ein geteiltes Vokabular über alle Software-Applikationen hinweg bereitstellt. Der Mappinglayer definiert letztlich die notwendigen Verbindungen zwischen Inhalten der Ontologie und Inhalten der Datenquellen. [Civ+13]

Ein Beispiel für ein OBDA-System ist in [Pet+17] zu finden. Eine Applikation, die auf die ABox zugreifen soll, muss demnach ihre SPARQL-Anfrage an eine Middleware senden. Die Middleware sorgt dann mit Hilfe vorliegender Mappings dafür, dass die Anfragen in die entsprechenden Sprachen der Zielsysteme übersetzt werden, und führt diese Anfragen dann in dieser Sprache aus. Hierzu befindet sich in der Middleware der konzeptuelle Layer, sowie der Mappinglayer. Im Beispiel von [Pet+17] befindet sich im konzeptuellen Layer eine TBox, welche Begriffe und Relationen zur Beschreibung von Fabriken und Werkzeugmaschinen enthält. Diese Begriffe und Relationen können in den SPARQL-Anfragen des Clients verwendet werden. Der Mappinglayer beinhaltet Mappings, welche die Begriffe und Relationen mit real vorliegenden Datenbankeinträgen verbindet, bspw. mit EInträgen in SQL-Datenbanken. Die Middleware führt diese Mappings bei Vorliegen einer Anfrage aus und übersetzt somit die SPARQL-Anfrage in eine Abfrage des Zielsystems (z.B. SQL). Dabei ist es auch möglich, dass bestimmte Teile der ABox materialisiert vorliegen. Die Architektur eines OBDA-Systems wird in [Hil+20*] erläutert und ist in Abbildung IV.3 vereinfacht dargestellt. Demnach sendet ein Client eine SPARQL-Anfrage an eine Graph-Datenbank, welche als die oben genannte Middleware dient. Die Graph-Datenbank enthält die Menge aller Tripel der TBox und stellt somit den konzeptuellen Layer dar. Darüber hinaus besitzt sie die Fähigkeit Mappings auszuführen und somit die ABox in Form materialisierter oder virtueller Graphen aus Datenquellen zu extrahieren und in die Beantwortung der SPARQL-Anfrage einzubeziehen.

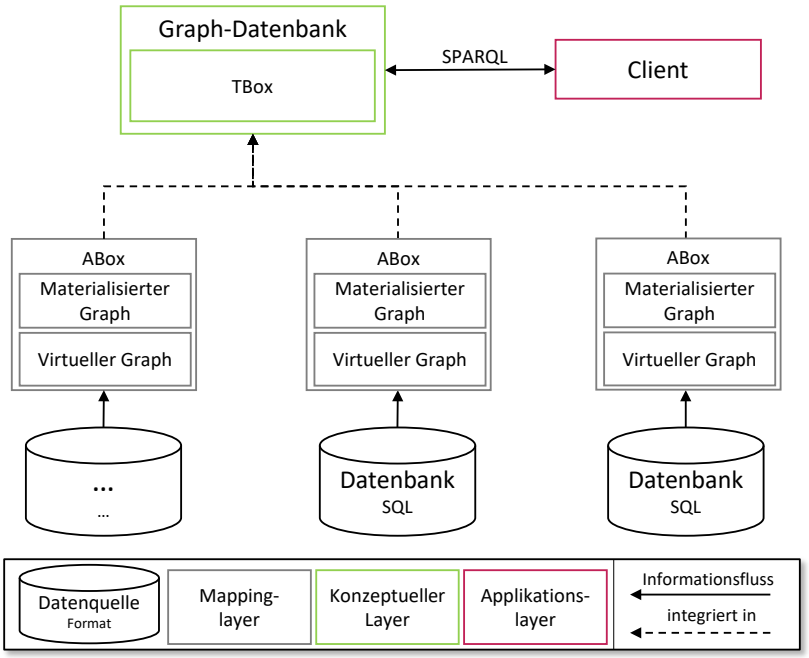

Abbildung IV.3.: Architektur eines OBDA-Systems in Anlehnung [Hil+20*]

IV-1.2. Artefakte und deren Eignung für automatisierte Verfahren

Der vorliegende Abschnitt stellt relevante Grundlagen zu Artefakten dar, welche für eine automatisierte Erstellung der ABox herangezogen werden können. Hierzu werden in Abschnitt IV-1.2.1 zunächst zwei Kategorien von Artefakten eingeführt. Abschnitt IV-1.2.2 geht dann auf bestimmte Charakteristika von Artefakten ein, welche für die automatisierte Erstellung der ABox von Relevanz sein können.

IV-1.2.1. Engineering- und Laufzeit-Artefakte

Da sowohl bei einer Virtualisierung als auch bei einer Materialisierung (siehe Abschnitt IV-1.1.2) stets die Daten der Datenquelle zu einem spezifischen Zeitpunkt in die ABox einer Ontologie überführt werden, ist es von Interesse, welcher zeitlichen Veränderung die Daten der Datenquelle unterworfen sind. Hierbei kann grundsätzlich in quasi-statische und dynamische Daten unterschieden werden [Gra18]. Quasi-statische Daten ändern sich eher im Bereich von Tagen und Wochen, dynamische Daten hingegen können sich im Millisekundenmaßstab (z.B. Zustandsvariablen einer Steuerung) ändern. Grundsätzlich eignet sich ein Virtualisierungsansatz eher bei dynamischen Daten als ein Materialisierungsansatz, da hier die Menge an auszuführenden Mappings durch die Anzahl der Anfragen und nicht die Änderungsrate der Daten bestimmt wird. [Gra18]

In Bezug auf quasi-statische und dynamische Daten werden im Rahmen dieser Arbeit zwei Kategorien von Artefakten referenziert: Engineering- und Laufzeit-Artefakte. Als Engineering-Artefakte werden dabei Artefakte der Gewerke Maschinenbau, Elektrotechnik

und Automatisierungstechnik (siehe Abschnitt I-2.1.1) in den Datenformaten entsprechend Abschnitt I-2.1.2.2 bezeichnet, welche als quasi-statisch betrachtet werden. Als Laufzeit-Artefakte werden Daten eines Systems aus dem Betrieb bezeichnet, wobei hierzu das entsprechende Schema (z.B. OPC UA Informationsmodell als XML, oder SQL-Schema) und die dazugehörigen Daten der Datenquelle zu einem spezifischen Zeitpunkt gehören.

OPC Unified Architecture (OPC UA) ist eine Middleware-Lösung für die Machine-to-Machine-Kommunikation, welche die Möglichkeit bietet, ein Informationsmodell zur Strukturierung der ausgetauschten Daten zu definieren [MLD09]. OPC UA wird durch die Normenreihe DIN 62541 standardisiert, wobei [DIN 62541-1] die Übersicht über die Norm darstellt. Teil fünf der Norm [DIN EN 62541-5] stellt das Informationsmodell dar, wobei ein XML Schema definiert wird, welches zur Darstellung aller OPC UA Informationsmodelle genutzt werden kann. Hierbei wird grundlegend ein objektorientiertes Prinzip als Aufbau genutzt.

IV-1.2.2. Qualität, Offenheit und Reifegrade

Die Qualität der für ein Mapping herangezogenen Artefakte hat einen Einfluss auf den Aufwand der Erstellung der Mappings und kann die generelle Möglichkeit, ein Mapping durchzuführen, beeinflussen. Die Qualität von Daten bzw. Informationen wird meist hinsichtlich des Beitrags zur Erreichung der Ziele des Datenempfängers bestimmt [Hel02]. Im Rahmen der Erstellung eines Mappings ist es notwendig, die relevanten Datenobjekte in einem Artefakt zu identifizieren und bei der Erstellung des Mappings zu berücksichtigen. Bestimmte Eigenschaften des Artefakts können diese Tätigkeit erschweren. Handelt es sich bei dem Artefakt bspw. um ein Artefakt mit einem proprietären Format, so muss dies unter Umständen erst in ein Format überführt werden, welches durch einen Mappingansatz (siehe Abschnitt IV-2) verarbeitet werden kann. Analog kann die Vergabe von Bezeichnern, welche lediglich im Kontext des Artefakts eindeutig sind, die Erstellung des Mappings erschweren, da dann im Rahmen des Mappings die Eindeutigkeit im globalen Kontext sichergestellt werden muss [Lie+19]. Eine Bewertung der Offenheit und Reife mehrerer Artefakte ist somit notwendig, um den Einfluss der Qualität auf die resultierenden Aufwände abzuschätzen.

Offenheit von Engineering-Werkzeugen bezeichnet entsprechend [Bar+12] die Möglichkeit die Datenstrukturen der exportierten Artefakte für bestimmte Zwecke zu nutzen. Wie Abschnitt I-2.1.2.2 gezeigt hat, existieren unterschiedliche Formate von Engineering-Artefakten. Ist ein Engineering-Werkzeug nicht offen gestaltet, so ist dies hinderlich für die Interoperabilität des Werkzeugs mit anderen Engineering-Werkzeugen. In [Bar+12] wird daher eine Metrik eingeführt, mit deren Hilfe die Offenheit eines Engineering-Werkzeugs anhand der exportierten Daten bestimmt werden kann. Als Hauptkriterien werden hierbei das Exportformat, die Identifizierbarkeit von Datenobjekten, weitere formale Anforderungen wie Verifikation der Exportresultate und die Existenz und Erkennbarkeit von Objektbibliotheken, sowie die Vollständigkeit des Exports genannt.

In [Pun+16] wird ein Reifegradmodell zur systematischen Bewertung der Güte von Engineering-Daten (vornehmlich CAD-Layouts) anhand definierter Kriterien vorgestellt. Dies erfolgt im Kontext der virtuellen Inbetriebnahme [Pun17]. Das Reifegradmodell ist in Abbildung IV.4 dargestellt und wird nachfolgend in seinen wesentlichen Zügen erläutert. Das

Reifegradmodell nach [Pun+16] besteht aus vier Strängen (Hierarchie, Lesbarkeit, Einsetzbarkeit, Beschaffenheit), welche jeweils eine Menge von Kriterien besitzen. Diese Kriterien sind so angeordnet, dass eine Ableitung von Reifegradstufen möglich ist. Um beispielsweise Reifegrad 3 zu erreichen, ist es notwendig, alle Kriterien der Reifegrade 1, 2 und 3 vollständig zu erfüllen. Dem Strang Hierarchie werden alle Kriterien zugeordnet, die die Granularität der untersuchten Daten beschreiben (Grobgranulare, Mittelgranulare, Feingranulare und Feinstgranulare Hierarchie). Dem Strang Lesbarkeit hingegen werden alle Kriterien zugeordnet, die die Auswertung im Sinne der Lesbarkeit bzw. Verständlichkeit der zu untersuchenden Daten beschreiben (Lesbarkeit für den Menschen, standardisierte Form, eindeutige Auslegbarkeit, Maschinenlesbarkeit, Auswertbarkeit trotz fehlender Werte). Dem Strang Einsetzbarkeit werden alle Kriterien zugeordnet, die den konkreten Einsatz der zu untersuchenden Daten beschreiben. Zum Strang Beschaffenheit werden letztlich alle Kriterien zugeordnet, die ihre Ursachen im zeitlichen Verlauf haben. [Pun+16]

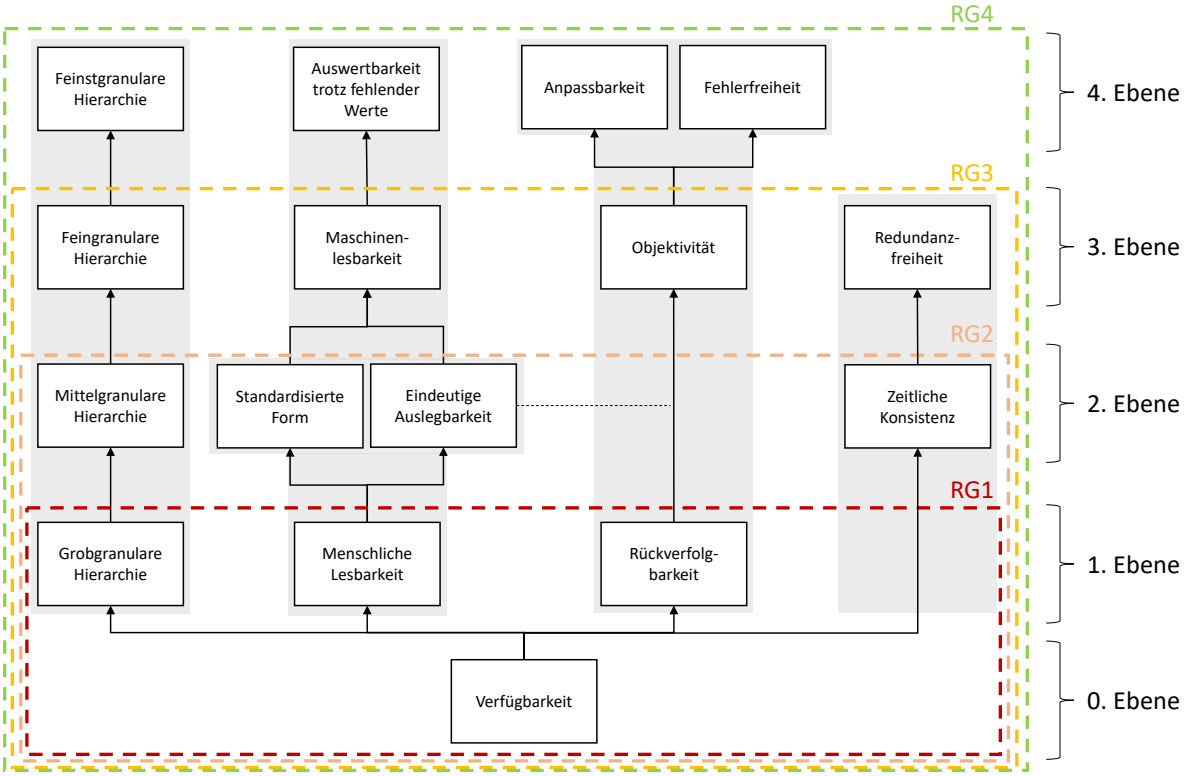

Abbildung IV.4.: Reifegradmodell für Engineering-Daten [Pun+16]

Die Offenheitsmetrik und das Reifegradmodell werden im Rahmen dieser Arbeit genutzt, um die Eignung von Engineering-Artefakten für ein automatisiertes Verfahren zur ABox Erstellung zu bestimmten. Abschnitt IV-3.2.2 wird hierauf näher eingehen.

IV-2. Stand der Wissenschaft zur ABox-Modellierung

Der vorliegende Abschnitt stellt den Stand der Wissenschaft zur ABox-Modellierung dar. Der Abschnitt IV-2.1 wird zunächst Kriterien vorstellen, welche für die Analyse des Stands der Wissenschaft herangezogen werden. Abschnitt IV-2.2 wird dann Methoden zur automatisierten ABox-Erstellung einführen. Die im Rahmen der vorliegenden Arbeit zu entwickelnde Methode ist als komplementär und nicht als Substitut zu den vorgestellten Methoden zu verstehen. Aus diesem Grund findet kein Abgleich mit den Anforderungen aus Abschnitt I-4.4 statt. Der Mangel dieses Abgleichs ist im Neuheitsgrad der vorliegenden Methode begründet. Da die Methode sich nicht auf ein einzelnes manuelles oder automatisiertes Vorgehen beschränkt, sondern stattdessen als Resultat ein Projekt-spezifisches Vorgehensmodell liefert, welches automatisierte und manuelle Vorgehen vereint, ist ein Abgleich mit existierenden manuellen oder automatisierten Vorgehen nur bedingt sinnvoll. Darüber hinaus fokussieren bisherige Ansätze eher auf technologische Umsetzungen als auf Engineering-Methoden.

IV-2.1. Kriterien zur Analyse des Stands der Wissenschaft

Zur Analyse des Stands der Wissenschaft werden nachfolgend Wissensfragen definiert, die für jede einzelne Veröffentlichung getrennt beantwortet werden sollen. Auf diese Weise soll ein einheitliches Bild des Stands der Wissenschaft zur automatisierten Erstellung der ABox geschaffen werden.

1. Welche Art von Artefakt wird im Rahmen der Methode berücksichtigt?
2. Welche Verfahrensart der Erstellung der ABox liegt vor?
3. Welche Reife hat der verwendete Formalismus der Mappings?
4. Wie können erstellte Mappings ausgeführt werden?

Im ersten Schritt ist die Frage zu klären, welche Art von Artefakt berücksichtigt wird, hinsichtlich des möglichen Formats des Artefakts. Die zweite Frage zielt auf den Unterschied zwischen Materialisierung und Virtualisierung ab, was einen entsprechenden Einfluss auf den Nutzungskontext des Verfahrens hat. Die dritte Frage zielt darauf ab, ob eine wiederverwendbare Sprache für die Mappings genutzt wird und ob diese evtl. standardisiert ist. Hierbei wird demnach in drei Kategorien unterteilt (standardisierte Mappingsprache vorhanden / offene, nicht-standardisierte Mappingsprache vorhanden / keine Mappingsprache vorhanden). Die vierte Frage zielt auf die Verfügbarkeit von Software-Lösungen zur Ausführung von Mappings ab. Dies ist darin begründet, dass eine entsprechende Mapping-Engine sowohl für ein ETL-Verfahren als auch eine „Query Translation“-Lösung benötigt wird, um einsetzbar zu sein. Es wird eine Unterteilung in drei Kategorien vorgenommen (Kommerzielle Lösung vorhanden / Nicht-kommerzielle Open-Source Lösung vorhanden / Keine Software-technische Lösung vorhanden).

IV-2.2. Verfahren für die automatisierte ABox-Erstellung

Die nachfolgenden Abschnitte stellen Verfahren für die automatisierte ABox-Erstellung vor. Hierzu geht Abschnitt IV-2.2.1 auf Verfahren ein, welche relationale Formate (CSV, XLS, SQL) nutzen. Abschnitt IV-2.2.2 stellt Verfahren vor, welche auf CAEX und darauf basierende Formate abzielen, und Abschnitt IV-2.2.3 stellt Verfahren vor, welche diverse Formate überführen können.

IV-2.2.1. Verfahren für Daten in relationalen Formaten

Nachfolgend werden Verfahren vorgestellt, welche Daten in relationalen Formaten wie z.B. CSV, XLS oder SQL-Datenbanken für eine Erstellung der ABox berücksichtigen.

IV-2.2.1.1. Methode nach O'Connor et al.

In [OHM10] wird eine Mapping-Sprache (Mapping-Master) beschrieben, die für die Überführung von Tabellenstrukturen nach OWL genutzt werden kann. Als Mapping-Engine ist ein Plug-In für Protégé verfügbar, mit dessen Hifle die Mappings definiert und ausgeführt werden können. Mögliche Artefakte sind CSV und XLS Dokumente, wobei ausschließlich Materialisierungen in lokale Dokumente möglich sind.

IV-2.2.1.2. Methode nach Blfgeh et al.

In [Blf+17] wird eine Mapping-Sprache (Tawny-OWL) beschrieben, welche eine domänenspezifische Sprache definiert, die wiederum auf der OWL-API aufbaut. Mit Hilfe dieser Mapping-Sprache besteht die Möglichkeit, die Mappings direkt im Code einer entsprechenden Mapping-Applikation zu definieren und auf diese Weise Tabellenstrukturen in die ABox einer Ontologie zu überführen. Mögliche Artefakte sind CSV bzw. XLS Dokumente, wobei lediglich Materialisierungen in ein lokales Dokument vorgesehen sind.

IV-2.2.1.3. RDB to RDF Mapping Language

Der W3C Standard „RDB to RDF Mapping Language“ (R2RML) ist in [W3C12[@]b] standardisiert und bietet eine Mapping-Sprache, welche auf die Überführung relationaler Daten in die ABox einer Ontologie spezialisiert ist. Da es sich um einen W3C Standard handelt, existieren diverse Mapping-Engines, die R2RML unterstützen, welche einerseits Open-Source (z.B. [KoSp15]), aber auch kommerzieller Natur ([Sta20[@]]) sind. Darüber hinaus existieren sowohl Ansätze zur Materialisierung (siehe [KoSp15]) als auch Virtualisierung (siehe [Cal+16]).

IV-2.2.2. CAEX-basierte Verfahren

Nachfolgend werden Verfahren vorgestellt, welche Daten im CAEX- bzw. AML-Format für eine Erstellung der ABox berücksichtigen.

IV-2.2.2.1. Methode nach Runde

In [Run10] wird ein Mapping-Ansatz vorgestellt, welcher auf die vollständige Überführung eines CAEX-Modells nach OWL abzielt. Hierzu wird dargestellt, welche CAEX-Modellelemente in welche OWL-Modellelemente überführt werden müssen. Der Ansatz stellt demnach keine Mapping-Sprache bereit, sondern sorgt für eine vollständige Nachbildung des CAEX-Modells in OWL. Eine Mapping-Engine ist ebenfalls nicht frei verfügbar, könnte aufgrund der Kürze des Ansatzes jedoch mit verhältnismäßig geringem Aufwand in einem Projekt erstellt werden.

IV-2.2.2.2. Methode nach Hua und Hein

Im Ansatz von [HuHe19] wird der Ansatz von [Run10] aufgegriffen und erweitert. Basierend auf einer Gegenüberstellung von OWL- und AML-Modellelementen sorgt eine entsprechende Mapping-Engine mit grafischer Oberfläche für die Überführung einer AML-Datei nach OWL. Hinzu kommt, dass der Ansatz eine bidirektionale Transformation ermöglicht und somit streng genommen eine dritte Kategorie von Ansätzen eröffnet, was jedoch im Rahmen dieser Arbeit nicht weiter betrachtet werden soll. Der Ansatz soll daher als Ansatz für die Materialisierung gelten.

IV-2.2.3. Formatübergreifende Verfahren

Nachfolgend werden Verfahren vorgestellt, welche Daten in unterschiedlichen Formaten für eine Erstellung der ABox berücksichtigen.

IV-2.2.3.1. Methode nach Dimou et al.

In [Dim+14] wird ein Ansatz vorgestellt, welcher die Mapping-Sprache von R2RML als Ausgangspunkt für die Generalisierung hin zu weiteren Formaten nutzt. Die Mapping-Sprache „RDF mapping Language“ (RML) ähnelt in ihrer Syntax daher stark der von R2RML und kann für die Überführung von CSV, XML, JSON verwendet werden. Als Mapping-Engines sind Implementierungen in unterschiedlichen Programmiersprachen verfügbar. Diese werden stetig um neue, performantere ergänzt (siehe z.B. [SKF19]). Wesentlicher Anwendungsfall ist die Nutzung im Rahmen einer Materialisierung.

IV-2.2.3.2. SPARQL-Generate

In [LZB17] wird eine Ergänzung der Abfragesprache SPARQL vorgestellt, mit deren Hilfe Mappings von XML, JSON, CSV, HTML und CBOR (Concise Binary Object Representation) nach OWL ermöglicht werden. Essentieller Unterschied zu anderen Mapping-Sprachen ist, dass mit Ausnahme von einigen Ergänzungen die Syntax von SPARQL-Abfragen genutzt werden kann um aus einer Datenquelle Tripel zu erstellen. Um die Anwendbarkeit das Ansatzes zu fördern, wurde durch die Autoren eine grafische Oberfläche entwickelt sowie eine entsprechende Bibliothek zur Einbettung in projektspezifische Applikationen. Wesentliches Einsatzfeld ist die Materialisierung.

IV-2.2.3.3. Methode nach Glawe

In [Gla+15] wird ein Ansatz vorgestellt, mit dessen Hilfe CAEX- bzw. AML-Dokumente nach OWL überführt werden könne. Hierzu wird die W3C Submission SWRL (Semantic Web Rule Language, siehe [Hor+04@]) verwendet. In diesem Ansatz beinhalten SWRL-Regeln auf der Prämissenseite eine Darstellung von zu findenden Datenobjekten der Datenquelle und auf der Konklusionsseite eine zu erstellende Menge an Tripeln. In [Gla+18*] wird gezeigt, dass der Ansatz grundsätzlich auch auf andere Engineering-Artefakte, wie z.B. 3D-CAD-Formate, anwendbar ist.

IV-2.3. Zusammenfassung der Analyse des Stands der Wissenschaft

Die in Abschnitt IV-2.2 dargestellten Methoden sind in Tabelle IV.1 zusammengefasst. Wie der Tabelle zu entnehmen ist, existiert eine Vielzahl von Mapping-Sprachen zur automatisierten ABox-Erstellung für unterschiedliche Datenformate. Darüber hinaus existiert auch eine entsprechende Menge an Software-Applikationen für die Ausführung derartiger Mappings. Die in Abschnitt IV-3 definierte Methode wird die hier dargestellten Ansätze aufgreifen und in die übergeordnete Engineering-Methodik einbetten.

Tabelle IV.1.: Stand der Wissenschaft zur automatisierten ABox-Erstellung

Referenz	**Kriterium**				
	Artefakt	Verfahrensart	Formalismus der Mappings	Software-Lösungen	Nutzungskontext
[OHM10]	CSV, XLS	M	O	OS	GUI
[Blf+17]	CSV, XLS	M	O	OS	E
[W3C12@b]	CSV, XLS, SQL	M/V	S	KO/OS	GUI/E
[Run10]	CAEX	M	-	-	E
[HuHe19]	CAEX/AML	M	-	OS	GUI
[Dim+14]	CSV, XML, JSON	M	O	OS	E
[LZB17]	XML, JSON, CSV, HTML, CBOR	M	O	OS	GUI/E
[Gla+15]	CAEX, A*	M	S*	-	E

„A*“ = Propritäre 3D-CAD-Formate des Werkzeugs „Autodesk Inventor“; „E“ = Einbettung in Applikationscode vorgesehen; „GUI“ = Grafische Benutzeroberfläche; „KO“ = Kommerzielle Software vorhanden; „M“ = Materialisierung; „O“ = offene, nicht-standardisierte Mapping-Sprache; „OS“ = Open-Source Software vorhanden; „S*“ = Abwandlung einer standardisierten Sprache für Regeln; „S“ = standardisierte Mapping-Sprache; „V“ = Virtualisierung; „-“ = nicht vorhanden;

IV-3. Methodik zum werkzeugunterstützten ABox-Engineering

Der vorliegende Abschnitt stellt die Artefakte vor, welche zur Erreichung des Forschungsteilziels III notwendig sind. Hierzu stellt Abschnitt IV-3.1 zunächst den konzeptionellen Ansatz dar, welcher auf der Nutzung semi-automatischer und automatischer Bestandteile beruht. Abschnitt IV-3.2 widmet sich dann einem Methodenteil, welcher die Wahl einer Modellierungsstrategie (automatisch vs. semi-automatisch) unterstützt, bevor Abschnitt IV-3.3 auf die Erstellung eines Werkzeugs zur semi-automatischen ABox-Erstellung eingeht. Abschnitt IV-3.4 ordnet die bisherigen Methodenbestandteile dann in den Kontext des Engineerings von automatisierten Produktionssystemen ein.

IV-3.1. Methodischer Überblick

Der vorliegende Abschnitt führt die ABox-Entwicklungsmethodik ein. Hierzu stellt Abschnitt IV-3.1.1 zunächst die charakteristischen Bestandteile des Konzeptes vor, bevor Abschnitt IV-3.1.2 eine ablauf- und artefaktflussorientierte Sicht auf die Methode beschreibt.

IV-3.1.1. Charakterisierung des ABox-Entwicklungskonzptes

Wie Ausdruck III.1 entnommen werden kann, wird eine Ontologie im Rahmen dieser Arbeit als eine Menge von Tripeln T aufgefasst, wobei für jeden Tripel t gilt, dass er durch ein Subjekt s, ein Prädikat p und ein Objekt o charakterisiert ist. Die drei letzteren werden unter dem Begriff „Entität“ e zusammengefasst. Die Menge aller Tripel T setzt sich dabei aus den zwei Teilmengen der TBox T_{TBox} und der ABox T_{ABox} zusammen. Wie in Abschnitt IV-1 dargestellt, ist die ABox einer Ontologie in der Domäne der Automatisierungstechnik durch zwei Teile charakterisiert, die es methodisch zu berücksichtigen gilt. Dies sind einerseits die quasi-statischen Tripel T_{qs}, welche insbesondere Engineering-Artefakte aus der Systementwicklung repräsentieren, und dynamische Tripel T_d, welche zur Betriebszeit des Systems erzeugte Daten repräsentieren. Es wird angenommen, dass ein Tripel entweder Teil der Menge T_{qs} oder T_d ist und die Schnittmenge von T_{qs} und T_d somit einer leeren Menge entspricht. Ausdruck IV.1 fasst dies zusammen. Wesentliches Charakteristikum der dynamischen Tripel ist, dass sie sich in relativ kurzen Zeitabständen verändern können, während dies für quasi-statische Tripel nicht der Fall ist, da diese i.d.R. nur durch manuellen Einfluss verändert werden, z.B. durch eine Veränderungen mit Hilfe eines Engineering-Tools. Da sie sich dennoch verändern können, ist sowohl die Menge der quasi-statischen Tripel T_{qs} als auch die Menge der dynamischen Tripel T_d von der Zeit x abhängig. Sowohl quasi-statische als auch dynamische Entitäten der Tripel der ABox können mit Hilfe eines Mappings automatisiert erzeugt werden, wie der Abschnitt IV-2 aufgezeigt hat.

$$\begin{aligned} T_{ABox}(x) &= T_{qs}(x) \cup T_d(x) \\ T_{qs}(x) \cap T_d(x) &= \emptyset \end{aligned} \tag{A.IV.1}$$

Wie im Abschnitt IV-1.2.1 dargestellt, werden Daten in Form von Artefakten aus der Planung eines Systems und Daten, welche im Betrieb eines Systems entstehen, durch unterschiedliche Technologien bzw. Formate gespeichert und verfügbar gemacht. Aufgrund der unterschiedlichen Anforderungen im Rahmen der Entwicklung bzw. des Betriebs, werden diese Daten meist auch nicht zentral, sondern auf viele unterschiedliche Dateien und Formate verteilt gespeichert (siehe [Sch18]). Ausdruck IV.2 fasst dies zusammen. Demnach ist ein quasi-statisches Datenobjekt $d_{qs}(x)$ Teil der Menge aller quasi-statischen Datenobjekte $D_{qs}(x)$ und ein dynamisches Datenobjekt $d_d(x)$ Teil der Menge aller quasi-statischen Datenobjekte $D_d(x)$. Auch hier sind die Datenobjekte von der Zeit x abhängig, da alle Datenobjekte grundsätzlich veränderlich sein können. Die Menge der quasi-statischen Datenobjekte $D_{qs}(x)$ und der dynamischen $D_d(x)$ weist keine Schnittmenge auf, da aufgrund unterschiedlicher Anforderungen im Rahmen der Entwicklung bzw. des Betriebs unterschiedliche Speichermedien und Formate vorliegen.

$$\begin{aligned} d_{qs}(x) &\in D_{qs}(x) \\ d_d(x) &\in D_d(x) \\ D_{qs}(x) \cap D_d(x) &= \emptyset \end{aligned} \tag{A.IV.2}$$

Existieren quasi-statische Daten in Artefakten aus der Planungsphase eines Systems, so können diese mit Hilfe eines Mappings automatisiert in eine Tripel-Struktur materialisiert und im Rahmen der ABox-Entwicklung weiter genutzt werden. Dies ist insbesondere dann von Interesse, wenn durch ein Mapping aus einem einzelnen Artefakt lediglich eine Teilmenge der relevanten Tripel erzeugt werden kann und somit eine Verkettung von Mappings unterschiedlicher Artefakte notwendig ist. Dynamische Daten hingegen können im Rahmen der ABox-Entwicklung nicht verfügbar gemacht werden, da sie erst im Rahmen des Betriebs des Systems erzeugt und durch ein Mapping der dynamischen Daten in eine Tripel-Struktur virtualisiert werden. Im Rahmen der ABox-Entwicklung können diese Mappings zwar entwickelt und getestet werden, sie werden jedoch nicht im Rahmen der ABox-Entwicklung eingesetzt. Ausdruck IV.3 fasst dies zusammen. Demnach kann die Menge der quasi-statischen Datenobjekte $D_{qs}(x)$ in eine Menge von quasi-statischen Tripeln $T_{qs}(x)$ mittels eines Materialisierungs- oder Virtualisierungsverfahrens überführt werden. Die Menge der dynamischen Datenobjekte $D_d(x)$ kann mittels eines Virtualisierungsverfahrens in eine Menge von dynamischen Tripeln $T_d(x)$ überführt werden.

$$\begin{aligned} D_{qs}(t) &\xrightarrow{\text{Materialisierung oder Virtualisierung}} T_{qs}(t) \\ D_d(t) &\xrightarrow{\text{Virtualisierung}} T_d(t) \end{aligned} \tag{A.IV.3}$$

Im Rahmen der ABox-Entwicklung soll ein möglichst hoher Automatisierungsgrad angestrebt werden, da dies annahmegemäß am effizientesten ist und somit dem Forschungsziel am dienlichsten ist. Es kann jedoch der Fall auftreten, dass ein automatisierter Mappingansatz nicht in Frage kommt, da der notwendige Aufwand der Erstellung der Mappings nicht in Relation zum erzielten Nutzen steht. Dies kann bspw. durch eine vergleichsweise kleine zu erstellende ABox, nicht erfüllte technologische Bedingungen, oder mangelnde Informationen in existierenden Artefakten begründet sein. Für diesen Fall wird im Rahmen

der vorliegenden Methodik ein semi-automatisierter Ansatz vorgeschlagen, für welchen die nachfolgende Charakterisierung von Interesse ist. Wie in [Skj+18] dargestellt, können im Rahmen der Ontologie-Entwicklung Vorlagen dazu genutzt werden, um wiederkehrende ontologische Strukturen vergleichsweise effizient zu erzeugen. Hierbei werden vordefinierte Tripel mit Variablen versehen und als Software-Methoden modelliert. Dies wurde auch bereits durch [Los+12] erkannt, wobei hier Vorlagen manuell ausgefüllt wurden.

In Anlehnung an [Skj+18] und [Los+12] werden die in Abschnitt III-3.3.1.3 definierten assertionalen Muster aufgegriffen und als Vorlagen genutzt. Diese Vorlagen werden dann mit einer entsprechenden Werkzeugunterstützung versehen, sodass Domänenexperten und Software-Entwickler die variablen Bestandteile der assertionalen Muster in einer geeigneten Benutzerschnittstelle befüllen können, ohne tieferes ontologisches Wissen mitbringen zu müssen. Der Automatisierungsgrad dieses semi-automatischen Ansatzes der ABox-Erstellung ist dann auf das Verhältnis der varianten zu invarianten Entitäten in den assertionalen Mustern limitiert (siehe Abschnitt III-3.3.1.3). Nimmt man beispielsweise an, dass die assertionalen Muster zu zwei Dritteln aus invarianten Entitäten bestehen, also nur eine variable Entität pro Tripel, dann ist der Automatisierungsrad auf 66,6% beschränkt. Das verbleibende Drittel muss dann manuell durch Nutzereingaben bestimmt werden. Das bedeutet auch, das der semi-automatische Ansatz auf die Modellierung quasi-statischer Daten beschränkt ist, wenn man annimmt, dass $T_d(x)$ aufgrund hoher Veränderlichkeit über die Zeit x nicht abschließend bestimmt werden kann.

IV-3.1.2. Methodischer Überblick zur ABox-Entwicklungsmethodik

Wie bereits die ersten beiden Teile der vorliegenden Methodik, basiert auch die ABox-Entwicklung auf einem iterativen, inkrementellen Vorgehen, welches in Abbildung IV.5 dargestellt ist und nachfolgend erläutert wird. Basierend auf den Ergebnissen der TBox-Entwicklung wird zunächst eine ABox-Modellierungsstrategie hergeleitet (siehe Abschnitt IV-3.2). Hierzu wird zunächst der Modellierungsbedarf und die generelle Möglichkeit zur semi-automatischen Erstellung bestimmt. Im Anschluss werden verfügbare Artefakte dahingehend untersucht, inwiefern sie für ein Mapping in Betracht gezogen werden können, unter Beachtung technologischer, modellspezifischer und aufwandsspezifischer Aspekte. Basierend auf dem identifizierten Modellierungsbedarf und der Möglichkeit zur Nutzung automatisierter Mappings wird dann eine Modellierungsstrategie abgeleitet, die darstellt, welche Bestandteile der ABox automatisch und welche semi-automatisch erzeugt werden sollen. Die nächste Phase befasst sich mit der Modellierung der Mappings zur Überführung quasi-statischer und dynamischer Daten in RDF-Strukturen. Im Rahmen der vorliegenden Methodik werden diese Tätigkeiten zwar aufgegriffen, aber aufgrund des breiten Stands der Wissenschaft (siehe Abschnitt IV-2) nicht weiter vertieft. In der dritten Phase wird das im vorherigen Kapitel angesprochene Konzept zur semi-automatischen ABox-Modellierung angewandt (siehe Abschnitt IV-3.3). Hierzu wird die im Rahmen der Arbeit vorgestellte Werkzeug-Architektur zur Erstellung eines entsprechenden Werkzeugs für ein spezifisches ODM anhand des Model-View-Controller-Konzepts genutzt. Die letzte Phase des Vorgehens besteht aus der eigentlichen ABox-Modellierung (siehe Abschnitt IV-3.4). Hierbei werden schrittweise die quasi-statischen Mappings ausgeführt und die somit erzeugte Menge an Tripeln mit Hilfe des in

der vorherigen Phase entworfenen Werkzeug angereichert, sodass alle relevanten Tripel erzeugt werden können.

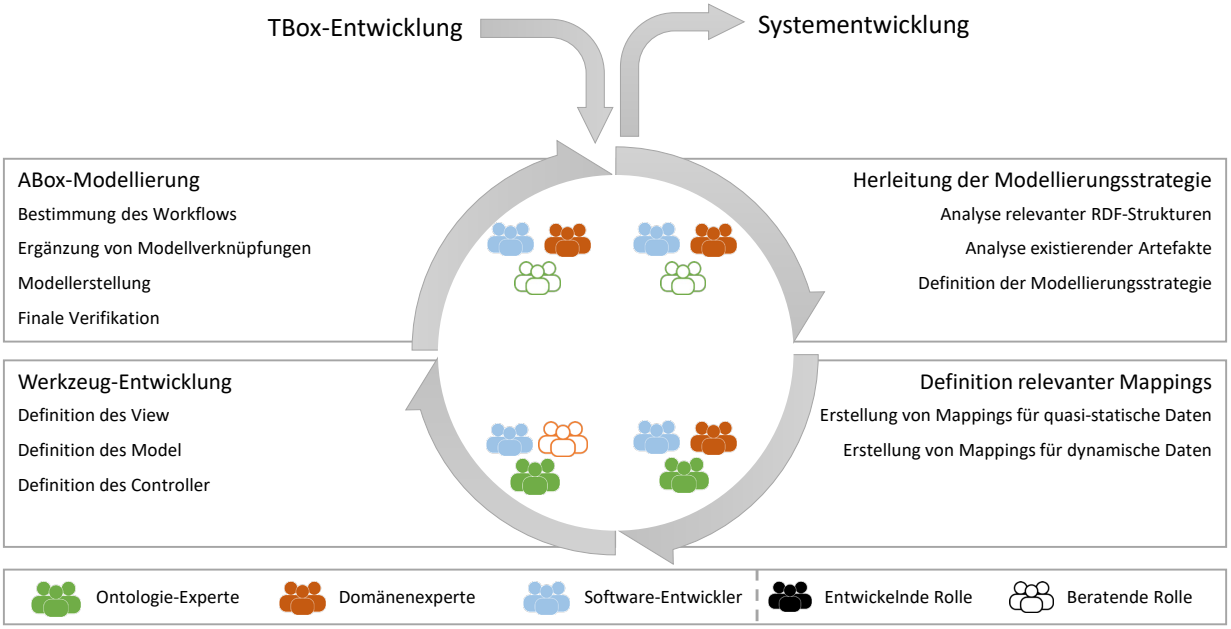

Abbildung IV.5.: Iterativer ABox-Entwicklungsprozess basierend auf Ontologie-Design-Mustern

Betrachtet man den im Rahmen der Methodik auftretenden Artefaktfluss, so lässt sich Abbildung IV.6 heranziehen. Der erste Methodenbaustein befasst sich mit der Herleitung einer Modellierungsstrategie, wobei einerseits existierende Engineering-Artefakte als auch das Ontologie-Anforderungsmodell und die relevante TBox und die darin enthaltenen Ontologie-Design-Muster herangezogen werden. Das Resultat ist eine Modellierungsstrategie, welche das Design der Mappings und des zu entwickelnden Werkzeugs beeinflusst. Im Rahmen der Werkzeugentwicklung kann die in Abschnitt IV-3.3 vorgestellte Architektur wiederverwendet werden, wobei darüber hinaus die projektspezifische TBox und die darin enthaltenen Ontologie-Design-Muster herangezogen werden. Input für den letzten Methodenbaustein sind dann die erstellten Mappings, sowie die dazugehörigen Engineering-Artefakte und das erstellte Software-Werkzeug, wobei das Resultat die Ergebnisse der Gesamtmethodik darstellen.

IV-3.2. Herleitung der Modellierungsstrategie

Im vorliegenden Abschnitt wird das Vorgehen zur Herleitung der Modellierungsstrategie vorgestellt. Hierzu ist es zunächst nötig, die relevanten assertionalen Muster zu analysieren und den Modellierungsbedarf zu bestimmen (siehe IV-3.2.1). Mit Hilfe des Modellierungsbedarfs kann eine Aufwandsschätzung für die semi-automatische Abdeckung des Modellierungsbedarfs durchgeführt werden. Anschließend kann der identifizierte Modellierungsbedarf mit den Möglichkeiten automatisierter Mappings abgeglichen werden (siehe IV-3.2.2), bevor eine Auswahl einer Modellierungsstrategie erfolgt (siehe Abschnitt IV-3.2.3).

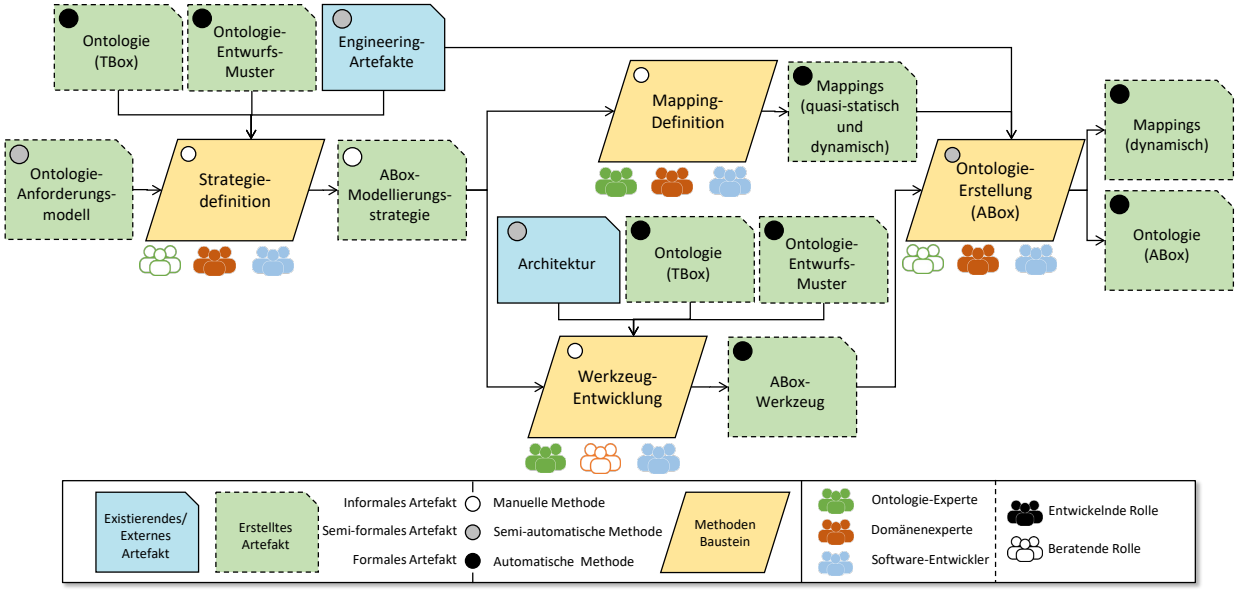

Abbildung IV.6.: Artefaktflussorientierter Überblick zur ABox-Entwicklung der Ontologie

IV-3.2.1. Analyse relevanter RDF-Strukturen

Im Zuge der Analyse der relevanten RDF-Strukturen wird der Modellierungsbedarf im Rahmen eines Projektes näherungsweise bestimmt, indem die Menge der zu modellierenden Tripel bzw. Entitäten im Rahmen eines Projektes geschätzt wird. Die dabei entstehenden Kennzahlen bestimmen maßgeblich die Wahl der Modellierungsstrategie. Je höher die Anzahl der zu modellierenden Tripel in der ABox, desto eher besteht der Bedarf nach einer automatisierten Lösung zur Erstellung der relevanten Tripel. Die nachfolgenden Abschnitte stellen die Bestimmung des Brutto- (siehe Abschnitt IV-3.2.1.1) und Nettomodellierungsbedarfs (siehe Abschnitt IV-3.2.1.2) dar.

IV-3.2.1.1. Bestimmung des Bruttomodellierungsbedarfs

Der Bruttomodellierungsbedarf τ_{Brutto} stellt die Menge der zu modellierenden Entitäten der ABox im Rahmen eines Projektes dar, ohne Berücksichtigung eventueller automatisierter oder semi-automatisierter Ansätze. Um den Bruttomodellierungsbedarf zu bestimmen, werden die im Rahmen der TBox-Entwicklung erstellten Ontologie-Design-Muster (ODMs) herangezogen, für welche assertionale Muster (siehe Abschnitt III-3.3.1.3) erstellt wurden. Unter der Annahme, dass die ABox eines ODMs diesen Mustern folgt, kann der Bruttomodellierungsbedarf hiermit näherungsweise ermittelt werden, da die Menge der Entitäten in der ABox $E(\Lambda_i)$ von den assertionalen Mustern Λ_i eines ODM i abhängig ist. Zusätzlich ist die ABox einer Ontologie von der Zeit x abhängig, da die darin beschrieben Fakten lediglich für einen bestimmten Zeitpunkt gelten. Nimmt man in einer Grenzwertbetrachtung an, dass die ABox unabhängig von der Zeit lediglich aus einer Anzahl an Varianten v_h derselben assertionalen Muster $\lambda_{i,h}$ eines ODM i besteht, kann der zeitliche Faktor durch die Anzahl von Varianten ersetzt werden. Auf diese Weise ist lediglich die Menge an Tripeln der assertionalen Muster zu bestimmen und mit der Anzahl auftretender Varianten zu multiplizieren.

Um die Anzahl der Eingaben bei einer rein manuellen Definition der Tripel zu bestimmen, muss noch der Faktor 3 ergänzt werden, da für jeden Tripel bei einer manuellen Modellierung Subjekt, Prädikat und Objekt einzeln zu bestimmen wären. Diese näherungsweise Betrachtung ist in Formel IV.4 dargestellt. Demnach kann der Bruttomodellierungsbedarf τ_{Brutto} aus einer Grenzwertbetrachtung der von der Zeit abhängigen Menge der Tripel der ABox $T_{ABox}(x)$ bestimmt werden, in welcher die Zeit gegen unendlich geht. Der Faktor drei ist zu berücksichtigen, da bei einem manuellen Modellierungsansatz jede Entität eines Tripels einzeln modelliert werden muss. Da entsprechend Abschnitt III-3.3.1.3 die Menge der Tripel der ABox $T_{ABox}(x)$ durch eine Menge an assertionalen Mustern Λ_i eines ODM i bestimmt werden kann, kann die Grenzwertbetrachtung auch analog auf diese Menge angewandt werden. Näherungsweise kann die Grenzwertbetrachtung durch Einführung einer Anzahl von über die Zeit auftretenden Varianten v_h eines assertionalen Muster h durchgeführt werden. Daher entspricht der Bruttomodellierungsbedarf τ_{Brutto} der Anzahl der Entitäten in einem assertionalen Muster $|E(\lambda_{i,h})|$ multipliziert mit der Anzahl der Varianten v_h für dieses Muster, wobei die Summe über alle ODMs i und alle assertionalen Muster h des ODMs gebildet wird.

$$\tau_{Brutto} = 3 \cdot \lim_{x \to \infty} |T_{ABox}(x)| \cong \lim_{x \to \infty} \sum_{i=0}^{I} |E(\Lambda_i, x)| \cong \sum_{i=0}^{I} \sum_{h=0}^{H} |E(\lambda_{i,h})| \cdot v_h \tag{F.IV.4}$$

Im Beispiel des ODMs der Formalisierten Prozessbeschreibung entsprechend Abschnitt III-4.1 wäre somit für die Varianten v_h lediglich die Anzahl unterschiedlicher, zu beschreibender Prozesse zu bestimmen. Diese Anzahl an Varianten wäre dann zur Bestimmung des Bruttomodellierungsbedarfs mit der Anzahl an Entitäten des assertionalen Musters $|E(\lambda_{i,h})|$ zu multiplizieren. Da, wie im Beispiel der Formalisierten Prozessbeschreibung, eine Bestimmung der Varianten für jedes einzelne assertionale Muster h unter Umständen einen hohen Aufwand erzeugen könnte, kann für die Menge der Entitäten der assertionalen Muster auch ein arithmetischer Mittelwert aus der Anzahl der Entitäten der assertionalen Muster gebildet werden. Wie in den assertionalen Mustern der Formalisierten Prozessbeschreibung in Listing C.1 und C.2 im Anhang C erkennbar, ist die Anzahl der zu modellierenden Entitäten (siehe INSERT-Klausel) im minimalen Muster in Listing C.1 entsprechend $|E(\lambda_{VDI3682,minimal})| = 24$ und im maximalen Muster in Listing C.2 entsprechend $|E(\lambda_{VDI3682,maximal})| = 73$. Der daraus gebildete Mittelwert entspricht 48,5 und wäre nun mit der Anzahl zu modellierender Prozesse $v_{VDI3682}$ zu multiplizieren um den Bruttomodellierungsbedarf näherungsweise zu bestimmen. Dieses vereinfachte Verfahren ist in Formel IV.5 dargestellt.

$$\begin{aligned} |E(\bar{\lambda}_i)| &= \frac{\sum_{h=0}^{H} |E(\lambda_{i,h})|}{H} \\ \tau_{Brutto} = 3 \cdot \lim_{x \to \infty} |T_{ABox}(x)| &\cong \lim_{x \to \infty} \sum_{i=0}^{I} |E(\Lambda_i, x)| \cong \sum_{i=0}^{I} |E(\bar{\lambda}_i)| \cdot v_i \end{aligned} \tag{F.IV.5}$$

IV-3.2.1.2. Bestimmung des Nettomodellierungsbedarfs

Der Bruttomodellierungsbedarf stellt die Menge an Entitäten dar, welche für ein Projekt ohne einen automatisierten oder semi-automatisierten Ansatz zu definieren wären und stellt somit den Modellierungsbedarf einer rein manuellen Modellierung dar. Da, wie in Abschnitt III-3.3.1.3 gezeigt, bei der Nutzung von ODMs lediglich eine Teilmenge dieser Entitäten variabel ist, kann der Nettomodellierungsbedarf τ_{Netto} herangezogen werden, um lediglich die Anzahl der variablen Entitäten des Bruttomodellierungsbedarfs darzustellen. Dies ist in Formel IV.6 dargestellt. Demnach berechnet sich τ_{Netto} aus der Anzahl der variablen Entitäten $|E_{var}(\lambda_{i,h})|$ eines assertionalen Musters $\lambda_{i,h}$ eines ODMs i multipliziert mit der Anzahl an Varianten der assertionalen Muster v_h, wobei die Summe über alle ODMs i und alle assertionalen Muster h gebildet wird.

$$\tau_{Netto} = \sum_{i=0}^{I} \sum_{h=0}^{H} |E_{var}(\lambda_{i,h})| \cdot v_h \tag{F.IV.6}$$

Analog zur Bestimmung des Bruttomodellierungsbedarfs kann auch hier die Berechnung eines arithmetischen Mittelwerts hilfreich sein, um den für die Bestimmung der Varianten notwendigen Aufwand in Grenzen zu halten. Formel IV.7 stellt dies analog zu Formel IV.5 dar. Im Beispiel der Formalisierten Prozessbeschreibung entsprechend Abschnitt III-4.1 kann in der WHERE-Klausel des minimalen Musters (siehe Listing C.1) eine Anzahl variabler Entitäten $|E_{var}(\lambda_{VDI3682,minimal})| = 4$ und in der WHERE-Klausel des maximalen Musters (siehe Listing C.2) $|E_{var}(\lambda_{VDI3682,maximal})| = 9$ festgestellt werden. Daraus ergibt sich ein Mittelwert von 6,5 variablen Entitäten. Dieser Wert würde dann mit der Anzahl der zu modellierenden Prozesse $v_{VDI3682}$ multipliziert und ergäbe den Nettomodellierungsbedarf.

$$\begin{aligned} |\bar{E_{var}}(\lambda_i)| &= \frac{\sum_{h=0}^{H} |E_{var}(\lambda_{i,h}))|}{H} \\ \tau_{Netto} &\cong \sum_{i=0}^{I} |\bar{E_{var}}(\lambda_i)| \cdot v_i \end{aligned} \tag{F.IV.7}$$

IV-3.2.1.3. Bestimmung des Aufwands im semi-automatisierten Fall

Basierend auf den Berechnungen aus Abschnitt IV-3.2.1.1 und IV-3.2.1.2 kann der Automatisierungsgrad berechnet werden, welcher sich aus der Nutzung von ODMs und den zu einem ODM gehörenden assertionalen Mustern ergibt. Dies ist in Formel IV.8 dargestellt. Demnach ergibt sich der Automatisierungsgrad η aus dem Verhältnis von Nettomodellierungsbedarf τ_{Netto} und dem Bruttomodellierungsbedarf τ_{Brutto}. Basierend auf den bisher errechneten Beispielen für die Formalisierte Prozessbeschreibung ergibt sich ein Automatisierungsgrad von 86,6%, mit $\tau_{Brutto} = 48,5$ und $\tau_{Netto} = 6,5$. Es sei darauf hingewiesen, dass bei der Herleitung der Brutto- bzw. Nettomodellierungsbedarfe eine konsistente Verwendung (oder nicht-Verwendung) von arithmetischen Mittelwerten vorzusehen ist.

$$\eta = 1 - \frac{\tau_{Netto}}{\tau_{Brutto}} \tag{F.IV.8}$$

Der in Formel IV.8 dargestellte Automatisierungsgrad bezieht sich auf die durch das assertionale Muster bestimmte Möglichkeit zur Automatisierung im Falle eines semi-automatisierten Vorgehens. Im semi-automatisierten Vorgehen werden, wie in Abschnitt IV-3.1.1 angeführt, die assertionalen Muster im Sinne von Vorlagen mit Hilfe von Nutzereingaben ausgefüllt. Demnach entspricht die Anzahl der variablen Entitäten im Nettomodellierungsbedarf τ_{Netto} in etwa der Anzahl der Nutzereingaben. Somit lässt sich der zeitliche Aufwand mit Hilfe zweier zusätzlicher Variablen aus den bisherigen Berechnungen bestimmen. Dies ist in Formel IV.9 dargestellt. Demnach wird der zeitliche Aufwand Z_s im semi-automatischen Fall s aus drei Variablen berechnet. Neben dem in Abschnitt IV-3.2.1.2 bestimmten Nettomodellierungsbedarf τ_{Netto} ist die Definition eines zeitlichen Aufwands pro Nutzereingabe z_u notwendig. Diese ist von diversen Faktoren abhängig, wie z.B. Gestaltung der Benutzerschnittstelle, Erfahrung der Nutzer, Performance des Werkzeugs bei der Modellierung etc. Es sei an dieser Stelle angenommen, dass ein entsprechender Wert in Projekten als Erfahrungswert ermittelt werden kann. Darüber hinaus ist eine Zeit anzunehmen, welche für die Erstellung des Werkzeugs Z_w aufgewendet werden muss, sofern noch kein Werkzeug existiert. Dieser Wert ist ebenfalls von diversen Faktoren abhängig (z.B. Erfahrung der Systementwickler, siehe auch Abschnitt IV-3.3), wobei auch hier gelten soll, dass ein ein entsprechender Wert in Projekten als Erfahrungswert ermittelt werden kann. Für das laufende Beispiel der Formalisierten Prozessbeschreibung könnte somit bei 10 zu modellierenden Prozessen und einer Zeit pro Nutzereingabe von $z_u = 30s$ eine zeitlicher Aufwand von $Z_s = 1950s$, bzw. 32,5 Minuten berechnet werden. Dies unter der Annahme, dass ein Werkzeug bereits existiert und $Z_w = 0$ sei.

$$Z_s = \tau_{Netto} \cdot z_u + Z_w = \tau_{Brutto} \cdot (1 - \eta) \cdot z_u + Z_w \tag{F.IV.9}$$

IV-3.2.2. Analyse existierender Artefakte

In diesem Abschnitt wird eine Metrik zur Analyse existierender Artefakte vorgestellt, welche auf drei Teilen beruht und Aussagen über die Vorteilhaftigkeit eines automatisierten Mappings erlaubt. Die drei Teile sind wie folgt:

- Technologieorientierte Bewertung des Artefakts: Im Rahmen der technologischen Bewertung wird ein Artefakt hinsichtlich der technologischen Eignung für ein automatisiertes Mapping analysiert.

- Bewertung der Anwendbarkeit und Skalierbarkeit möglicher Mappings: Im Rahmen der Bewertung existierender Mappings wird analysiert, ob es existierende Mapping-Sprachen und Mapping-Engines für eine bestimmte Art von Artefakt gibt. Darüber hinaus werden einige subjektive (abhängig von Fähigkeiten des Projektteams) Kriterien ergänzt.

- Modellspezifische Bewertung des Artefakts: Im Rahmen der modellspezifischen Bewertung wird das Artefakt dahingehend analysiert, inwiefern es die notwendigen Datenobjekte zur Erzeugung der relevanten RDF-Strukturen liefern kann und wie hoch der Anteil an nicht-relevanten Datenobjekten im Artefakt ist.

Nachfolgend wird entsprechend Ausdruck IV.10 unterschieden in Klassen k bis K von Artefakten. Eine Artefaktinstanz a_k ist Teil der Menge aller Artefakte A_k einer Artefaktklasse k. Dabei kann es sich bei einer Artefaktklasse k z.B. um AutomationML, CAEX, 3D-CAD, SQL, oder OPC UA Informationsmodelle handeln. Ein bestimmtes AutomationML-Modell wäre dann bspw. $a_{AutomationML}$ und die Menge aller AutomationML-Modelle in einem Projekt wäre $A_{AutomationML}$.

$$\begin{gathered} k...K \\ a_k \in A_k \end{gathered} \tag{A.IV.10}$$

Für ein Datenobjekt einer Artefaktinstanz d_{a_k} gilt nachfolgend, dass es Bestandteil der Menge aller Datenobjekte D_{a_k} der Artefaktinstanzen einer Artefaktklasse a_k ist. Darüber hinaus kann die Menge relevanter Datenobjekte D_{r,a_k} genutzt werden, um die Menge der varianten Entitäten des ABox-Modellierungsbedarf E_{var} zu erstellen. Dies geschieht durch Definition und Ausführung eines Mappings, welches die relevanten Datenobjekte D_{r,a_k} in die varianten Entitäten E_{var} überführt. Für die Menge relevanter Datenobjekte D_{r,a_k} gilt, dass sie eine Teilmenge aller Datenobjekte D_{a_k} einer Artefaktinstanz a_k sind. Der beschriebene Sachverhalt ist in Ausdruck IV.11 dargestellt. Es werden im Gegensatz zur Bestimmung des Modellierungsbedarfs keine Varianten berücksichtigt, da diese durch Ausführung eines Mappings, je nach Gestalt des Datenobjektes, entstehen.

$$\begin{gathered} d_{a_k} \in D_{a_k} \\ D_{r,a_k} \xrightarrow{\text{Mapping}} E_{var} \\ D_{r,a_k} \subseteq D_{a_k} \end{gathered} \tag{A.IV.11}$$

IV-3.2.2.1. Technologieorientierte Bewertung des Artefakts

Die nachfolgend beschriebene technologieorientierte Bewertung der Artefakte basiert auf den Kriterien des Reifegradmodells nach [Pun+16] und wurde mit der Offenheitsmetik aus [Bar+12] kombiniert und auf den vorliegenden Anwendungsfall angepasst. Die nachfolgende Betrachtung bezieht sich dabei stets auf eine Menge von Artefakten A_k der Klasse k. Ziel der Anwendung der Metrik ist die Berechnung eines Faktors zur Bewertung der technologischen Eignung A_F einer Artefaktklasse für ein Mapping. Hierzu werden für die zwei Kategorien: Lesbarkeit der Inhalte des Artefakts A_{Le} sowie Einsetzbarkeit A_E Punktwerte ermittelt. Diese Punktwerte werden mit der maximal erreichbaren Punktzahl FA_{max} normiert, um den Faktor A_F zu bestimmen, siehe Formel IV.12. Die Kriterien der Metrik sind in Tabelle IV.2 zu finden und werden nachfolgend erläutert. Der jeweilige Punktwert der Kriterien der Metrik (1, wenn zutreffend) wird zur entsprechenden Kategorie aufsummiert und kann je nach Fähigkeiten des Projektteams angepasst werden, es sei allerdings darauf hingewiesen, dass eine Vergleichbarkeit zwischen Projekten lediglich bei gleichen Punktwerten und Faktorgewichtungen möglich ist.

$$A_F = \frac{A_{Le} + A_E}{FA_{max}} \tag{F.IV.12}$$

Die Kriterien der Lesbarkeit wurden aus [Pun+16] übernommen und beziehen sich auf die Lesbarkeit der Inhalte des Artefakts. Hierbei wird einerseits die Menschen- aber auch die Maschinenlesbarkeit betrachtet, wie auch die Form und Interpretation der Datenobjekte. Hinsichtlich der Unterscheidung von Maschinen- und Menschenlesbarkeit wird die Trennung eingeführt, dass die Artefakte dann maschinenlesbar sind, wenn es eine existierende Bibliothek zum analysieren (englisch: „parsen“) der Daten gibt, wie z.B. für XML-Dokumente. Auf die Bewertung der Granularität wird mangels der Übertragbarkeit auf Artefakte außerhalb des 3D-CAD-Bereichs verzichtet. Die Kriterien der Kategorie „Einsetzbarkeit“ wurden basierend auf der Offenheitsmetrik entsprechend [Bar+12] eingefügt und angepasst.

Tabelle IV.2.: Kriterien zur Bestimmung der technologischen Eignung

Kategorie	Kriterium	Beschreibung
Lesbarkeit A_{Le}	Menschliche Lesbarkeit	Die Daten sind durch einen Menschen lesbar/-auswertbar.
	Standardisierte Form	Die standardisierte Form bezeichnet die strukturelle Konsistenz der Datenobjekte. Sie sind bezüglich des Aufbaus formal einheitlich. Formal einheitliche Datenobjekte haben ein standardisiertes Format und sind einheitlich dargestellt. Sie sind dazu durchgängig in einer gleichbleibenden Form aufgebaut.
	Eindeutige Auslegbarkeit	Die Datenobjekte sind inhaltlich eindeutig interpretierbar. Aus ihnen erschließt sich, was mit den Werten ausgesagt werden soll.
	Maschinenlesbarkeit	Die Datenobjekte sind maschinell lesbar/auswertbar mit Hilfe einer existierenden Bibliothek.
	Auswertbarkeit trotz fehlender Werte	Nicht vorhandene Werte mindern zwar den Informationsgehalt, verhindern aber nicht die Verarbeitung der Daten.
Einsetzbarkeit A_E	Identifizierbarkeit	Datenobjekte sind durch einen eindeutigen Bezeichner identifizierbar.
	Typisierung von Datenobjekten	Die Bedeutung von Datenobjekten ist durch eine entsprechende Typisierung auswertbar.
	Objektivität	Die Typisierung wird objektiv einheitlich genutzt. Sie sind weitestgehend frei von subjektiven Einflüssen und inhaltlich neutral dargestellt.
	Anpassbarkeit	Bibliotheken, wie z.B. die zur Typisierung, können exportiert und angepasst werden.
	Fehlerfreiheit	Die Fehlerfreiheit beschreibt die Korrektheit der Inhalte des Artefakts. Es ist sichergestellt, dass die Fehlerfreiheit automatisch überprüft werden kann.

Neben der Ermittlung eines Punktwertes erlauben die vorliegenden Kriterien auch die Ermittlung eines Reifegrades für das Artefakt, analog zum Vorgehen in [Pun+16]. Das aus den Kriterien hervorgehende Reifegradmodell ist in Abbildung IV.7 zu finden. Auf diese Weise

lässt sich auch festlegen, welche Kriterien ein Artefakt mindestens erfüllen muss, um aus technologischer Sicht für einen automatisierten Ansatz in Frage zu kommen. Hierfür wird der Reifegrad „RG1“ festgelegt. Dies inkludiert die Verfügbarkeit, menschliche Lesbarkeit und Identifizierbarkeit von Datenobjekten. Die Verfügbarkeit beschreibt die Zugänglichkeit der Artefakte. Artefakte sind zugänglich, sofern sie zur Bearbeitung vorliegen, wobei eine fehlende Verfügbarkeit technisch, rechtlich oder organisatorisch begründet sein kann [Pun+16]. Da bei einem Mapping die Datenobjekte des Artefakts ausgelesen werden müssen, muss mindestens eine menschliche Lesbarkeit vorliegen, sodass eine entsprechende Software zu deren Auswertung geschrieben werden kann. Das Vorliegen einer Bibliothek für das Auslesen des Artefakts (maschinelle Lesbarkeit) ist nicht zwingend nötig, auch wenn es den Aufwand erheblich senken kann. Da bei der Überführung im Rahmen eines Mappings darüber hinaus einzelne Datenobjekte unterscheidbar sein müssen, um sie gezielt zu überführen, ist die Identifizierbarkeit ein notwendiges Kriterium.

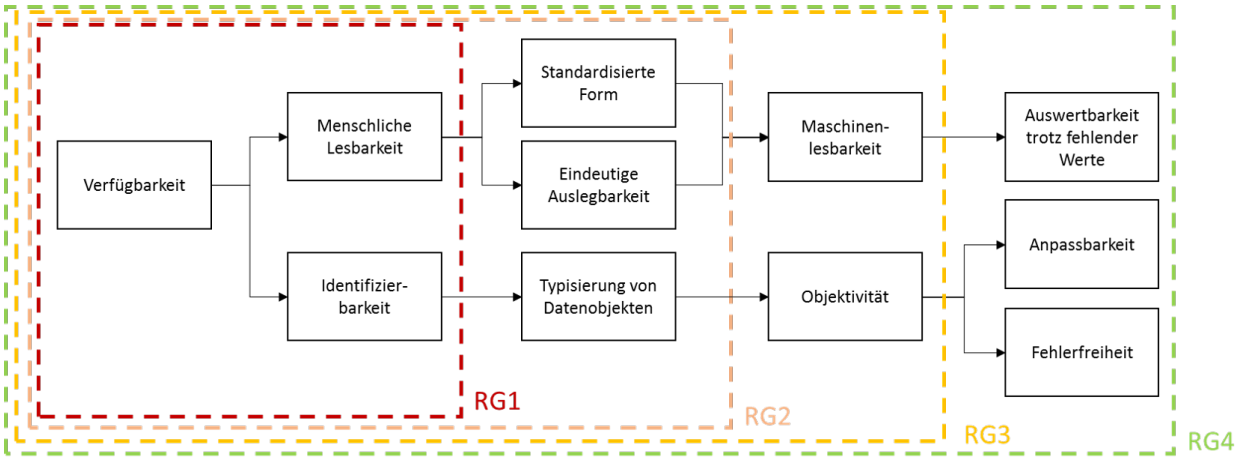

Abbildung IV.7.: Reifegradmodell für die Bestimmung der technologischen Eignung von Artefakten

IV-3.2.2.2. Anwendbarkeit und Skalierbarkeit möglicher Mappings

Um zu bestimmen, inwiefern ein Mapping für die Überführung der Datenobjekte einer Artefaktklasse in eine RDF-Struktur grundsätzlich in Frage kommt, ist dessen generelle Anwendbarkeit und Skalierbarkeit zu bestimmen. Für eine Bewertung der Anwendbarkeit existierender Mappings M_F werden drei Kategorien betrachtet: die Existenz bzw. Standardisierung einer Mapping-Sprache M_L, Nutzbarkeit existierender Mapping-Engines M_E und die vorhandene Erfahrung im Umgang mit einem Mappingansatz im entsprechenden Projektteam M_S. Die drei Kategorien orientieren sich an den Wissensfragen aus Abschnitt IV-2.1 und sind in Tabelle IV.3 zu finden. Formel IV.13 setzt die drei Kategorien miteinander in Beziehung. Demnach ergibt sich die Anwendbarkeit existierender Mappings M_F aus der Summe der Punktwerte der drei genannten Kategorien M_L, M_E und M_S, wobei diese Summe durch die maximal erreichbare Punktzahl FM_{max} normiert wird.

$$M_F = \frac{M_L + M_E + M_S}{FM_{max}} \tag{F.IV.13}$$

Kriterien der ersten Kategorie betrachten die Existenz bzw. Standardisierung einer Mapping-Sprache M_L für eine bestimmte Klasse von Artefakten. Die grundsätzliche Existenz einer Mapping-Sprache ist dahingehend relevant für die Erstellung von Mappings, als dass ein wiederverwendbarer Formalismus für die Definition des Mappings existiert. Dabei können Mapping-Sprachen einerseits standardisiert sein (z.B. der W3C Standard R2RML entsprechend [W3C12@b]), offen aber nicht standardisiert (z.B. Mapping-Master entsprechend [OHM10]) oder inexistent sein. Die höchste Reife hinsichtlich der Fähigkeiten und zugehöriger Dokumentation einer Mapping-Sprache ist annahmegemäß bei einer Standardisierung erreicht, weshalb hierfür die höchste Punktzahl (2) anzunehmen ist. Sollte der Fall eintreten, dass kein Mapping existiert, so soll gelten: $M_L = 0 = M_F$.

Die zweite Kategorie betrachtet existierende Mapping-Engines M_E. Die grundsätzliche Existenz einer Mapping-Engine ist dahingehend relevant, als das eine wiederverwendbare Software-Lösung für die Ausführung von erstellten Mappings existiert und nicht im Rahmen eines Projektes erstellt werden muss. Dabei wird unterschieden in kommerzielle Lösungen, nicht-kommerzielle Open-Source Lösungen, sowie die nicht-Existenz einer Mapping-Engine. Unter der Annahme, dass eine kommerzielle Lösung einer nicht-kommerziellen Lösung aufgrund von vorhandenem Support vorgezogen wird, erhält die kommerzielle Lösung die höhere Punktzahl.

Die dritte Kategorie betrachtet die vorhandenen Erfahrung im Umgang mit einem Mappingansatz im entsprechenden Projektteam M_S. Der Aspekt ist somit subjektiv zu bestimmen. Der Nutzung dieses Aspekts liegt die Annahme zugrunde, dass der Aufwand zur Nutzung umso geringer ist, je größer die Erfahrung im Umgang mit dem Mappingansatz ist.

Tabelle IV.3.: Kriterien zur Bestimmung der Anwendbarkeit von Mappingansätzen

Kategorie	Kriterium	Punktzahl
Mapping-Sprache M_L	Standardisierte Mapping-Sprache vorhanden	2
	Offene, nicht standardisierte Mappingsprache vorhanden	1
	Keine Mappingsprache vorhanden	0
Mapping-Engine M_E	Kommerzielle Lösung vorhanden	2
	Nicht-kommerzielle Open-Source Lösung vorhanden	1
	Keine Software-technsiche Lösung vorhanden	0
Erfahrung M_S	Fundierte Erfahrung im Umgang mit Mapping- Sprache und/oder Engines vorhanden	2
	Erste Erfahrungen im Umgang mit Mapping- Sprache und/oder Engines vorhanden	1
	Keine Erfahrung vorhanden	0

Auf Basis der Kriterien der Tabelle IV.3, können die in der Literatur existierenden Ansätze bewertet werden, was für die in Abschnitt IV-2 dargestellten Ansätze zu Tabelle IV.4 führt. Dort dargestellt ist der jeweilige Wert für M_F eines Mappingverfahren der Literaturanalyse, ohne den subjektiven Anteil M_S. Diese Tabelle kann in einem Projekt als Basis herangezogen werden und um projektspezifische Recherchen erweitert werden.

Tabelle IV.4.: Bewertung automatisierter Mappingverfahren

Referenz	Format						
	CSV	XLS	SQL	XML	JSON	CAEX, AML	3D CAD
[OHM10]	2	2	-	-	-	-	-
[Blf+17]	2	2	-	-	-	-	-
[W3C12@b]*	4	4	4	-	-	-	-
[Run10]	-	-	-	-	-	0	-
[HuHe19]	-	-	-	-	-	1	-
[Dim+14]	2	-	-	2	2	-	-
[LZB17]	2	-	-	2	2	-	-
[Gla+15]	-	-	-	-	-	0	0

„*“ = Nutzbar für Virtualisierung

Neben der Bewertung der Anwendbarkeit eines Mappingverfahrens im Rahmen eines Projektes ist darüber hinaus die Skalierbarkeit ρ_{A_k} eines erstellten Mappings für alle Artefaktinstanzen A_k der Artefaktklasse k von Interesse, da ein Mapping im Optimalfall nur einmal für eine Klasse von Artefakten erstellt werden muss. In diesem Fall ist eine Überführung aller Artefaktinstanzen mit Hilfe eines Mappings möglich. In der Praxis kann jedoch der Fall auftreten, dass innerhalb einer Klasse von Artefakten gewisse Unterschiede im verwendeten Schema existieren, was sich in einer unterschiedlichen Strukturierung der gleichen Klasse von Artefakten äußert. Dies ist beispielsweise bei AutomationML-Artefakten der Fall, welche unterschiedliche Rollen- und Klassenbibliotheken nutzen, oder eine unterschiedliche Strukturierung der Instance-Hierarchie. Soll für die Klasse von AutomationML-Artefakten dann ein Mapping erstellt werden, muss ein Mapping für jedes unterschiedliche AutomationML-Modell erstellt werden [Gla+15]. Das gleiche Problem ergibt sich auch bei anderen Klassen von Artefakten, wie z.B. MS-Excel-Dokumenten [OHM10], und muss daher systematisch berücksichtigt werden.

Hierzu wird zunächst die Anzahl unterschiedlicher, gleichzeitig existierender Schemata einer Artefaktklasse $\check{s}_{A_k}$ ermittelt, welche darstellt, welche Heterogenität hinsichtlich der Struktur zwischen den Instanzen einer Artefaktklasse existiert. Da die den Artefakten zugrundeliegenden Schemata grundsätzlich einer Versionierung unterworfen sein können, welche eine Überarbeitung eines Mappings erfordern kann, ist die Versionierung als zusätzlich beeinflussender Faktor zu berücksichtigen. Dabei wird angenommen, dass für ein Schema s zu einem Zeitpunkt x eine bestimmte Version $\check{v}_{s,x}$ vorliegt, welche Bestandteil der Menge der Versionen $\check{V}_{s,X}$ über einen bestimmten Zeitraum X ist. Um die Menge der Versionen zu bestimmen, kann entweder ein historischer oder ein zukünftiger Zeitraum herangezogen werden. Die mittlere Versionshäufigkeit $\check{\bar{v}}_{A_{k,X}}$ der Schemata einer Artefaktklasse $A_{k,X}$ über den Zeitraum X kann dann bestimmt werden, siehe Formel IV.14. Demnach ergibt sich die mittlere Versionshäufigkeit $\check{\bar{v}}_{A_{k,X}}$ aus der der Anzahl von Versionen $|\check{V}_{s,X}|$ über den Zeitraum X, wobei die Summe über alle Schemata s bis $\check{s}_{A_k}$ gebildet und mit der Anzahl gleichzeitig existierender Schemata einer Artefaktklasse $\check{s}_{A_k}$ normiert wird. Existieren bspw. für die Artefaktklasse $k = AutomationML$ gleichzeitig drei unterschiedliche Schemata, so wäre $\check{s}_{A_{AutomationML}} = 3$. Nimmt man darüber hinaus an, dass eines der drei Schemta über einen Zeitraum von $X = 1Jahr$ drei Versionen aufweisen wird, so wäre $|\check{V}_{s=1,X}| = 3$. Sofern für die verbleibenden zwei Schemata keine Versionsänderungen im genannten Zeitraum angedacht

sind gilt $|\check{V}_{s=2,X}| = 1$ und $|\check{V}_{s=3,X}| = 1$. Daraus resultiert eine mittlere Versionshäufigkeit $\check{\bar{v}}_{A_{k,X}} = 1,667$.

$$\check{\bar{v}}_{A_{k,X}} = \frac{\sum_{s=1}^{\check{s}_{A_k}} |\check{V}_{s,X}|}{\check{s}_{A_k}} \; ; \; mit \; \check{v}_{s,x} \in \check{V}_{s,X} \tag{F.IV.14}$$

Mit Hilfe der dargestellten Berechnungsbestandteile kann abschließend der Skalierungsfaktor ρ_{A_k} bestimmt werden. Dieser ist der Kehrwert der Anzahl der zu erstellenden Mappings pro Artefaktklasse A_k und ist in Formel IV.15 dargestellt. Der Skalierungsfaktor sinkt demnach mit der Anzahl zu erstellender Mappings pro Artefaktklasse und ist im Optimalfall 1. Dies trifft zu, wenn es exakt ein Schema $\check{s}_{A_k}$ einer Artefaktklasse A_k gibt, welches keiner Versionierung unterworfen ist. Mit den Daten des obigen Beispiels wären fünf Mappings zu entwerfen, da drei Schemata $\check{s}_{A_{AutomationML}} = 3$ und eine mittlere Versionshäufigkeit $\check{\bar{v}}_{A_{k,T}} = 1,667$ vorliegen. Daraus ergäbe sich ein Skalierungsfaktor $\rho_{A_k} = 0,2$.

$$\rho_{A_k} = \frac{1}{\check{s}_{A_k}} \cdot \frac{1}{\check{\bar{v}}_{A_{k,t}}} \tag{F.IV.15}$$

IV-3.2.2.3. Modellspezifische Bewertung des Artefakts

Für die modellspezifische Bewertung sind zweierlei Aspekte von Interesse. Der erste Aspekt repräsentiert den Anteil am Nettomodellierungsbedarf (siehe auch Formel IV.6) τ_{Netto}, der durch eine Menge von Artefakten A_k einer Artefaktklasse k abgedeckt werden kann. Insbesondere bei einem Vergleich zweier Artefaktklassen, welche grundsätzlich die gleichen Inhalte beschreiben (z.B. Prozesse) ist diese Kennziffer von Relevanz. Zur Berechnung dieses relevanten Anteils $R_{A_k \rightarrow \tau_{Netto}}$ ist zunächst der Betrag der Menge der relevanten Datenobjekte $|D_{r,A_k}|$ zu bestimmen, welche in allen Instanzen einer Artefaktklasse zu finden sind. Dieser Betrag ist mit der Anzahl der auftretenden Varianten v_h eines assertionalen Muster h zu multiplizieren, welche in Abschnitt IV-3.2.1.1 angenommen wurden. Die Berechnung ist in Formel IV.16 dargestellt. Enthält eine Artefaktklasse $k = AutomationML$ bspw. eine Prozessbeschreibung, aus welcher variante Entitäten der assertionalen Muster der Formalisierten Prozessbeschreibung (siehe WHERE-Klausel der Listings C.1 und C.2) durch Mapping erstellt werden können, dann kann der automatisierbare Anteil $R_{A_k \rightarrow \tau_{Netto}}$ des Nettomodellierungsbedarfs wie folgt bestimmt werden. Zunächst wird bestimmt, welche Datenobjekte der AutomationML-Artefakte welche varianten Entitäten der assertionalen Muster bestimmen können. Hierzu wird die WHERE-Klausel betrachtet. Können bspw. fünf Entitäten der WHERE-Klausel mittels eines Mappings bestimmt werden, so sind die korrespondierenden Datenpunkte gleich der Menge der relevanten Datenpunkte $D_{r,A_{AutomationML}}$. Der Betrag $|D_{r,A_{AutomationML}}|$ wäre somit fünf. Berücksichtigt man, dass 10 Prozesse im Beispiel in Abschnitt IV-3.2.1.3 angenommen wurden und diese 10 Prozesse auch in Instanzen $a_{AutomationML}$ zu finden sind, dann ergibt sich mit dem Beispiel des Nettomodellierungsbedarf $\tau_{Netto} = 65$ ein relevanter Anteil von $R_{A_k \rightarrow \tau_{Netto}} = 0,77$.

$$R_{A_k \to \tau_{Netto}} = \frac{|D_{r,A_k}| \cdot \sum_{h=0}^{H} v_h}{\tau_{Netto}} = \frac{\sum_{a_k=0}^{A_k} |D_{r,a_k}| \cdot \sum_{h=0}^{H} v_h}{\tau_{Netto}} \tag{F.IV.16}$$

Der zweite Aspekt betrachtet den Overhead an Datenobjekten D_{o,A_k} in Instanzen einer Artefaktklasse, welche für die Modellierung der ABox keine Relevanz besitzen und daher die Erstellung der Mappings erschweren. Zur Bestimmung des Overheads ist zunächst die mittlere Anzahl an Datenobjekten der Instanzen einer Artefaktklasse $|\bar{D}_{A_k}|$ zu bestimmen. Hierzu wird die Anzahl der Datenobjekte der jeweiligen Artefaktinstanzen aufsummiert und durch die Anzahl der Artefaktinstanzen $|A_k|$ geteilt. Anschließend kann der Overhead D_{o,A_k} mit Hilfe der vorher bestimmten Menge der relevanten Datenobjekte $|D_{r,A_k}|$ einer Artefaktklasse bestimmt werden. Formel IV.17 stellt dies dar. Basierend auf dem oben genannten Beispiel der relevante Datenpunkte $|D_{r,A_{AutomationML}}| = 5$ und der Annahme, die durchschnittliche Anzahl an Datenpunkte wäre $|D_{A_{AutomationML}}| = 100$, dann wäre der Overhead $D_{o,A_k} = 0,95$.

$$\begin{aligned} |\bar{D}_{A_k}| &= \frac{\sum_{a_k=0}^{A_k} |D_{a_k}|}{|A_k|} \\ D_{o,A_k} &= 1 - \frac{|D_{r,A_k}|}{|\bar{D}_{A_k}|} \end{aligned} \tag{F.IV.17}$$

IV-3.2.2.4. Abschließende Bewertung existierender Artefakte

Für eine abschließende Bewertung eines automatisierten Ansatzes werden die bisher erarbeiteten Kriterien in Beziehung gesetzt und in einem Faktor γ zusammengefasst. Hierbei können die in den vorherigen Abschnitten definierten Kriterien noch gewichtet werden. Formel IV.18 stellt dies dar. Je näher der Wert von γ an 1, desto besser ist das Artefakt für ein automatisiertes Verfahren geeignet. Dieser Fall tritt ein, wenn der Skalierungsfaktor ρ_{A_k} gleich eins ist, kein Overhead an Datenobjekten D_{o,A_k} vorliegt und die Kriterien A_F und M_F die Bestbewertung „1“ zugewiesen bekommen haben. Wie der Formel zu entnehmen ist, können die Kriterien D_{o,A_k}, A_F und M_F noch mit Gewichten g_{1-3} versehen werden. Jegliche Abweichung vom Optimalfall lässt den Faktor γ über eins ansteigen.

$$\gamma = \frac{\sum g}{g_1 \cdot (1 - D_{o,A_k}) + g_2 \cdot M_F + g_3 \cdot A_F} \cdot \frac{1}{\rho_{A_k}} \tag{F.IV.18}$$

Spezialfälle können eine mangelnde Eignung der Artefaktklasse an sich, als auch das nicht-Vorhandensein geeigneter Mapping-Sprachen bzw. Mapping-Engines sein. Je nachdem, inwiefern in einem Projekt Ressourcen für die Optimierung der Artefakte oder die Entwicklung von Mapping-Sprachen und Mapping-Engines bereitstehen, müssen diese Fälle gesondert betrachtet werden. Sind keine Ressourcen vorhanden, führt dies dazu, dass ein automatisierter Ansatz ausgeschlossen wird: $M_F = 0 = \gamma$ bzw. $A_F = 0 = \gamma$.

Für eine Abschätzung des zeitlichen Aufwands für die Erstellung von Mappings Z_M aller relevanten Datenobjekte einer Artefaktklasse $|D_{r,A_k}|$ ist zunächst die mittlere Zeit für das

Mapping eines einzelnen Datenobjektes z_{a_k} einer Instanz einer Artefaktklasse zu bestimmen. Analog zur Bestimmung des Faktors im semi-automatischen Vorgehen, ist dieser Faktor projektspezifisch basierend auf den Fähigkeiten der betroffenen Stakeholder (Domänenexperten, Software-Entwickler und Ontologieexperten) zu bestimmen. Hieraus kann mit Hilfe von Formel IV.19 der zeitliche Aufwand bei Nutzung eines automatisierten Ansatzes bestimmt werden. Hieran wird ersichtlich, dass der Wert von γ den Aufwand negativ beeinflusst. Dies ist darin begründet, dass geringe Punktzahlen die im Rahmen der Metrik bestimmt wurden, den Aufwand tendenziell erhöhen. So ist bspw. bei einer geringen technologischen Eignung davon auszugehen, dass die Zeit für die Erstellung eines Mappings für ein einzelnes Datenobjekt z_{a_k} entsprechend höher ausfällt. Es sei jedoch darauf hingewiesen, dass die Genauigkeit der Berechnung stark davon abhängig ist, wie die Gewichtungen der einzelnen Bereiche und der Gesamtwertung in Formel IV.18 vorgenommen werden. Es ist daher davon auszugehen, dass die Metrik erst bei mehrmaligem Einsatz und entsprechender, erfahrungsbasierter Feinabstimmung der Gewichte eine entsprechende Genauigkeit hinsichtlich der Aufwandsabschätzung gewinnt.

$$Z_M = z_{a_k} \cdot |D_{r,A_k}| \cdot \gamma \tag{F.IV.19}$$

Abschließend wird der Anteil an variablen Entitäten E_{var} bestimmt, welcher maximal durch einen automatisierten Ansatz erreicht werden kann. Hierzu werden alle Instanzen a_k aller Artefaktklassen k bis K berücksichtigt. Die Berechnung ist in Formel IV.20 zu finden und wird analog zu Formel IV.16 durchgeführt. Wesentlicher Unterschied ist jedoch, dass nicht ein einzelnes Artefakt betrachtet wird, sondern alle Artefakte und auch alle variablen Entitäten E_{var}. Dieser Wert wird nachfolgend als Mappingpotential bezeichnet und wird für die Herleitung der Modellierungsstrategie benötigt.

$$R_{A \rightarrow E_{var}} = \frac{\sum_{k=0}^{k} \sum_{a_k=0}^{A_k} |D_{r,a_k}|}{E_{var}} \tag{F.IV.20}$$

IV-3.2.3. Definition der Modellierungsstrategie

Im Rahmen dieser Arbeit werden drei Arten von Modellierungsstrategien berücksichtigt, siehe Abschnitt IV-3.1. Dies ist erstens der semi-automatische Ansatz, welcher darauf beruht, dass die varianten Entitäten der ABox durch Nutzereingaben bestimmt werden. Zweitens der automatisierte Ansatz, bei welchem die varianten Entitäten der ABox durch die Ausführung von Mappings bestimmt werden, welche Datenobjekte aus Artefakten in RDF-Strukturen überführen. Drittens besteht die Möglichkeit einer hybriden Strategie, deren Automatisierungsgrad über dem ersten Ansatz, aber unter dem zweiten Ansatz liegt.

Nachfolgend wird eine Fallunterscheidung (siehe Formel IV.21) anhand des Mappingpotentials aus Formel IV.20 erläutert, welches die relevanten Datenobjekten über alle Artefaktinstanzen und Artefaktklassen ins Verhältnis zur Menge an varianten Entitäten setzt. Im ersten Fall (F1) ist das Mappingpotential $R_{A \rightarrow E_{var}}$ gleich null. Das bedeutet, dass es keine

Datenobjekte in den zur Verfügung stehenden Artefaktinstanzen gibt, welche für die ABox-Modellierung bzw. ein Mapping herangezogen werden können. Daher schließt der erste Fall die Verwendung eines automatisierten Ansatzes aus. Es ist zwingend ein semi-automatisierter Ansatz zu wählen, bei dem die ABox durch Nutzereingaben modelliert wird. Im zweiten Fall (F2) beträgt das Mappingpotential $R_{A \to E_{var}}$ 100%. In diesem Fall sind alle varianten Entitäten der ABox durch Datenobjekte in Artefakten bestimmbar. In diesem Falle müssen die zeitlichen Aufwände des semi-automatischen mit den zeitlichen Aufwänden des automatischen Ansatzes verglichen werden. Die Strategie mit dem kleineren Aufwand, unter Berücksichtigung unsicherer Faktoren wie z.B. der Gewichtung oder der Schätzung der zeitlichen Aufwände pro Nutzereingabe bzw. Mapping eines Datenobjektes in der Berechnung, kann dann gewählt werden. Im dritten Fall (F3) ist eine differenzierte Betrachtung notwendig, da nicht der vollständige Nettomodellierungsbedarf automatisiert erzeugt werden kann. Es ist daher zu berechnen, wie groß der zeitliche Aufwand einer hybriden Strategie Z_h ist. Hierzu wird der Anteil des Nettomodellierungsbedarfs berücksichtigt, der automatisiert erstellbar ist und der Anteil, der basierend auf Nutzereingaben erstellt werden muss. Für die Bestimmung des automatisiert erstellbaren Anteils sein auf Formel IV.16 und IV.19 verwiesen. Für die Bestimmung des semi-automatisiert erstellbaren Anteils sei auf Formel IV.6 und IV.9 verwiesen.

$$\begin{aligned} &F1\colon R_{A \to E_{var}} = 0\% \\ &F2\colon R_{A \to E_{var}} = 100\% \\ &F3\colon 0\% < R_{A \to E_{var}} < 100\% \end{aligned} \tag{F.IV.21}$$

IV-3.3. Werkzeug-Entwicklung zur semi-automatischen ABox-Entwicklung

Im vorliegenden Abschnitt wird eine Architektur (siehe Abschnitt IV-3.3.1) sowie eine Entwurfsmethodik (siehe Abschnitt IV-3.3.2) zur Implementierung der genannten Architektur vorgestellt. Darüber hinaus geht Abschnitt IV-3.3.3 auf ein Datenmodell ein, welches zur Erzeugung von Entitäten (z.B. IRIs) basierend auf Nutzereingaben herangezogen werden kann und eine Validierung der Nutzereingaben berücksichtigt. Abschnitt IV-3.3.4 stellt dann vor, wie unter Zuhilfenahme des Datenmodells aus Abschnitt IV-3.3.3 die in Abschnitt III-3.3.1.3 definierten assertionalen Muster zur semi-automatisierten Erstellung der ABox herangezogen werden können. Abschnitt IV-3.3.5 stellt dann die vorgenommene Implementierung des anhand des vom Autor konzipierten Software-Werkzeugs „LiOnS“ vor.

IV-3.3.1. Architektur des Werkzeugs

Die in diesem Abschnitt definierte Architektur basiert auf dem Model-View-Controller-Softwarearchitekturmuster (MVC) nach [AvZd05], siehe auch Abbildung IV.8. Hierbei ist das Werkzeug generell in drei Teile unterteilt, welche zwar unabhängig voneinander erstellt

werden können, aber aufeinander angewiesen sind, um die Funktionalität des Werkzeugs zu gewährleisten. Dabei besteht das MVC-Muster aus:

- Einem Modell (englisch: „Model“), welches die Daten des Werkzeugs umfasst, sowie Methoden zu deren Manipulation.
- Einer Benutzerschnittstelle (englisch: „View“), welche die Daten des Modells entsprechend für den Nutzer aufbereitet und Eingaben vom Benutzer entgegen nimmt.
- Einer Applikationslogik (englisch: „Controller“) die einer Benutzerschnittstelle zugeordnet ist und die Verarbeitung der Nutzereingaben übernimmt, sowie den Aufruf von Methoden des Modells übernimmt.

Ein Nutzer interagiert strikt über die Benutzerschnittstelle und die dazugehörige Applikationslogik mit dem Werkzeug und hat damit keinerlei direkten Zugriff auf das Modell des Werkzeugs.

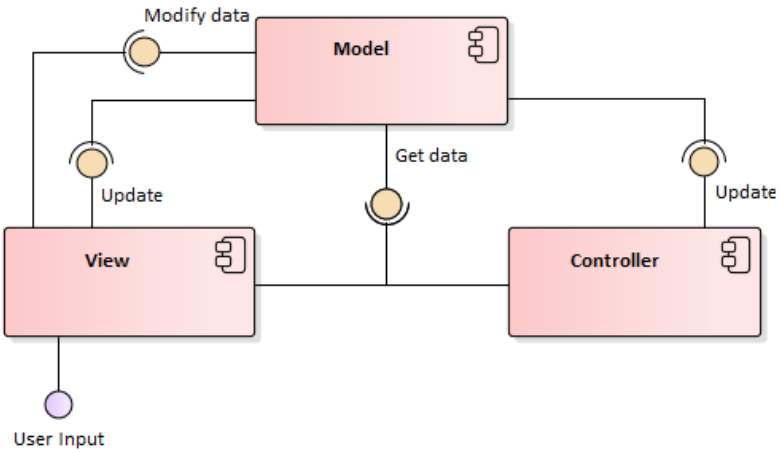

Abbildung IV.8.: Model-View-Controller Architekturmuster entsprechend [AvZd05]

Im vorliegenden Konzept wird das Werkzeug so gestaltet, dass pro Ontologie-Design-Muster (ODM) je ein MVC-Modell definiert wird, siehe auch Abbildung IV.9. Dabei definiert die Benutzerschnittstelle die Eingaben und Anzeigen, die lediglich für die ABox-Erstellung im Kontext eines ODM relevant sind, und blendet alle anderen Modellierungsmöglichkeiten zu Gunsten der Komplexitätsreduktion für den Nutzer aus. Die Applikationslogik hingegen definiert in diesem Zusammenhang alle für die Benutzerschnittstelle relevanten Methoden zur Verarbeitung von Eingaben (bspw. Anlegen von Individuen) und zum Aufruf von Ausgaben (bspw. Anzeigen aller bereits modellierten Individuen). Darüber hinaus bietet die Applikationslogik die Möglichkeit zur Validierung von Nutzereingaben (bspw. prüfen ob, Datentyp zur Range eines DatatypeProperties der TBox passt), sodass eine konsistente Syntax der Eingaben gesichert ist. Das Modell hingegen definiert alle für die Applikationslogik relevanten Methoden zur Modifikation der ABox basierend auf den Nutzereingaben der Benutzerschnittstelle und zum Abruf der aktuell existierenden bzw. modellierten ABox zum Zeitpunkt des Abrufs, sofern relevant für die Benutzerschnittstelle. Für das Hinzufügen von Tripeln in der

ABox nutzt das Modell, bzw. das AB_{t++}-Modell, die in Abschnitt III-3.3.1.3 definierten assertionalen Muster des ODM. Letztere lassen sich auch in SPARQL-SELECT-Abfragen überführen, um den aktuellen Zustand der ABox zu erfragen, was im AB_t-Modell implementiert wird. Alle Abfragen und Updates werden über eine entsprechende Schnittstelle an eine Graph-Datenbank übermittelt. Diese Graph-Datenbank enthält die TBox der relevanten ODMs und besitzt Reasoning-Fähigkeiten entsprechend Abschnitt III-1.3. Ein Hinzufügen von Tripeln, welche zu einer Inkonsistenz führen würden, wird somit proaktiv durch die Graph-Datenbank vermieden. Des Weiteren kann mit Hilfe der Modelle auch die aktuelle ABox aus der Graph-Datenbank exportiert werden.

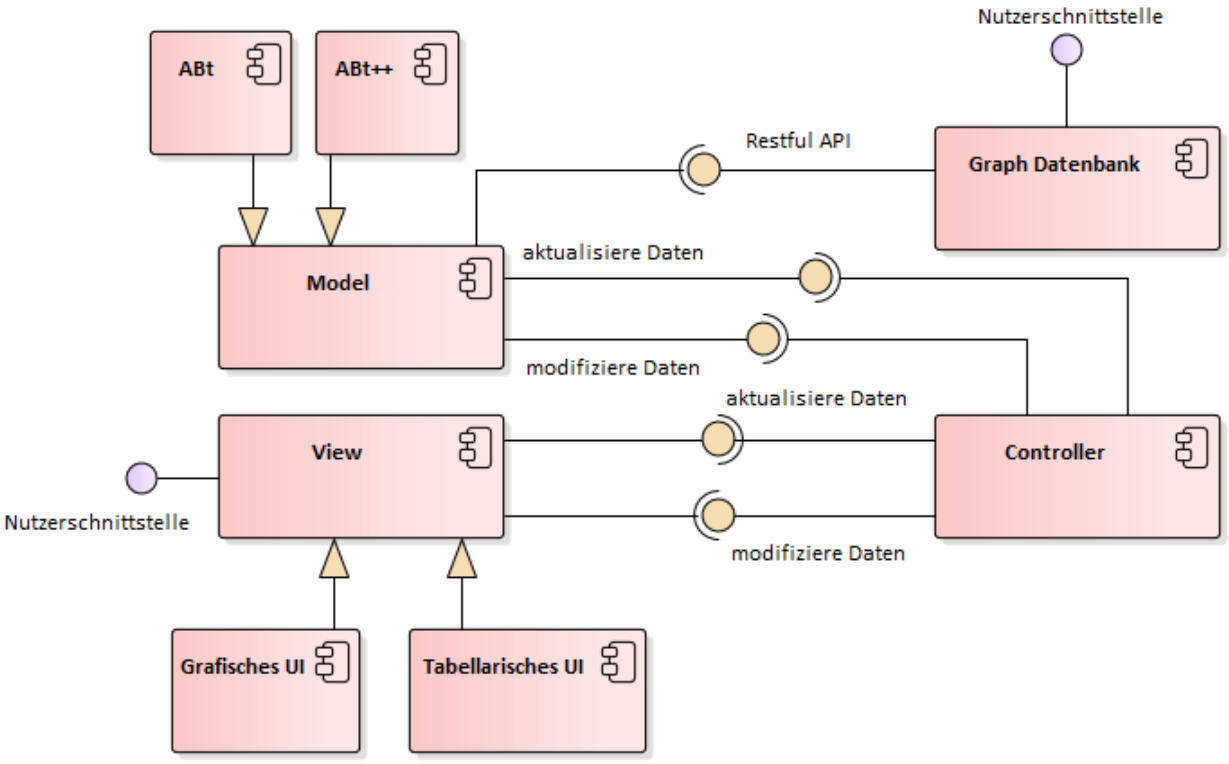

Abbildung IV.9.: Architektur des Werkzeugs im semi-automatischen Ansatz für ein ODM

IV-3.3.2. Methodik der Werkzeugentwicklung

Da das zu entwerfende Werkzeug insbesondere die Stakeholder Domänenexperte und Softwareentwickler und deren Bedürfnisse bei der ABox-Modellierung in den Vordergrund stellen soll, wird eine Methode des User-Centric-Design angewandt, welche auf [Sil+11] basiert und die wesentlichen Elemente der Methoden von [FSM08], [Sy07] und [FNB07] vereint. Der Ablauf der Methode zur Entwicklung einer MVC-Komponente für einen ODM wird in Abbildung IV.10 in Anlehnung an [Sil+11] zusammengefasst und nachfolgend in seinen wesentlichen Elementen erläutert. Essenziell für den Ablauf der Methode ist die iterative Vorgehensweise, welche sich in das Design der Benutzerschnittstelle (grün) und die Implementierung von Modell und Applikationslogik (schwarz) aufteilt. Beginnend mit der Iteration „0“ wird zunächst die Benutzerschnittstelle für einen ODM definiert. Mögliche Vorgehensweisen sind hier die

Definition von User Stories, Papier-Prototypen der Benutzerschnittstelle, Feature-Cards für View-Elemente und vor allem Interviews der Stakeholder, siehe auch [FNB07]. Basierend auf diesen Artefakten wird eine initiale Benutzerschnittstelle für einen ODM erstellt. Für diese Benutzerschnittstelle wird dann in Iteration 1 ein entsprechendes Modell und eine Applikationslogik definiert. Das Ergebnis der Iteration 1 ist dann ein vollständiger MVC für einen ODM, der dann in der Gruppe der Stakeholder evaluiert werden kann, wobei erneut die oben genannten Vorgehensweisen angewendet werden können. Das Ergebnis der Evaluation geht dann in eine neue Iterationsschleife ein.

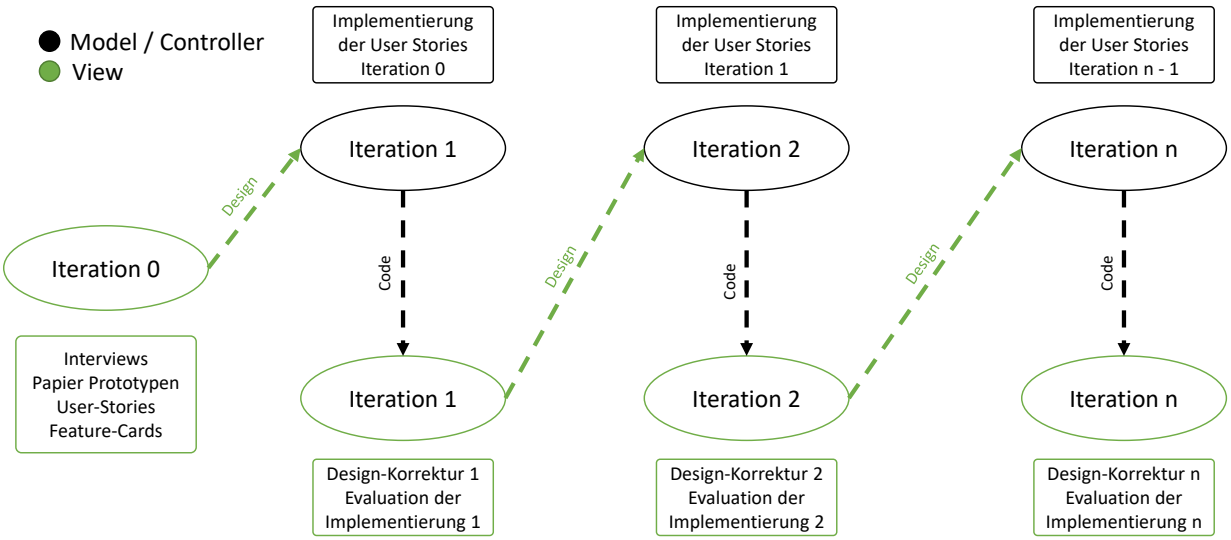

Abbildung IV.10.: Nutzerzentrierte Entwurfsmethode in Anlehnung an [Sil+11]

Aus der Architektur in Kombination mit dem iterativen Vorgehen, ergeben sich pro Iteration drei Hauptaufgaben mit entsprechenden Unteraufgaben, welche für jeden ODM durchzuführen sind. Im ersten Schritt wird die Benutzerschnittstelle definiert. Hierbei sind zunächst die Präferenzen der Stakeholder mit entsprechenden Vorgehensweisen (z.B. User Stories, Papier-Prototypen der Benutzerschnittstelle, Feature-Cards für Benutzerschnittstellen-Elemente) zu ermitteln. Im zweiten Schritt wird das Modell implementiert, siehe auch Abschnitt IV-3.3.4. Hierbei müssen im Wesentlichen zwei Teilmodelle erstellt werden. Einerseits ist zu definieren, welche Informationen über den aktuellen Zustand der ABox der Nutzer benötigt, um zielgerichtet weitere Tripel anlegen zu können. Hierzu wird das AB_t-Modell aus einer Menge an Methoden definiert, welche SPARQL-SELECT-Abfragen der assertionalen Muster entsprechend modifizieren, um den genannten Zweck zu erfüllen. Darüber hinaus werden im AB_{t++}-Modell alle notwendigen SPARQL-Update-Operationen dargestellt, sowie Methoden zu deren Manipulation, die für das Herbeiführen einer Veränderung der ABox relevant sind. Für das AB_t- und AB_{t++}-Modell werden die assertionalen Muster eines ODM herangezogen. Die assertionalen Muster können im Rahmen dieser Tätigkeit zwar abgewandelt werden, allerdings ist sicherzustellen, dass sie für einen gegebenen ODM eine entsprechend dessen TBox konsistente ABox erzeugen. Im dritten Schritt ist dann die zur Benutzerschnittstelle gehörige Applikationslogik zu definieren, welche die Verknüpfung zwischen Benutzerschnittstelle und Modell herstellt sowie die Validierung von Nutzereingaben

übernimmt.

IV-3.3.3. Modellierung von Entitäten

Die Aufgabe der Applikationslogik ist es einerseits, die von der Benutzeroberfläche eingegebenen Daten zu validieren und andererseits für die korrekte Verwendung im Modell aufzubereiten. Insbesondere für letzteren Teil ist die Definition eines Datenmodells notwendig, welches einerseits im Rahmen des Informationsaustausches zwischen Modell und Applikationslogik im Sinne eines Interfaces genutzt werden kann, aber auch Methoden enthält, um Entitäten wie z.B. ein Literal oder IRI basierend auf einer Nutzereingabe zu erstellen und ggf. an das Modell weiterzugeben. Zu diesem Zweck wurde das in Abbildung IV.11 befindliche Modell entworfen, welches auf [Ber+20[@]] beruht und nachfolgend erläutert wird.

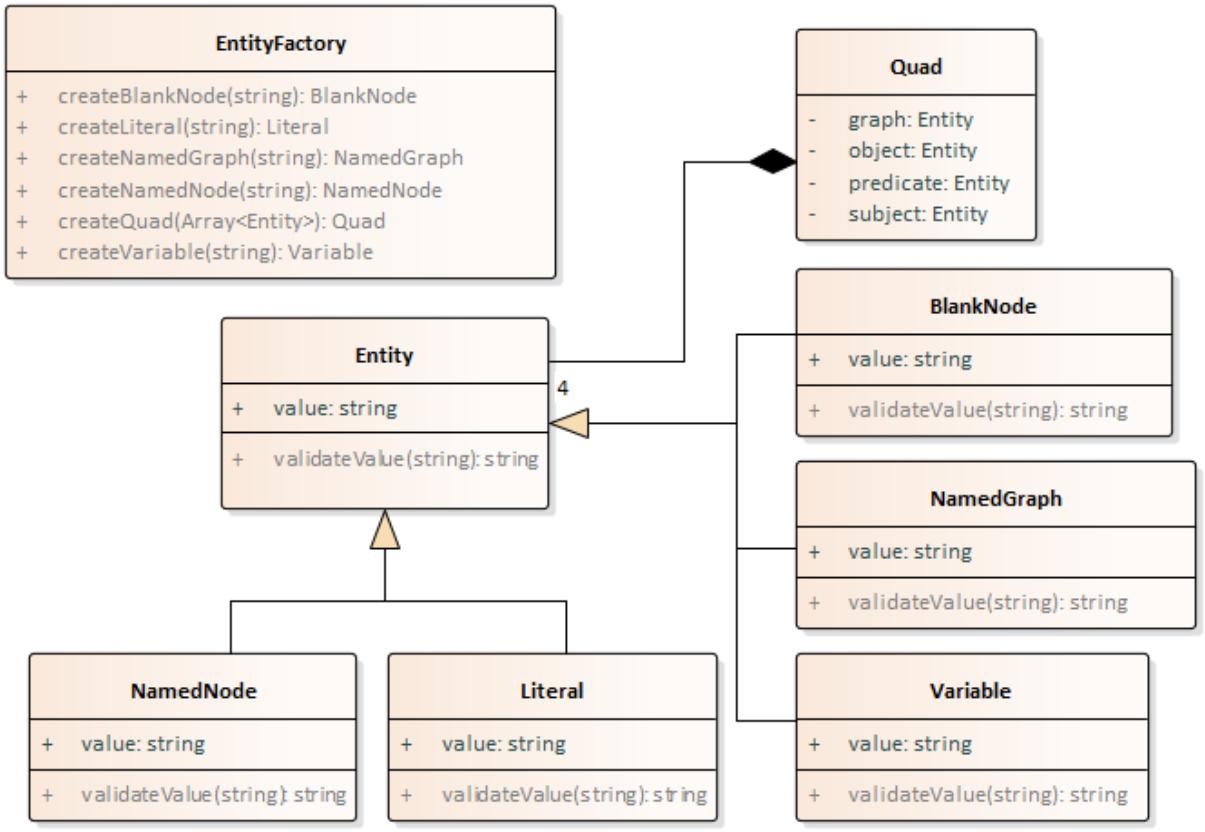

Abbildung IV.11.: RDF-Datenmodell zur Verknüpfung von Modell und Applikationslogik, basierend auf [Ber+20[@]]

Grundsätzlich beruht das in Abbildung IV.11 befindliche Modell auf dem „Factory“ Software-Design-Muster, welches zur Herstellung von Entitäten genutzt werden kann. Eine Entität kann dabei ein `NamedNode`, `Literal`, `Variable`, `NamedGraph` oder `BlankNode` sein. Während ein `NamedNode` im Wesentlichen RDF-Ressourcen wie Individuen, Klassen, Dataproperties u.ä. repräsentiert, repräsentieren Literale hingegen bestimmte Attribute bspw. ei-

nes DataTypeProperties und können noch weitere Sub-Klassen enthalten, je nach möglichen Datentypen der „http://www.w3.org/2001/XMLSchema#“ Ontologie (siehe [Pet+12@]). Die Klasse `Variable` repräsentiert Variablen, welche bspw. in SPARQL-Abfragen genutzt werden können. Die Klasse `NamedGraph` hingegen definiert einen benannten Graphen, welcher analog zur Definition unterschiedlicher Tabellen in relationalen Datenbanken genutzt wird, um Mengen von Tripeln zu beinhalten. `BlankNodes` wiederum werden in RDF genutzt, um komplexere Sprachkonstrukte im Sinne von Verzweigungsknoten abbilden zu können, und beinhalten keine weitere Semantik. Aus einem Tripel (drei Entitäten) und einem NamedGraph kann dann auch ein Quadrupel (`Quad`) definiert werden. Das beschriebene Modell kann nun verwendet werden, um eine entsprechende Typisierung durchzuführen. Darüber hinaus besitzt jede Sub-Klasse von Entity weitere Methoden, welche spezifisch für die jeweilige Klasse sind, was aus Komplexitätsgründen nicht dargestellt ist. So kann bspw. die Klasse NamedNode noch Methoden besitzen, welche den Namensraum oder Bezeichner zurückgeben. Darüber hinaus beinhaltet jede Klasse eine eigene Validierungsfunktion, welche es erlaubt, einen eingegebenen String auf seine Gültigkeit hinsichtlich der intendierten Verwendung zu überprüfen. So implementiert die Methode `validateValue` Methode der Klasse NamedNode bspw. die Validierung, ob es sich bei dem übergebenen Parameter um einen IRI-konformen String handelt. Diese Validierungsfunktionen müssen jedoch nicht aufgerufen werden, sondern werden durch die `Factory` Klasse ohnehin berücksichtigt. Die Factory beinhaltet je eine Methode zur Erstellung einer bestimmten Entität bzw. eines Quadrupels. Bei Aufruf der Factory werden alle notwendigen Verifizierungsschritte unternommen, um eine entsprechende Entität herzustellen und zurück zu geben.

IV-3.3.4. Modellierung assertionaler Muster

Das Modell jeder MVC-Komponente, welche für jeden ODM erstellt wird, soll die durch die Applikationslogik bereitgestellten Daten den variablen Bestandteilen der assertionalen Muster eines ODM zuweisen und ist somit für die Erstellung bzw. Abfrage von ABox-Inhalten aus einer Graph-Datenbank verantwortlich. Wesentlicher Input hierfür sind die im Rahmen der TBox-Entwicklung definierten assertionalen Muster, siehe Abschnitt III-3.3.1.3. Jedes Modell hat die in Abbildung IV.12 dargestellte Struktur, welche nachfolgend beschrieben wird und grundsätzlich aus den drei Klassen `ABox-Service`, `ABt++` und `ABt` besteht. Die Klasse `ABox-Service` implementiert die entsprechenden Schnittstellen zum Bezug und zur Modifikation der ABox für den Controller. Sie aggregiert die `ABt`-Klasse, welche alle relevanten SPARQL-SELECT-Abfragen im Kontext dieses ODM bereitstellt und die Klasse `ABt++`, welche alle relevanten Methoden zur Erstellung der SPARQL-Update-Operationen definiert.

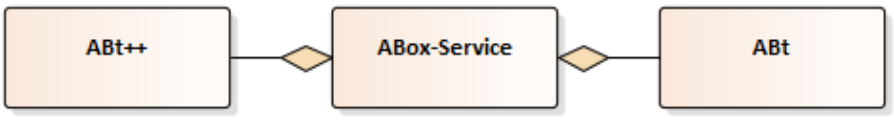

Abbildung IV.12.: Exemplarische Struktur des Modells einer MVC-Komponente eines ODM

Die `ABt`-Klasse definiert die relevanten SPARQL-SELECT-Abfragen, die für die Bereit-

stellung der aktuellen ABox relevant sind. Dabei können sowohl statische (zur Laufzeit des Werkzeugs) nicht zu verändernde Abfragen definiert werden, als auch dynamische Abfragen, welche bspw. durch Nutzereingaben modifiziert werden. Darüber hinaus kann in Abfragen unterschieden werden, welche auf Entitäten der ABox oder TBox abzielen. Während ABox-Abfragen spezifisch für das assertionale Muster sind (siehe auch Listing III.15), sind TBox-Abfragen in ihrer Struktur unabhängig vom assertionalen Muster und beziehen sich auf Entitäten der TBox. Abbildung IV.13 stellt diese Unterscheidung dar.

	ABox-Abfrage	TBox-Abfrage
Statische Abfrage	Definiert als Variable Spezifisch für ODM-Version	Definiert als Variable Spezifisch für ODM-Version
Dynamische Abfrage	Definiert als Methode Spezifisch für ODM-Version	Definiert als Methode Unabhängig von ODM-Version

Abbildung IV.13.: Abfrage-Arten der `ABt`-Klasse

Charakteristisch für eine statische ABox-Abfrage ist die Definition der Abfrage als Variable. Dies ist in Listing IV.1 beispielhaft für den VDI 3682 ODM dargestellt. Dabei können die Abfragen direkt aus den assertionalen Mustern generiert werden, siehe Abschnitt III-3.3.1.3. Die statische Nutzung der Abfrage hat zur Folge, dass die Abfrage zur Entwicklungszeit des Werkzeugs definiert wird und zur Laufzeit nicht mehr änderbar ist. Darüber hinaus bezieht sich die Abfrage explizit auf den Namensraum des ODM und liefert als Rückgabe eine Menge an Individuen, welche dem Abfragemuster entsprechen.

Listing IV.1: Statische ABox-Abfrage am Beispiel des VDI 3682 ODM

```
public SELECT_TECHNICAL_RESOURCE: string = `
    PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
    PREFIX VDI3682: <http://www.hsu-ifa.de/ontologies/VDI3682#>

    SELECT ?TechnicalResource WHERE {
        ?TechnicalResource a VDI3682:TechnicalResource.
    }
`;
```

Charakteristisch für eine dynamische ABox-Abfrage ist, dass sie zur Laufzeit des Werkzeugs mit entsprechenden Variablen dynamisch befüllt wird. Hierzu werden die Abfragen als Methoden definiert. Dies ist in Listing IV.2 beispielhaft dargestellt. Die übergebenen Parameter (z.B. „Any_Individual“) werden bei Aufruf der Methode genutzt, um variable Bestandteile der Abfrage in der WHERE-Klausel zu befüllen und dann den SPARQL-SELECT als Rückgabe zu liefern. Im Beispiel von Listing IV.2 wird der Methode ein Individuum übergeben, woraus eine SPARQL-Abfrage erstellt wird, welche nach der Klasse des Individuums sucht, welche

im Namensraum des VDI 3682 ODMs liegt. Auch dynamische ABox-Abfragen werden aus den assertionalen Mustern entsprechend Abschnitt III-3.3.1.3 generiert, allerdings werden bestimmte Bestandteile des Musters als variabel definiert.

Listing IV.2: Dynamische ABox-Abfrage am Beispiel des VDI 3682 ODM

```
public selectClass(Any_Individual: NamedNode): string {
    var selectString = `
    PREFIX owl: <http://www.w3.org/2002/07/owl#>
    PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

    SELECT ?Class WHERE {
    BIND(IRI("${Any_Individual.value}") AS ?Individual)
    ?Individual rdf:type ?Class.
    ?Class a owl:Class.
    FILTER(STRSTARTS(STR(?Class), "http://www.hsu-ifa.de/ontologies/VDI3682#"))
    }`;
    return selectString
}
```

Des Weiteren können für eine bestimmte Nutzerführung auch Abfragen von Interesse sein, welche nach Entitäten der TBox eines ODM fragen. TBox-Abfragen beziehen sich nicht auf Entitäten der ABox, sondern auf Entitäten der TBox und ermöglichen bspw. die Frage nach einer Menge an Klassen einer TBox, wie z.B. in Listing IV.3 dargestellt. Dies ist insofern von Interesse, als das die Rückgaben dieser Abfragen dem Nutzer oder anderen Software-Komponenten eine der Semantik der TBox entsprechenden Auswahlmöglichkeiten bieten. Auch hier werden die statischen TBox-Abfragen einmal zur Entwicklungszeit spezifiziert und können zur Laufzeit des Werkzeugs nicht verändert werden.

Listing IV.3: Statische TBox-Abfrage am Beispiel des VDI 3682 ODM

```
public SELECT_LIST_OF_ALL_CLASSES: string = `
    PREFIX owl: <http://www.w3.org/2002/07/owl#>

    SELECT DISTINCT ?type WHERE {
    ?type a owl:Class.
    FILTER(STRSTARTS(STR(?type), "http://www.hsu-ifa.de/ontologies/VDI3682#"))
    }
`;
```

Bei dynamischen TBox-Abfragen wird ähnlich zu dynamischen ABox-Abfragen eine Methode definiert, deren Parameter zur Definition der Abfrage zur Laufzeit genutzt werden. Bspw. kann danach gefragt werden, welche ObjectProperties mit einem Subjekt einer bestimmten Klasse verbunden werden dürfen, um dem Nutzer lediglich die erlaubten ObjectProperties als Auswahl zu bieten. Eine diesem Beispiel entsprechende Abfrage ist in Listing IV.4 zu finden. Das Listing stellt eine Methode dar, welche lediglich den IRI einer OWL-Klasse (NamedNo-

de) der TBox übergeben bekommt und alle ObjectProperties als Rückgabe liefert, welche diese Klasse als Domain besitzen. Auf diese Weise kann der Nutzer beispielsweise aus einer Auswahl eine Klasse wählen, woraufhin eine weitere Auswahl nur noch ObjectProperties anzeigt, welche weiterhin von Relevanz sind.

Listing IV.4: Beispiel einer dynamischen TBox-Abfrage

```
public selectPredicateByDomain(Any_OWL_Class: NamedNode): string {
    var SELECT_String = `
        PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
        PREFIX owl: <http://www.w3.org/2002/07/owl#>
        PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
        SELECT ?ObjectProperty WHERE {
        ?ObjectProperty rdfs:domain ?a.
        # optionally if the range is a blank node
        OPTIONAL {
               ?a owl:unionOf ?c.
               ?c rdf:rest* ?e.
               ?e rdf:first ?g.}
        # in case the range is a blank node, use the rdf:first as return
        BIND(IF(isBlank(?a),?g,?a) AS ?Property)
        # filter for class
        FILTER(?Property = IRI("${Any_OWL_Class.value}"))
    }`;
    return SELECT_String
}
```

Wie in dem Beispiel für die Methode in Listing IV.4 erkennbar, werden lediglich die Namensräume der W3C standardisierten Ontologien genutzt, während, wie im Beispiel für die Abfragen in Listing IV.1 - IV.3 erkennbar, die Abfrage stets den Namensraum des ODM enthält bzw. im Rahmen eines Filters nutzt. Die getroffene Unterscheidung hat einen wesentlichen Einfluss auf die Definition des AB_t-Modells. Alle SPARQL-Abfragen, welche das Vokabular eines entsprechenden ODM nutzen, müssen bei Änderungen dieses ODM auf Gültigkeit geprüft werden. Während für die W3C standardisierten Ontologien eine gewisse Stabilität in der Versionierung anzunehmen ist, können sich ODMs ständig weiter entwickeln, da jede weitere Nutzung eines ODMs aufgrund der iterativen Vorgehenswiese der Gesamtmethode auch Verbesserungspotentiale der darin enthaltenen Semantik offenbaren kann. Wenn nun ausschließlich statische ABox-Abfragen definiert werden, kann dies bei häufigen Versionierungen eines ODM dazu führen, dass ein hoher Wartungs- bzw. Migrationsaufwand entsteht. Dies kann dadurch eingegrenzt werden, dass die ABox-Abfragen möglichst immer in Kombination mit dynamsichen TBox-Abfragen ausgeführt werden, um bspw. mögliche ObjectProperties nicht statisch zu definieren, sondern in Abhängigkeit von der TBox eines ODM und dessen aktueller Version dynamisch zu wählen. Des Weiteren sind dynamische TBox-Abfragen aufgrund ihrer generellen Unabhängigkeit von einem ODM auch für unterschiedliche AB_t-Modelle von Interesse, da sie einfach wiederverwendet werden können.

Die `ABt++`-Klasse definiert relevante Methoden zur Erstellung der SPARQL-Update-

Operationen INSERT und DELETE und implementiert damit alle relevanten Methoden des $AB_{t_{++}}$-Modells. Dabei werden so viele Methoden definiert, wie unterschiedliche INSERT bzw. DELETE Operationen vorliegen. Die Mindestmenge ist dabei durch die Anzahl der assertionalen Muster eines ODM definiert (bspw. Minimal- und Maximalmuster). Die Operationen dieser Mindestmenge können jedoch je nach Benutzerführung erweitert werden, weil bspw. nicht alle notwendigen Nutzereingaben auf einmal in die ABox überführt werden sollen, sondern schrittweise. Listing IV.5 zeigt die Struktur einer Methode des $AB_{t_{++}}$-Modells. Die in Listing IV.5 übergebene Variable `graph` ist ein Objekt vom Typ `Quad`, welches alle relevanten Variablen der Update Operation definiert. Die boolesche Variable wird genutzt, um die Art der Update-Operation fest zu legen.

Listing IV.5: Beispielhafte Struktur einer Methode des $AB_{t_{++}}$-Modells

```
public buildSPARQL_Update(graph: Quad, operationType: boolean) {
    var operationKey: string;
    if (operationType === true) {operationKey = "INSERT"}
    else (operationType === false) {operationKey = "DELETE"}

    var Update_String = `
        ${ operationKey } {
            ?subject ?predicate ?object;
            a owl:NamedIndividual.
      } WHERE {
          BIND(IRI(STR("${graph.subject.value}")) AS ?subject).
          BIND(IRI(STR("${graph.predicate.value}")) AS ?predicate).
          BIND(IRI(STR("${graph.object.value}")) AS ?object).
      }`;
    return Update_String
}
```

IV-3.3.5. Implementierung im Werkzeug LIOnS

Das Werkzeug LIOnS („Lightweight Industrial Ontology Support“) ist ein Software-Werkzeug zur Modellierung der assertionalen Box einer Ontologie, welches auf GitHub durch den Autor veröffentlicht wurde [HiKö19@b]. LIOnS ist so konzipiert, dass eine gewisse Menge von Ontologie-Design-Mustern (ODM) bereits vorgegeben sind, aber ohne architektonische Änderungen auch weitere ODMs hinzugefügt werden können. Die Grundmenge von ODMs entspricht den im Rahmen dieser Arbeit Vorgestellten, wobei in den Abschnitten IV-3.3.5.1 und IV-3.3.5.2 je eine exemplarische Implementierung vorgestellt wird.

Der Ablauf zur Initialisierung und Nutzung von LIOnS ist in Abbildung IV.14 dargestellt und wird nachfolgend erläutert. Grundsätzlich ist LIOnS als Web-Applikation im von Google entwickelten Angular-Framework [Goo20@] implementiert, welcher im Wesentlichen auf der Programmiersprache Typescript basiert. Jeder ODM ist entsprechend der Architektur aus Abschnitt IV-3.3.1 als Angular-Komponente implementiert, wobei die Benutzerschnittstelle

in HTML 5 und die Applikationslogik und Modelle der Komponenten in Typescript erstellt sind. Um die Funktionsfähigkeit des Werkzeugs zu ermöglichen, nutzt LIOnS neben dem im Angular-Framework entwickelten Frontend ein in nodeJS [Ope20@] geschriebenes Backend, welches mit einer Graph-Datenbank, hier GraphDB [Ont19@], interagiert.

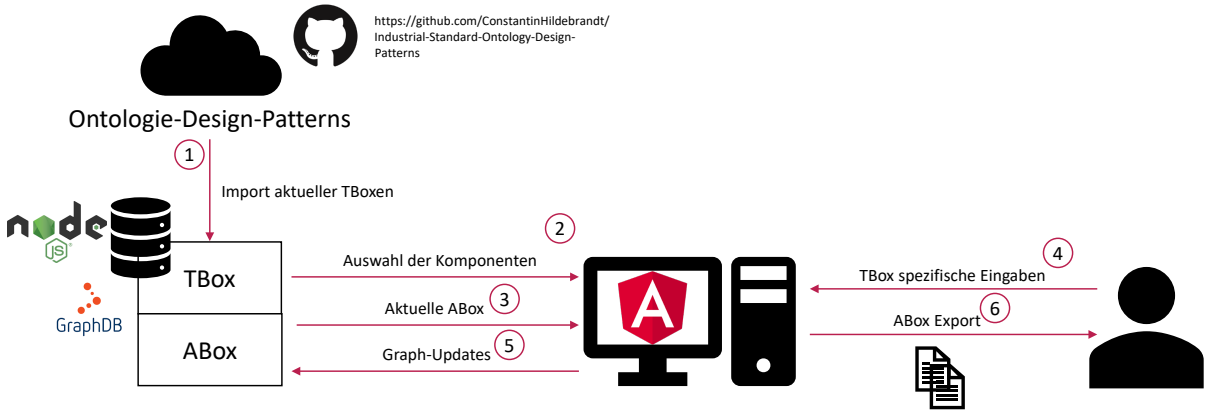

Abbildung IV.14.: Initialisierung und Nutzung von LIOnS

Die Interaktion zwischen Backend und der Graph-Datenbank erfolgt dabei über die RESTful RDF4J API [Ecl20@]. Bei Start des Werkzeugs, also Start von Frontend und Backend, lädt das Werkzeug zunächst die aktuellen Versionen der relevanten ODMs in die Graph-Datenbank, welche entsprechend Abschnitt III-3.3.3 dokumentiert und für Mensch und Maschine erreichbar sind. Der Nutzer kann nun aus den für sein Modellierungsprojekt relevanten ODMs wählen, was dann auf eine für den ODM angepasste Benutzerschnittstelle führt. Führt der Nutzer nun bestimmte Eingaben durch, indem er bspw. einen Prozess mit Hilfe des VDI3682 ODMs modelliert, werden die in Abschnitt IV-3.3.4 vorgestellten Arten von SPARQL-Abfragen und -Updates mit Nutzereingaben befüllt und somit einerseits die Benutzerschnittstelle mit Informationen versorgt, aber auch die ABox entsprechend den Nutzereingaben erweitert. Letztlich kann der Nutzer sich die modellierte ABox in verschiedenen Datenformaten exportieren lassen, unter anderem XML und JSON. LiOnS fungiert somit als eine Art „ETL-Pipeline“, wobei die Datenquelle das Wissen der nutzenden Personen ist und dieses Wissen in die ABox überführt wird.

IV-3.3.5.1. Merkmalsmodellierung

Für die Modellierung von Merkmalen entsprechend des DIN EN 61360 ODM (siehe Abschnitt III-4.4) wurde eine Benutzeroberfläche geschaffen, welche das Anlegen von Typbeschreibungen und Instanzbeschreibungen zulässt. Mit Hilfe einer ecl@ss Suche kann das Anlegen von Typbeschreibungen sowie das übergreifende Verständnis der modellierten Merkmale erheblich verbessert werden. Bei Nutzung dieser Funktion fragt das Backend in einer lokalen ecl@ss-Datenbank Merkmale an und bietet sie dem Nutzer zu Auswahl. Abbildung B.5 im Anhang B zeigt die Benutzerschnittstelle für das Anlegen einer Typbeschreibung unter Nutzung von ecl@ss-Merkmalen.

Nachdem eine neue Typbeschreibung durch den Nutzer angelegt oder aus der ecl@ss-

Datenbank übernommen wurde, kann eine Instanzbeschreibung angelegt werden. Hierzu wählt der Nutzer zunächst die Typbeschreibung aus, für welche die Instanzbeschreibung angelegt werden soll. Für die Ermittlung der existierenden Typbeschreibungen wird eine statische ABox-Abfrage (siehe Abschnitt IV-3.3.4) genutzt. Im Anschluss kann der Nutzer aus allen Entitäten wählen, welche entsprechend der Alignment-Ontologie (siehe Abschnitt III-4.5) zulässig sind, um die neue Instanzbeschreibung damit zu verknüpfen. Für die DIN EN 61360 sind das alle Entitäten, mit Ausnahme der Entitäten der DIN EN 61360 selbst. Anschließend können die Attribute der Instanzbeschreibung (Wert, Aussageziel und Aussagelogik) definiert werden. Die Auswahlmöglichkeiten für Aussageziel und Aussagelogik werden mit Hilfe einer statischen TBox-Abfrage (siehe Abschnitt IV-3.3.4) ermittelt und gehen somit auf die Auswahlmöglichkeiten in der TBox zurück.

IV-3.3.5.2. Prozessmodellierung

Für die Prozessmodellierung mit Hilfe des VDI3682 ODM (siehe Abschnitt III-4.1) wurde das in [NHF20*] vorgestellte Werkzeug „FPB.js“ zur Modellierung von Formalisierter Prozessbeschreibung (FPB) verwendet, um eine grafische Modellierung zu ermöglichen. Da jedoch eine Integration des Werkzeugs aus [NHF20*] und LiOnS einen erheblichen Aufwand bedeutet hätte, bietet LiOnS die Möglichkeit, die aus FPB.js exportierten Artefakte mit Hilfe einer auf dem Ansatz in Abschnitt IV-3.3.4 beruhenden Methodik in die ABox zu überführen. Hierzu wurde ein ETL-Verfahren implementiert, welches das Artefakt von FPB.js analysiert und die in Abschnitt IV-3.3.4 dargestellten Methoden aufruft, die wiederum assertionale Muster beinhalten. Der Nutzer wird somit in die Lage versetzt, die FPB grafisch zu modellieren, wobei die Modellierungsergebnisse genutzt werden, um die in Abschnitt IV-3.3.4 vorgestellten Arten von SPARQL-Abfragen und -Updates zu befüllen.

IV-3.4. Modellierungsmethodik im Kontext des Engineerings fertigungstechnischer Systeme

Wie in Abschnitt I-2.1.2.3 beschrieben, können die im Rahmen des Engineerings erstellten Informationen in drei Bereiche eingeteilt werden: Funktion, Struktur und Verhalten. Während der Entwicklung eines fertigungstechnischen Systems fallen eine Vielzahl an Daten, Informationen und Wissen an, welche diesen drei Bereichen zugeordnet werden können [Hil+17*a].

Für das Konzept dieser Arbeit bedeutet dies, dass potentiell eine Vielzahl an Ontologie-Design-Mustern zur Abbildung dieser Informationen vorliegen kann. Eine Nutzbarkeit der in Abschnitt III-4 vorgestellten Ontologie-Design-Muster ist nur dann gegeben, wenn sie die in einem Projekt vorliegenden Anforderungen erfüllen können. Die Entwicklungsmethodik für die ABox einer Ontologie muss somit von den genutzten Ontologie-Design-Mustern bzw. Artefakten abstrahieren, um eine entsprechende Generalisierbarkeit zu erreichen. Zu diesem Zweck teilt sich das generelle Vorgehen in drei Schritte auf:

1. Bestimmung des Workflows: Analyse bestehender Engineering-Workflows und Festlegung der chronologischen Reihenfolge der Artefakt-Erstellung.

2. Definition von Modellverknüpfungen: Ergänzung der Mapping-Artefakte oder Nutzerschnittstelle des Werkzeugs um Modellelemente, welche eine Verknüpfung zwischen verschiedenen Modellen ermöglichen.

3. Durchführung der Modellerstellung: Durchführung der Modellerstellung entsprechend der bestimmten Reihenfolge und der gewählten Modellierungsstrategie.

Die Notwendigkeit des ersten Schrittes ist insbesondere darin begründet, dass im Rahmen der Durchführung eines einzelnen semi-automatischen oder automatischen Verfahrensschrittes bestimmte Teile der ABox unter Umständen bereits materialisiert sein müssen, also eine Sequenz vorliegt. Dies ist beispielsweise der Fall, wenn zunächst ein Funktionsmodell materialisiert wird und anschließend physisch existierende Systeme aus einem 3D-CAD-Modell damit verbunden werden sollen.

Die Notwendigkeit des zweiten Schrittes ergibt sich aus der Tatsache, dass im Engineering eines automatisierten Systems ein durchgängiger Informationsfluss nicht auf natürliche Weise vorliegt, sondern stattdessen mit entsprechenden Aufwänden sichergestellt werden muss [Sch18]. Darüber hinaus werden i.d.R. diverse unterschiedliche Artefakte im Rahmen des Engineerings erzeugt (siehe I-2.1.2.2). Liegt kein durchgängiger Informationsfluss im Engineering und eine Vielzahl an Artefakten vor, so liegen innerhalb der einzelnen Engineering-Artefakte unter Umständen keinerlei Verweise auf andere Engineering-Artefakte vor. Würden nun lediglich die Inhalte der einzelnen Artefakte in die ABox überführt, so würde eine Vielzahl disjunkter Graphen entstehen und somit die Kohärenz der ABox nicht gewährleistet. Es ist also notwendig, etwaig implizit vorliegende Verknüpfungen zwischen Artefakten im Rahmen der ABox-Erstellung zu explizieren.

Abbildung IV.15 stellt dies exemplarisch dar. Die Artefakte A bis C werden nacheinander erzeugt und besitzen implizite Verknüpfungen untereinander, die lediglich den entsprechenden Domänenexperten bekannt sind. Würden diese Verknüpfungen nicht explizit in einem semi-automatischen oder automatischen Verfahrensschritt erstellt, würden drei disjunkte Graphen in der ABox erstellt werden.

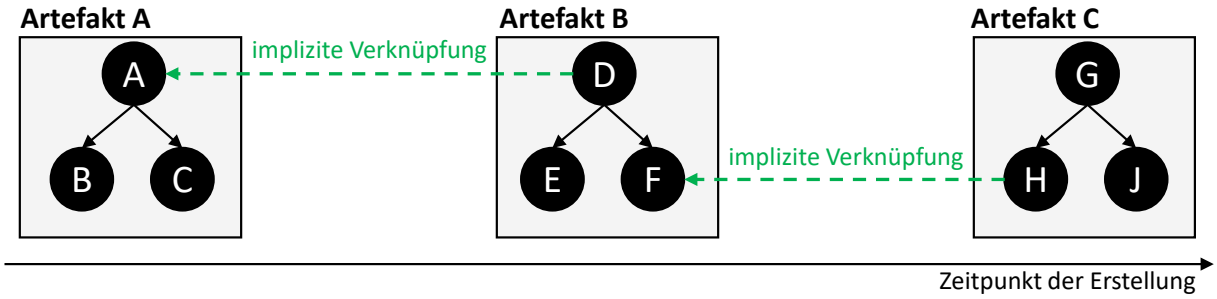

Abbildung IV.15.: Exemplarische Modellarchitektur mit lediglich impliziter Kohärenz

Der dritte Schritt ist lediglich der Vollständigkeit halber aufgeführt und verweist auf die eigentliche Durchführung des Projekt-spezifisch entwickelten Workflows und der darin enthaltenen Strategie zur ABox-Erstellung. Diese genannten drei Schritte werden in den nachfolgenden Abschnitten näher beschrieben, wobei als exemplarisches Beispiel der in [SHF17*] ge-

zeigte Engineering-Workflow und die in Abschnitt III-4 definierten Ontologie-Design-Muster herangezogen werden.

IV-3.4.1. Bestimmung des Workflows

Der erste Schritt befasst sich mit der Bestimmung der chronologischen Reihenfolge, in welcher die Entwicklungsartefakte eines fertigungstechnischen Systems entstehen, welche im Rahmen der ABox-Entwicklung berücksichtigt werden sollen. Zur Darstellung eines vorliegenden Engineering-Workflows bietet sich ein UML-Sequenzdiagramm an, wobei die entwickelnden Ingenieure als Akteure und die genutzten Werkzeuge als Objekte beschrieben werden. Als Nachrichten werden die im Rahmen des Engineerings ausgetauschten Informationen modelliert, wobei angenommen wird, dass diese ausgetauschten Informationen als Artefakt nutzbar sind. In dieser Darstellung wird nun ein Objekt zur Repräsentation der ABox ergänzt, bspw. eine Datenbank. Zusätzlich kann bei Bedarf auch noch ein Werkzeug entsprechend Abschnitt IV-3.3 als weiteres Objekt dargestellt werden. Der bestehende Engineering-Workflow wird nun um Nachrichten an die ABox ergänzt, um anzuzeigen, wann ein Artefakt in die ABox überführt werden soll. Dabei wird die mit Hilfe von Abschnitt IV-3.2 definierte Modellierungsstrategie angewandt.

Abbildung IV.16 stellt dies exemplarisch für den in [SHF17*] dargestellten Engineering-Workflow dar. Demnach sind klassisch je ein Vertreter der Gewerke Maschinenbau, Elektrotechnik und Automatisierungstechnik an der Entwicklung eines fertigungstechnischen Systems beteiligt. Die von den Gewerken genutzten Werkzeuge sind jeweils einzeln als Objekte dargestellt und darüber hinaus noch ein Werkzeug entsprechend Abschnitt IV-3.3. Das letzte Werkzeug stellt eine Datenbank (z.B. „GraphDB“) zur Speicherung der ABox dar. Als vereinfachende Annahme erstellt jedes Werkzeug eine Klasse von Artefakten und wird nur einmal genutzt. Als Nachrichten werden dann die Nutzung der Werkzeuge, sowie die Überführung der erstellten Informationen in andere Werkzeuge dargestellt. In Abbildung IV.16 werden somit vier Artefakte entworfen (Funktionsmodell, 3D-CAD, E-CAD, Steuerungscode). Gemäß dem Fall, dass das Funktionsmodell, das 3D-CAD-Artefakt sowie der Steuerungscode für die ABox-Erstellung von Relevanz sind, so können diese für die gewählte Strategie eingezeichnet werden. Die Strategie wird entsprechend Abschnitt IV-3.2 bestimmt. Im Beispiel von Abbildung IV.16 ist dies eine hybride Strategie, da für das Funktionsmodell eine semi-automatisierte Erstellung mittels eines ABox-Engineering-Werkzeugs und für 3D-CAD-Artefakt sowie Steuerungscode eine automatisierte ABox-Erstellung mittels Mappings vorgenommen wird.

IV-3.4.2. Modellverknüpfung

Nachdem die Reihenfolge der Erstellung der Informationen bekannt ist, kann die für die Erstellung der ABox gewählte Modellierungsstrategie (siehe Abschnitt IV-3.2) um Modellverknüpfungen ergänzt werden. Hierzu werden die möglichen Modellverknüpfungen der Alignment-Ontologie entnommen, siehe z.B. Abschnitt III-4.5. Dieser Schritt ist bei einer semi-automatisierten ABox-Erstellung grundsätzlich notwendig, da die genutzten assertiona-

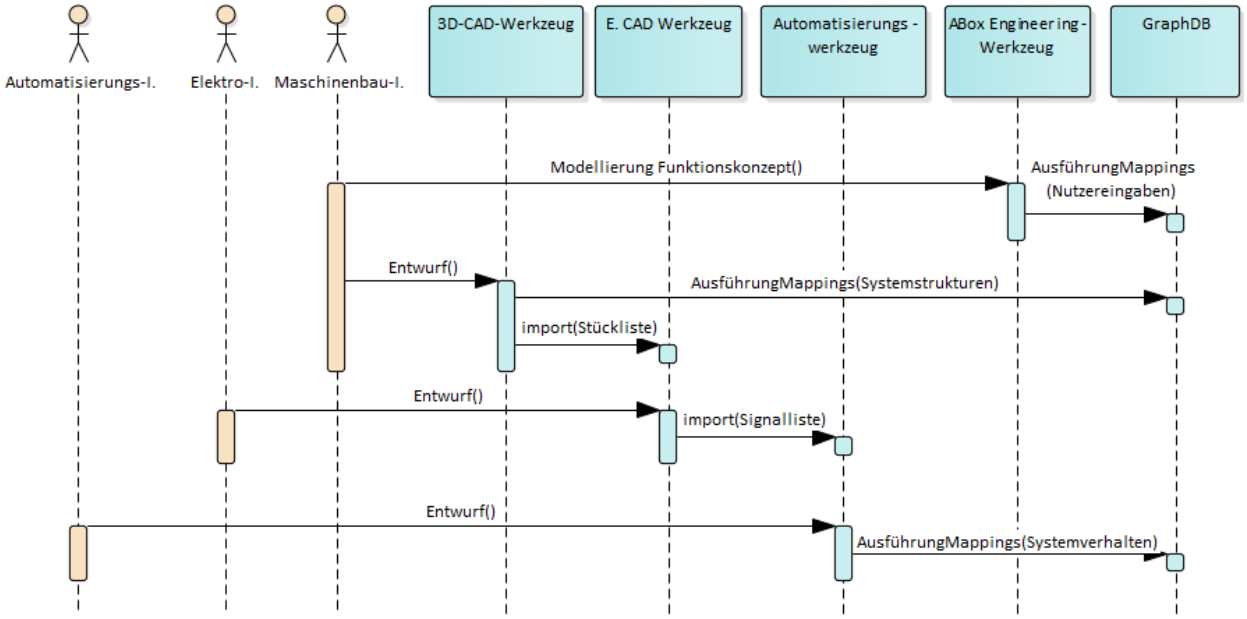

Abbildung IV.16.: Exemplarischer Engineering-Workflow basierend auf [SHF17*]

len Muster keine Elemente der Alignment-Ontologie enthalten und somit um diese erweitert werden müssen. Darüber hinaus muss die Nutzerschnittstelle des Werkzeugs auch eine entsprechende Eingabemöglichkeit bereitstellen. Bei einer automatisierten ABox-Erstellung ist dieser Schritt immer dann notwendig, wenn in den zugrundeliegenden Artefakten die geforderten Verbindungen nicht abgebildet wurden und daher in einem Mapping zu ergänzen sind. Letzteres ist bspw. der Fall, wenn aus einem 3D-CAD-Artefakt die Struktur eines fertigungstechnischen Systems in der ABox dargestellt und mit dem Funktionsmodell entsprechend Abschnitt III-4.5 verbunden werden soll. Sofern das 3D-CAD Artefakt keine Verweise auf Elemente des Funktionsmodells beinhaltet, müssen diese Verweise in den Mappings berücksichtigt werden. Abbildung IV.17 stellt eine exemplarische Modellarchitektur basierend auf den Ontologie-Design-Mustern und der Alignment-Ontologie aus Abschnitt III-4 dar. Angenommen, es wurden drei 3D-CAD-Artefakte erstellt, zwei für Produkte und eines für ein fertigungstechnisches System, dann können die Strukturen der Produkte und des fertigungstechnisches System mit Hilfe des VDI 2206 ODM dargestellt werden. Für das Verhaltensmodell des fertigungstechnischen Systems kann aus dem Steuerungscode die ABox des ISA 88 ODM erstellt werden und die Funktion des Systems kann mittels des VDI 3682 ODM modelliert werden. Die einzelnen Artefakte (3D-CAD und Steuerungscode) beinhalten zwar relevante Informationen, welche mit Hilfe eines Mappings in die ABox überführt werden können. Dass es sich bei diesen Modellelementen jedoch um Beschreibungen von Modellelementen einer anderen Modellierungsperspektive (bspw. Produkt vs. Technische Ressource) handelt, ist in den einzelnen Artefakten nicht explizit enthalten und muss somit in den jeweiligen Mappings ergänzt werden. Die grün dargestellten Verbindungen stellen dies explizit dar und entstammen der Alignment-Ontologie. Sie müssten in den jeweiligen Mappings der Artefakte ergänzt werden, bzw. im Falle einer semi-automatisierten ABox-Erstellung dem genutzten Werkzeug als Nutzereingabe durchgeführt werden.

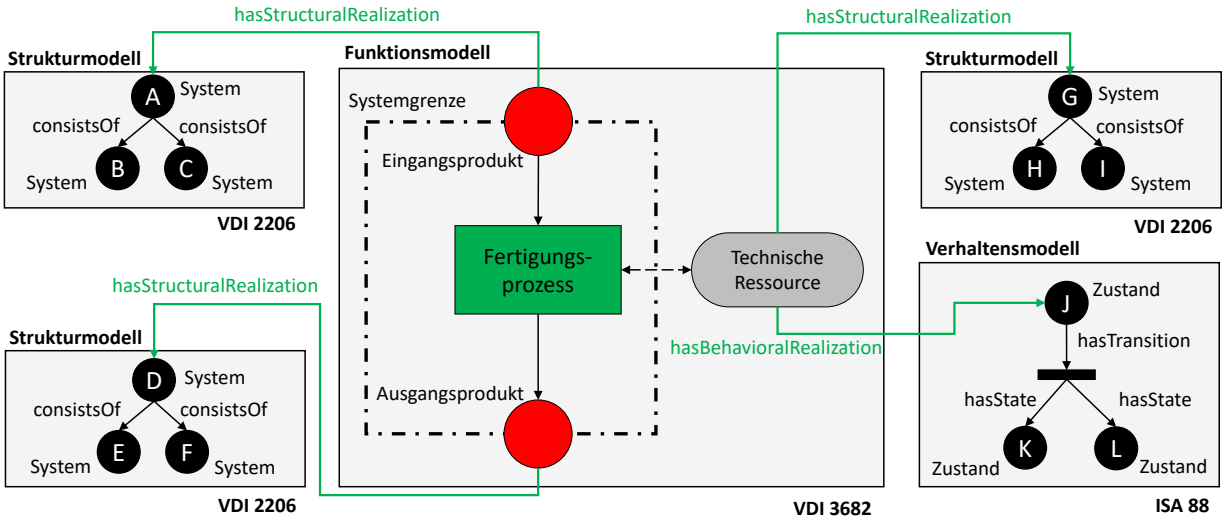

Abbildung IV.17.: Exemplarische Modellarchitektur mit kohärenten Graphen

IV-3.4.3. Modellerstellung und finale Verifikation

Unter Nutzung der im ersten Schritt beschriebenen Reihenfolge sowie den erstellten Mappings bzw. des ABox-Engineering-Werkzeugs kann die Erstellung der ABox erfolgen. Hierzu werden in der festgelegten Reihenfolge Mappings statischer Daten ausgeführt und evtl. durch Nutzereingaben im ABox-Engineering-Werkzeug ergänzt. Die Mappings dynamischer Daten erfolgen erst zur Laufzeit der Systeme durch eine entsprechende Middleware. Letzteres ist bspw. in [Gra18] dargestellt.

Abschließend ist in diesem Schritt eine finale Verifikation der erstellten Ontologie durchzuführen, indem die aus den Kompetenzfragen und Antworten erstellten SPARQL-Abfragen erneut ausgeführt werden. Somit wird die Überprüfung der Vollständigkeit entsprechend Formel III.29 erneut durchgeführt, um zu verifizieren, ob alle Anforderungen mit der erstellten ABox erfüllt werden können. Ist dies nicht der Fall, führen also bestimmte SPARQL-Abfragen nicht zum gewünschten Ergebnis, sind die entsprechenden Informationen zu identifizieren, welche bisher nicht durch Mappings oder Nutzereingaben erstellt wurden. Hierzu müssen die SPARQL-Abfragen mit dem ABox-Modell abgeglichen werden, um zu erkennen, welche varianten oder invarianten Entitäten in der ABox fehlen.

V. Evaluation

Das vorliegende Kapitel befasst sich mit der Evaluation der im Rahmen dieser Arbeit entwickelten Artefakte. Hierzu stellt der Abschnitt V-1 zunächst dar, auf welche Weise die Evaluation im Rahmen dieser Arbeit durchgeführt wurde. Der Abschnitt V-2 stellt dann die durchgeführten Evaluationen vor und beschreibt die daraus gewonnenen Erkenntnisse, bevor Abschnitt V-3 eine abschließende Bewertung hinsichtlich der Erreichung des in Abschnitt I-1.3 definierten Forschungsziels vornimmt.

V-1. Methodischer Überblick

Die nachfolgenden Abschnitte stellen die im Rahmen dieser Arbeit angewandte Methodik zur Evaluation dar. Hierzu geht Abschnitt V-1 zunächst auf die grundsätzliche Art der Evaluationsmethodik ein. Abschnitt V-1.2 stellt dann den im Rahmen dieser Arbeit vorgenommenen Verlauf der Evaluation dar.

V-1.1. Evaluationsmethodik

Für die Durchführung der Evaluation kommen grundsätzlich diverse Methoden in Frage: Expertenbefragungen, beobachtende Fallstudien, statistische Experimente und Einzelfall-Mechanismen-Experimente (kurz: Fallstudien) [Wie14]. Weitere Methoden wären Analysen (z.B. Architekturanalysen), Tests, sowie beschreibende Vorgehen (z.B. Szenariobeschreibungen) [Hev+04]. Bei den zu evaluierenden Methoden handelt sich es zu großen Teilen um komplexe Vorgehensmodelle, für welche nur eine sehr begrenzte Menge an Fachexperten existiert (siehe Teil III für Mangel an Ontologen). Des Weiteren ist eine Anwendung der entwickelten Methoden i.d.R. mit einem Aufwand verbunden, welcher sich über mehrere Wochen und Personenmonate erstreckt und nicht in einzelnen Workshops untergebracht werden kann. Daher werden Fallstudien als geeignete Evaluationsmethodik für die im Rahmen dieser Arbeit entwickelten Artefakte angesehen.

Im Zuge einer Fallstudie werden eines oder mehrere Artefakte unter Anleitung in einem Projekt angewendet. Somit werden einerseits eine „Instanz“ der zu untersuchenden Phänomene und Problemstellungen, aber auch eine „Instanz“ der entwickelten Artefakte betrachtet [JoPe14]. Der Vorteil einer Fallstudie liegt in ihrem Praxisbezug, da dies einerseits die Möglichkeit offeriert, die zu untersuchenden Phänomene und Problemstellungen in einer Tiefe zu untersuchen, welche nur die praktische Anwendung offenbaren kann, aber auch die

Konfrontation der entworfenen Artefakte mit praktischen Anforderungen ermöglicht, welche zu stark vereinfachende Annahmen verbieten [JoPe14]. Das zugrundeliegende Projekt, auch Fallbeispiel genannt, liefert dabei einen Anwendungskontext für die Artefakte, indem das Fallbeispiel die Nutzung der Artefakte motiviert und die Anforderungen an die Artefakte möglicherweise weiter spezifiziert. Durch die Anwendung der Artefakte im Kontext des Fallbeispiels werden Erkenntnisse über die Artefakte gewonnen, welche für eine Bewertung der Artefakte hinsichtlich der Anforderungserfüllung und somit auch der Erreichung der Forschungsteilziele bzw. des Forschungsziels herangezogen werden können [Wie14].

Entsprechend [Eis89] kann die Anwendung einer Fallstudie drei Ziele haben: erstens kann sie zur Beschreibung von Problemen und Problemstellungen genutzt werden, zweitens um ein Artefakt zu testen und drittens um ein neues Artefakt zu erstellen, bzw. dessen Erstellung zu motivieren. Im Rahmen der Evaluation der Artefakte dieser Arbeit werden alle drei Ziele verfolgt. Einerseits soll anhand der Fallbeispiele aufgezeigt werden, welche Phänomene und Probleme die Entwicklung einer Ontologie motivieren. Andererseits wurden solange Fallstudien durchgeführt, bis eine gewisse Reife der Artefakte erreicht wurde und keine Weiterentwicklung mehr motiviert werden konnte. Demnach wurde im Rahmen der vorliegenden Arbeit ein iteratives Vorgehen durchgeführt, welches erneut auf den Design-Zyklus nach [Wie14] führt, welcher in Abbildung I.2 zu finden ist. Hierbei findet im Rahmen des iterativen Vorgehens eine mehrmalige Evaluation statt, aus welcher jeweils ein neuer Design-Zyklus folgen kann, in welchem die Erfahrungen aus der vorherigen Evaluation eingebracht werden. In jedem Design-Zyklus wird die Erfüllung der Anforderungen und damit die Erfüllung der Forschungsteilziele bzw. die Erfüllung des Forschungsziels überprüft. Sind die Anforderungen nicht erfüllt und somit auch die Forschungsziele, werden die Ergebnisse der Anwendung der Fallstudie analysiert, um Aussagen über notwendige Weiterentwicklungen der Artefakte treffen zu können. Da ein Fallbeispiel unter Umständen nicht die Anwendung aller im Rahmen dieser Arbeit entwickelten Artefakte erfordert, muss bei der Analyse einer Fallstudie stets berücksichtigt werden, welchen Einfluss die Nichtnutzung von Artefakten auf die Fallstudie hat. Dieser Einfluss wird gemeinhin als Sensitivität eines Artefaktes bezeichnet [Wie14].

Im Rahmen der Evaluation dieser Arbeit werden Industriefallstudien angewandt. Der Vorteil einer Industriefallstudie liegt darin, dass sie die Anforderungen eines realen Projektes berücksichtigt und somit die Lösung eines realen Problems fokussiert. Somit sind die Erkenntnisse, die sich aus der Anwendung der Fallstudie ergeben, von besonderer Relevanz. Nachteilig hingegen ist, dass die im Rahmen einer Industriefallstudie erstellten Ergebnisse nicht vollständig veröffentlicht werden können, da sie u.U. betriebsinterne Informationen beinhalten. Zur Strukturierung der Durchführung der Fallstudien wurde ein an [DLA15] angelehntes Vorgehen genutzt, welches aus folgenden Schritten besteht:

1. Definition des Fallbeispiels: In diesem Schritt wird eine Spezifikation der Problemstellung vorgenommen, welche die Entwicklung einer Ontologie motiviert. Darüber hinaus wird die gewünschte Zielstellung definiert, welche nach Vorhandensein der Ontologie erreicht sein soll.

2. Planung der Fallstudie: Nachdem das Fallbeispiel definiert wurde, wird ein Projektplan entworfen, welcher einen strukturierten Weg zur Zielerreichung aufzeigen soll. Hierbei erfolgt auch eine Auswahl der im Fallbeispiel notwendigen Artefakte (Ontologie-

Anforderungserhebung, -Entwicklung (TBox), -Entwicklung (ABox)), da nicht jedes Fallbeispiel die Anwendung aller entwickelten Artefakte erfordert.

3. Anwendung entwickelter Artefakte: Während der eigentlichen Anwendung der Artefakte wird durch den Autor und die Projektpartner eine gemeinsame Anwendung der Artefakte vorgenommen und dokumentiert. Die Anwendung erfolgt dabei, um das im Rahmen der Fallstudie definierte Ziel zu erreichen.

4. Analyse der Ergebnisse der Anwendung: Die im Rahmen der Anwendung erzielten Ergebnisse werden dahingehend analysiert, inwiefern die Anforderungen an die Artefakte als erfüllt angesehen werden können. Ist dies nicht der Fall, kann sich eine erneute Iteration im Design-Zyklus ergeben.

5. Dokumentation der Anwendung: Die Ergebnisse der Anwendung der Artefakte werden so dokumentiert, dass die Ergebnisse in Zusammenhang mit der Fallstudie nachvollziehbar und soweit möglich reproduzierbar sind.

Die Abschnitt V-2.1 bis V-2.3 beinhalten folgende Aspekte in Bezug auf die oben genannten Schritte. Alle Abschnitte enthalten eine Kurzfassung der Spezifikation der Problemstellung sowie der Motivation für die Entwicklung der Ontologie anhand einer Darstellung des Fallbeispiels. Darüber hinaus werden die Ergebnisse der Durchführung der Fallstudien beschrieben. Eine Analyse der Ergebnisse der Durchführung findet in Abschnitt V-3 in Bezug auf die Forschungsziele der vorliegenden Arbeit statt.

V-1.2. Iterationen und zeitlicher Verlauf der Evaluation

Die im Rahmen dieser Arbeit angewandte Evaluationsmethodik basiert auf der Durchführung von Fallstudien in mehreren Iterationen. In jeder Fallstudie ist die Verwendung eines oder mehrerer Artefakte möglich, wobei nach jeder Iteration eine Überarbeitung der Artefakte basierend auf den Erkenntnissen der Fallstudie in Betracht gezogen wurde. Dies ist darin begründet, dass in jeder Fallstudie sowohl die Überprüfung der zu dem Zeitpunkt existierenden Anforderungen an die Artefakte erfolgte (Verifikation), als auch eine Überprüfung, ob die an die Artefakte gestellten Anforderungen den Anforderungen einer industriellen Einsatzumgebung gerecht werden (Validierung). Daraus ergibt sich, dass eine Versionshistorie notwendig ist, um den historischen Ablauf der Evaluation zu verstehen. Tabelle V.1 stellt die im Rahmen dieser Arbeit entwickelten Artefakte den durchgeführten Fallstudien gegenüber. Der Tabelle kann demnach für jedes Artefakt eine Versionshistorie entnommen werden, wobei ein Minus die Nicht-Nutzung des Artefaktes symbolisiert. Die Bezeichnung der Fallstudien bezieht sich auf das jeweils zugrundeliegende Fallbeispiel. Hierbei steht „WA“ für das industrielle Fallbeispiel „Wartungsassistenz“ (siehe Abschnitt V-2.1), „AF“ für das industrielle Fallbeispiel „automatisierte Fähigkeitsabfrage“ (siehe Abschnitt V-2.2), „PA“ für das industrielle Fallbeispiel „Prozessanalyse“ (siehe Abschnitt V-2.3). Die Aufzählung der Fallstudien bezieht sich auf den chronologischen Abschluss der jeweiligen Fallstudie (erster Abschluss links, letzter Abschluss rechts). Was der wesentliche Unterschied einer bestimmten Version im Vergleich zur finalen Version ist und warum eine entsprechende Weiterentwicklung vorgenommen wurde, ist in den jeweiligen Abschnitten der Fallstudien zu finden.

Tabelle V.1.: Versionsverlauf und Fallbeispiele im Rahmen der Evaluation

Artefakt	Fallstudien				
	WA - 1	WA - 2	PA - 1	AF	PA - 2
Methodik zur Anforderungserhebung	-	v1	v2	-	v2
Profil zur Anforderungserhebung	-	v1	v2	-	v2
Methodik zur Entwicklung von Ontologien (TBox)	v1	v2	v2	v2	-
Ontologie-Design-Muster	-	v1	v2	v2	v2
Methodik zur Modellierung von Ontologien (ABox)	-	v1	v2	v2	v3
Werkzeug zur Modellierung von Ontologien (ABox)	-	-	-	v1	v2

V-2. Darstellung der Fallstudien

In den nachfolgenden Abschnitten V-2.1 bis V-2.3 sind die im Rahmen der Evaluation genutzten Fallbeispiele zu finden, wobei die nachfolgenden Abschnitte stets in eine Beschreibung des Fallbeispiels an sich und eine Darstellung der Ergebnisse der Durchführung der Fallstudie bzw. Fallstudien aufgeteilt sind. Die dargestellten Ergebnisse wurden den jeweiligen Projektdokumentationen entnommen und aus Geheimhaltungsgründen durch Anonymisierung und Pseudonymisierung der Abbildungen und Daten aufbereitet. Da es sich bei den Projektdokumentation darüber hinaus um umfangreiche Dokumente handelt, wurden nur die für die Evaluation relevanten Ausschnitte entnommen.

V-2.1. Industrielle Fallstudie „Wartungsassistenz“

Das industrielle Fallbeispiel, welches nachfolgend vorgestellt wird, liefert die Grundlage für die Fallstudien WA-1 und WA-2 in Tabelle V.1, welche in Kooperation mit einem Komponentenhersteller „A“ durchgeführt wurden, welcher unter anderem automatisierte Schmiersysteme für Werkzeugmaschinen herstellt.

V-2.1.1. Beschreibung des Fallbeispiels

Im Fokus des Fallbeispiel steht ein automatisiertes Schmiersystem des Unternehmens „A“, welches beispielsweise in Werkzeugmaschinen zur Schmierung beweglicher Teile (z.B. Achsen) verbaut wird. Die Abgabe von Schmierstoff wird durch eine speicherprogrammierbare Steuerung auf Basis des Schmierstoffbedarfs gesteuert, wobei der Bedarf auf unterschiedlichen Einflussparametern beruht. Das Schmiersystem verfügt über eine Kartusche, welche den Schmierstoff enthält und bei Bedarf durch einen Wartungstechniker zu wechseln ist. Wird die Kartusche gewechselt, geht nicht genutzter Schmierstoff verloren. Da ein Mangel an Schmierstoff zu einer ungenügenden Schmierung und somit zu höherem Verschleiß führt, führt eine leere Kartusche u.U. zum Stillstand einer Maschine. Das Schmiersystem ist dazu in der Lage, den Füllstand einer Kartusche zu detektieren.

Innerhalb einer Werkzeugmaschine sind typischerweise mehrere Schmiersysteme dieses Typs verbaut, um Linear- und Rotationsachsen zu schmieren. Ein Wartungstechniker, welcher für das Auffüllen des Schmierstoffs zuständig ist, betreut dann in der Regel eine Vielzahl an Werkzeugmaschinen und somit eine Vielzahl an Schmiersystemen. Der Wartungstechniker muss hierzu für jede Schicht eine Route durch den Produktionsstandort anhand der zu wechselnden Kartuschen planen. Aus Effizienzgründen soll dem Wartungstechniker eine Applikation zur Seite gestellt werden, welche den aktuellen Füllstand aller Kartuschen aller Schmiersysteme anzeigt. Auf diese Weise soll der Wartungstechniker diejenigen Schmiersysteme erkennen können, welche bereits einen niedrigen Füllstand aufweisen. Da jedoch nutzungsabhängig auch über eine Schicht hinweg weitere Kartuschen hinzukommen, ist auch eine Prognose mit einzubeziehen, sodass auch vergleichsweise gut gefüllte Kartuschen mit in die Planung einbezogen werden. Darüber hinaus sind Fehlerzustände eines Schmiersystems entsprechend zu berücksichtigen und für den Wartungstechniker aufzubereiten.

Um für Wartungstechniker eine entsprechende Applikation entwickeln zu können, müssen unterschiedliche Datenquellen miteinander verknüpft werden. Zunächst sind die Daten jedes Schmiersystems, welche in Form einer SQL-Datenbank in der Maschinensteuerung vorliegen, zu berücksichtigen. Mit Hilfe dieser Daten kann eine Aussage über den aktuellen Füllstand getroffen werden. Um allerdings eine Aussage über den zukünftigen Schmierstoffverbrauch zu treffen, ist eine Verknüpfung der Füllstandsinformation mit Auftragsinformationen vorzunehmen, da diese wesentlich den Schmierstoffbedarf bestimmen. Diese Informationen sind ebenfalls in Form von relationalen Datenbanken vorhanden, jedoch in einem unterschiedlichen Datenbankschema. Die Beschreibung von Fehlerzuständen ist in Form von PDF-Dateien vorhanden, welche dem Wartungstechniker bei Bedarf anzuzeigen sind. Während in den Datenquellen eine potenziell hohe semantische Heterogenität herrscht, soll dies für die Applikation des Wartungstechnikers keine Rolle spielen. Letztere soll einen einheitlichen Zugriffspunkt für alle relevanten Informationen im Fallbeispiel erhalten.

V-2.1.2. Durchführung und Ergebnisse der Fallstudien

In Bezug auf das genannte Fallbeispiel wurden zwei Fallstudien (WA-1 und WA-2) durchgeführt. Dabei waren die Fallstudien so konzipiert, dass Erkenntnisse aus der ersten Fallstudie in die zweite Fallstudie einfließen konnten und somit ein iteratives Vorgehen sichergestellt war. Das Ziel beider Fallstudien war einerseits die Entwicklung einer TBox, welche die für den Anwendungsfall notwendigen Informationen abbilden kann und eine Verknüpfung der verschiedenen Informationsquellen zulässt. Darüber hinaus sollte die ABox alle quasi-statischen Informationen über ein real existierendes Schmiersystem einer Maschine enthalten und eine Menge von Mappings für die dynamischen Informationen (z.B. Füllstände) bereitstellen. Die Bereitstellung der Mappings übernahm das in der Fallstudie beteiligte Unternehmen, da für die Erstellung und Ausführung der Zugriff auf eine produktiv genutzte Maschinensteuerung notwendig war. Die Entwicklung der Applikation des Wartungstechnikers begann erst kurz vor Ende der Fallstudien WA-1 und WA-2. Nachfolgend sind die wesentlichen Erkenntnisse der Fallstudien WA-1 und WA-2 dargestellt. Als Kurzbezeichnung für Schmiersystem wird nachfolgend die Abkürzung „LRS“ verwendet.

V-2.1.2.1. Ontologie-Anforderungserhebung

Die Ontologie-Anforderungserhebung in den Fallstudien WA-1 und WA-2 wurde zusammen mit dem beteiligten Unternehmen im Rahmen eines jeweils eintägigen Workshops durchgeführt. Für die Anforderungserhebung in der Fallstudie WA-1 wurde das Vorgehen entsprechend [SGV09] genutzt, wobei mit Hilfe einer Excel-Vorlage und strukturierter Befragung sowie UML-Sequenzdiagrammen Kompetenzfragen und Antworten hergeleitet wurden. Dabei zeigte sich, dass ohne eine strukturierte Verknüpfung mit den Anforderungen der Applikation die Stakeholder eher dazu geneigt sind, vergleichsweise breit gefächerte Kompetenzfragen mit hoher Abstraktion zu stellen. Somit wurden relativ viele Wissensbereiche mit jeweils relativ wenigen Kompetenzfragen angesprochen. Dies wirkte sich negativ auf die TBox-Entwicklung (siehe Abschnitt V-2.1.2.2) aus, da vergleichsweise viele Informationsressourcen herangezogen werden mussten. Auch fiel es den Stakeholdern mit wenig Ontologie-spezifischer Erfahrung eher schwer, mit einer lediglich abstrakten Vorstellung der zu realisierenden Applikation die Kompetenzfragen und Antworten zu definieren. Aufgrund der oben genannten Mängel, wurde in der Fallstudie WA-2 das in dieser Arbeit beschriebene Vorgehen in einer initialen Version („v1“ in Tabelle V.1) genutzt. Die Ergebnisse werden nachfolgend vorgestellt. Zunächst wurden diverse User Stories erstellt, um einen Einstiegspunkt für die Anforderungserhebung zu definieren. Die folgende User Story wurde mit höchster Priorität bearbeitet: „Als Wartungstechniker möchte ich mich über die Zustände einer Maschine informieren, sodass ich die Materialplanung der Schmierstoffe bedarfsgerecht durchführen kann.“. Als Zweck der Ontologie wurde die Integration heterogener Informationsquellen bestimmt.

Anschließend wurde der Anwendungsfall mittels UML-Use-Case-Diagramm modelliert. Hierbei wurden die in Abbildung V.1 dargestellten Funktionen ermittelt. Demnach beinhaltet die im Fallbeispiel gewünschte Funktion „Materialplanung“ zunächst eine Funktion „Sichtung von Systeminformationen“. Diese Funktion bezieht Informationen und Daten aus der Steuerung eines Schmiersystems (z.B. Füllstände) und stellt sie im Zusammenhang der Materialplanung dar. Erweitert wird diese Funktion durch eine Funktion „Sichtung von Einflussfaktoren auf Materialverbrauch“. Diese Funktion bezieht Informationen und Daten bezüglich der Auftragsplanung einer Maschine, in der sich das Schmiersystem befindet, um den zukünftigen Verbrauch von Schmierstoff abschätzen zu können. Die Abschätzung selbst wird durch die Funktion „Bedarfsprognose“ wahrgenommen.

Im Anschluss wurde eine Modellierung der Funktionsstruktur mittels UML-Profil für Funktionsstrukturdiagramme durchgeführt. Die Ergebnisse sind in Abbildung V.2 dargestellt. Die Funktion „Materialplanung“ wurde dabei als zentrale Funktion modelliert, welche die drei anderen Funktionen beinhaltet. Da der „Bezug von Einflussfaktoren“ sowie die „Bedarfsprognose“ als erweiternde Funktionen in Abbildung V.1 modelliert wurden, wurden die Outputs aller Sub-Funktionen der Materialplanung dem Nutzer zur Verfügung gestellt. Wie in der Abbildung dargestellt, besitzt der Wartungstechniker einige Attribute, welche als Nutzungskontext aufgefasst werden. Darüber hinaus übergibt der Nutzer eine Menge von Informationen (z.B. Ansicht einer bestimmten Maschine oder Halle) an die Materialplanungsfunktion. Die Funktion „Bezug von Systeminformationen“ nimmt diese Eingangsinformationen an und bezieht für eine Menge an Schmiersystemen (kurz „LRS“) die für die Materialplanung relevanten Informationen wie Füllstände, Art des Schmierstoffs etc. Diese Informationen werden an den Nutzer übergeben, aber auch an die Funktionen „Bezug von Einflussfaktoren“ und „Bedarfs-

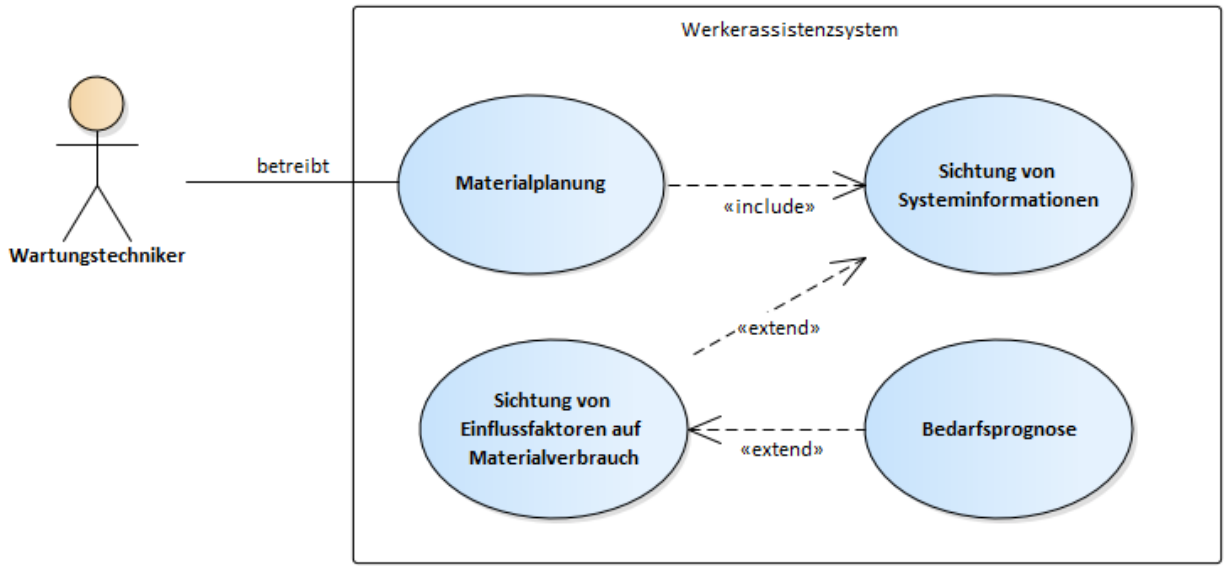

Abbildung V.1.: Modellierung des Anwendungsfalls der Fallstudie WA

prognose“. Erstere bezieht für eine Menge an Maschinen, in denen Schmiersysteme verbaut sind, die Auftragsinformationen aus einem Manufacturing-Execution-System. Diese Auftragsinformationen werden zusammen mit den LRS-Informationen der Bedarfsprognosefunktion übergeben, welche die Prognose an den Wartungstechniker übergibt. Wie der Abbildung zu entnehmen ist, wurden zwei Punkt ontologischen Interesses definiert: Die LRS-Informationen und die Auftragsinformationen.

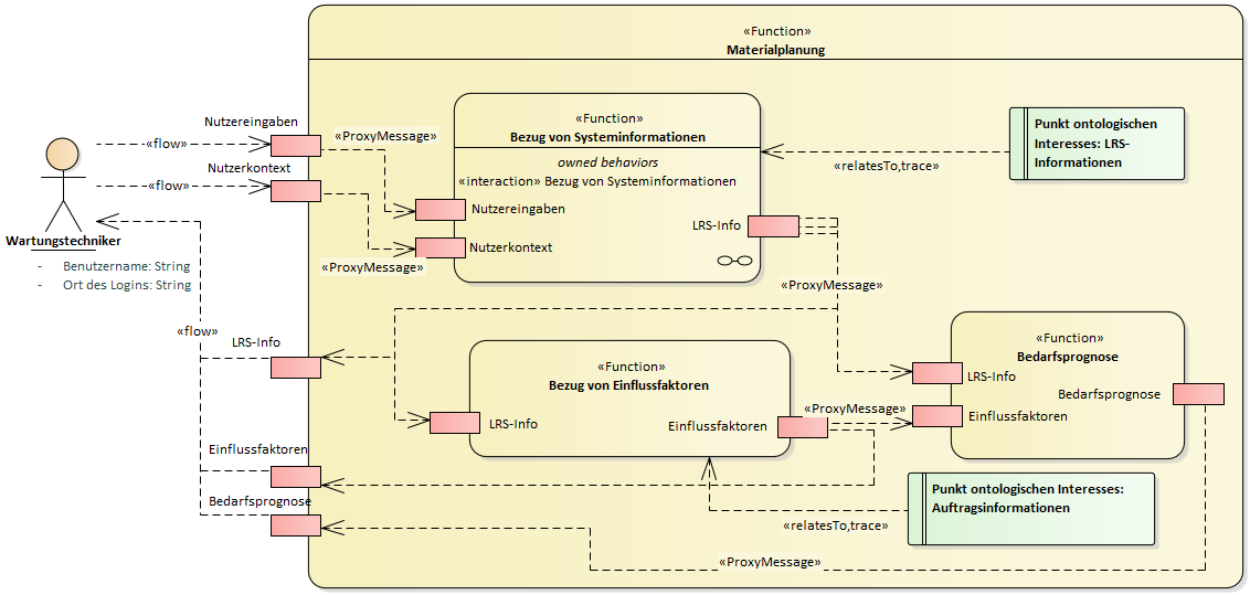

Abbildung V.2.: Modellierung der Funktionsstruktur des Fallbeispiels WA

Die Modellierung des Funktionsverhaltens erfolgte nicht mit dem UML-Profil entsprechend II-4.3.3, sondern in einer früheren Variante („v1“ in Tabelle V.1). Daher wird an

dieser Stelle auf eine Darstellung verzichtet. Zwar erfolgte eine Modellierung eines UML-Sequenzdiagramms, jedoch wurden die funktionalen ontologischen Anforderungen darin nicht, wie in Abschnitt II-4.4.1 gezeigt, integriert. Es erfolgte jedoch eine Modellierung der Systemgruppenstruktur, was in Abbildung V.3 dargestellt ist. Demnach interagiert der Wartungstechniker mit dem Frontend eines Werkerassistenzsystems. Das Frontend bezieht die darzustellenden Daten von einem Backend. Letzteres besitzt eine Menge von SPARQL-Abfragen, um in einer Graph-Datenbank anzufragen, in welcher sich TBox und ABox der zu entwickelnden Ontologie befinden. Die Graph-Datenbank soll dynamische Informationen wie Auftragslage (aus MES) oder Zustände des Schmiersystems (aus SPS) mit Hilfe von Mappings virtualisieren. Für die Wartung relevante technische Dokumentationen sollen durch das Backend direkt von einem Online-Dateispeicher bezogen werden. Die Informationen darüber, wo sich die Dokumentation befindet, soll ebenfalls in der ABox zu finden sein. Die Systemgruppe wurde als geschlossen definiert.

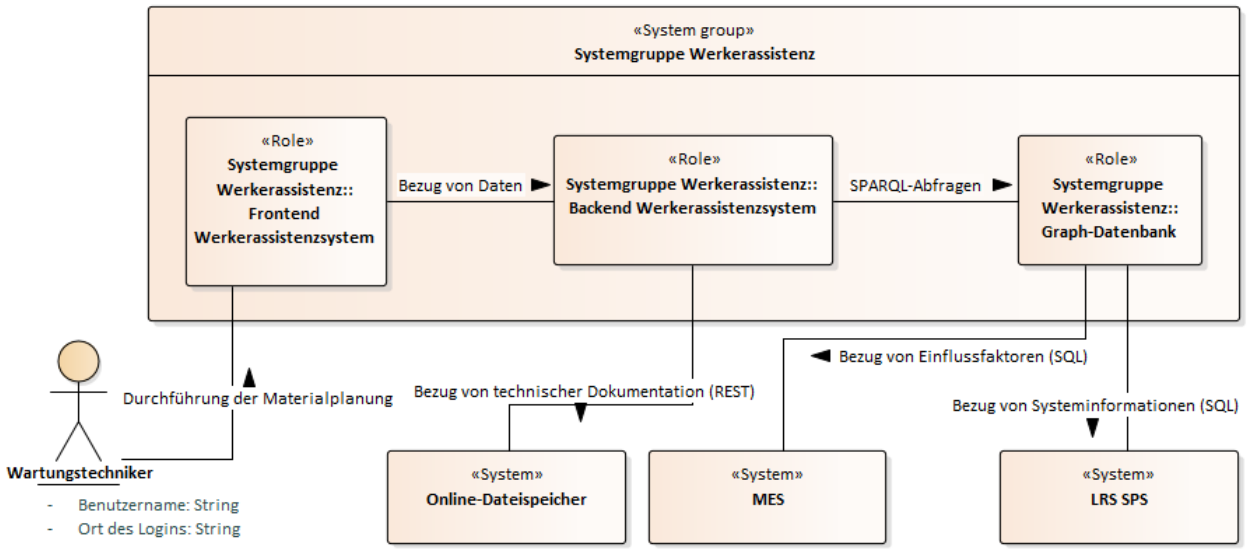

Abbildung V.3.: Modellierung der Systemgruppenstruktur des Fallbeispiels WA

Auf dieser Basis erfolgte eine Modellierung der ontologischen Anforderungen, wobei nicht das UML-Profil entsprechend Abschnitt II-4.4.1 genutzt wurde. Stattdessen wurden ein SysML-Anforderungsdiagramm genutzt, um die in Abbildung V.2 gezeigten Punkte ontologischen Interesses mittels Kompetenzfragen und Antworten näher zu beschreiben. Ein Ausschnitt dieses Diagramms ist in Abbildung V.4 gezeigt. Der Nachteil der Verwendung von SysML-Anforderungen zur Modellierung der Kompetenzfragen und Antworten ist, dass der Stereotyp „SysML-Anforderung“ nicht in ein Sequenzdiagramm integriert werden kann. Es kann somit basierend auf den bisher dargestellten Diagrammen keine einzelne ontologische Anforderung in den Kontext des Verhaltens der zu entwickelnden Applikation gesetzt werden. Darüber hinaus ist eine Nachverfolgbarkeit der ontologischen Anforderungen nur auf Ebene der Punkte ontologischen Interesses möglich, da eine Einbindung in anderen Diagrammen nicht möglich ist. Diese methodische Schwäche führte zur Weiterentwicklung („v2“ in Tabelle V.1) der Methode zur Ontologie-Anforderungserhebung. Auf diese Art wurden 37 Kompetenzfragen und Antworten modelliert, wovon 12 mit hoher Priorität weiter verfolgt wurden. Vier der hoch priorisierten Anforderungen sind in Abbildung V.4 dargestellt. Für

die in den Anforderungen eins und zwei angesprochenen Informationen bzw. Daten wurde eine Dynamik im Sekundenbereich definiert. Die in den Anforderungen drei und vier angesprochenen Informationen wurden als quasi-statisch definiert.

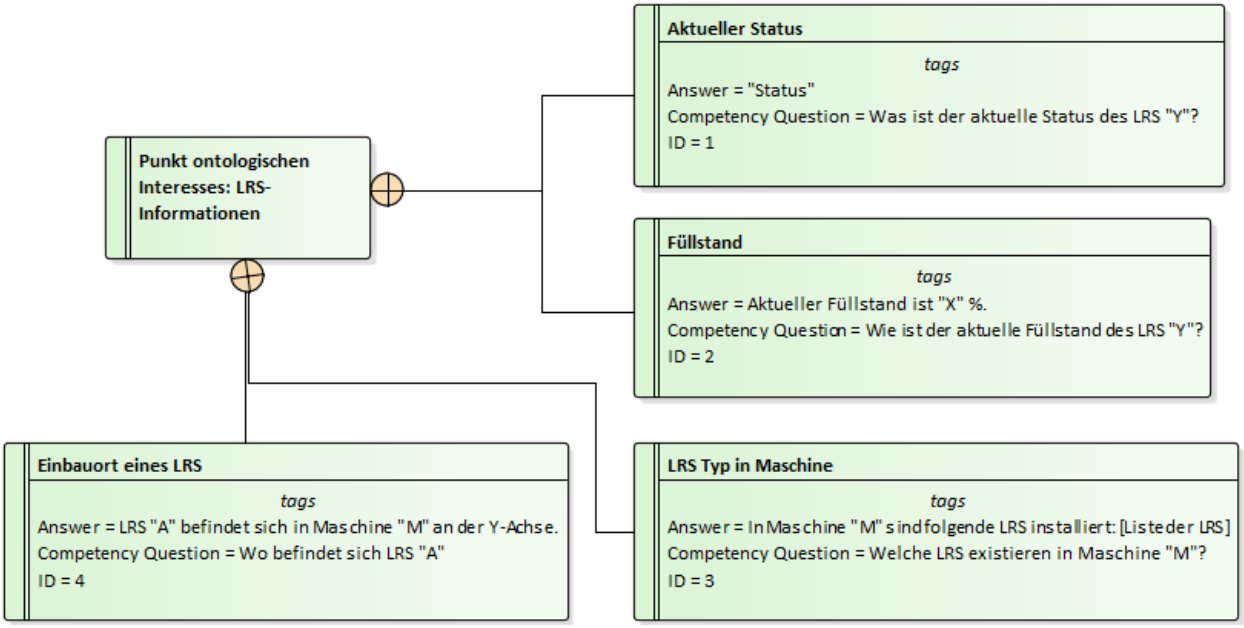

Abbildung V.4.: Ontologische Anforderungen des Fallbeispiels WA

V-2.1.2.2. Ontologie-Entwicklung TBox

Im Rahmen der TBox-Entwicklung wurde in der Fallstudie WA-1 das in dieser Arbeit beschriebene Vorgehen in einer initialen Version angewendet. Im Rahmen der Suche nach Informationsressourcen wurden fünf Ressourcen identifiziert. Die [DIN EN 62264 -1] für die Beschreibung von Auftragsinformationen, die [ISO 10303-44] für die Beschreibung von Produktinformationen, die [DIN EN 61360-1] für die Beschreibung von Merkmalen und die [ANSl/ISA 88]) für die Beschreibung von Zuständen. Darüber hinaus wurde das Begriffsmodell des Forschungsprojektes „SemAnz40“ entsprechend [Hil+17*a] herangezogen, um eine Verknüpfung der Standards vorzunehmen.

Die in Abschnitt III-3.2.1 beschriebene funktionale Eignung wurde im Anschluss geschätzt. Die beteiligten Domänenexperten attestierten den Informationsressourcen eine qualitative funktionale Eignung. Es wurde somit angenommen, dass die Informationsressourcen für den Anwendungsfall geeignet sind. Die quantitative funktionale Eignung wurde geschätzt. Die begriffliche Abdeckung war durch die genannten Informationsressourcen gegeben, jedoch wurde eine geringe Präzision erkannt. Die Informationsressourcen beinhalteten also wesentlich mehr Begriffe und Relationen als für das Fallbeispiel notwendig waren. Es wurde angenommen, dass dies durch die in der Fallstudie WA-1 vergleichsweise breit gefächert Kompetenzfragen begründet ist. Somit wurden lediglich einzelne Teile vieler Informationsressourcen bzw. Wissensbereiche angesprochen. Basierend auf den gefundenen Informationsressourcen wurde das in Abschnitt III-3.2 beschriebene Vorgehen angewandt. Auf eine Darstellung der einzelnen Lightweight-Ontologien soll jedoch an dieser Stelle verzichtet werden, da sie lediglich

initiale Versionen der in Abschnitt III-4 definierten ODMs darstellen oder im Rahmen der Fallstudie WA-2 nicht mehr herangezogen wurden (siehe unten). Abbildung V.5 stellt die Übersicht über das Ontologie-Alignment aus Fallstudie WA-1 dar. Erkennbar ist, dass jede Informationsressource als eigenständiges Ontologie-Design-Muster definiert wurde.

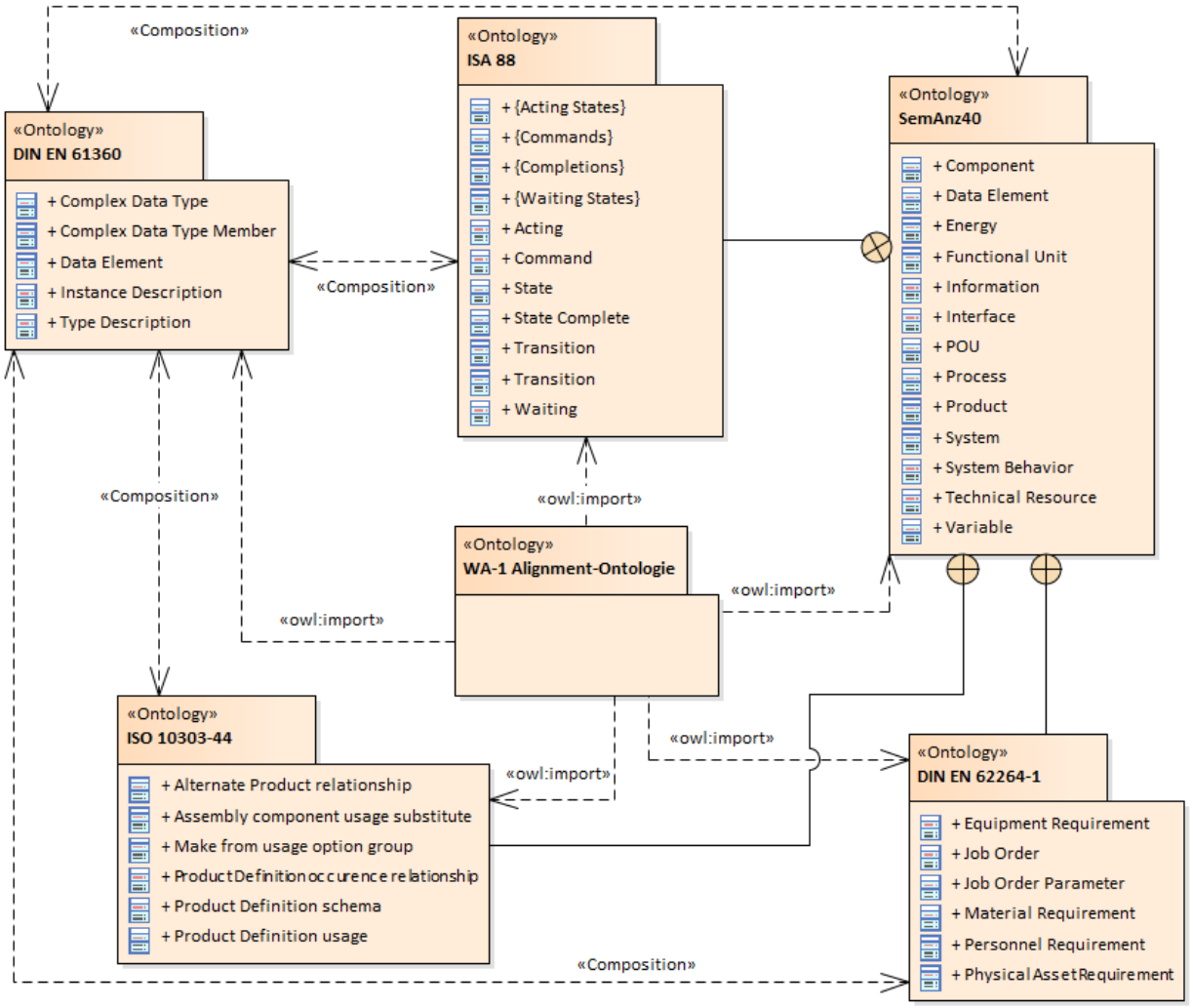

Abbildung V.5.: Übersicht Ontologie-Alignment aus Fallstudie WA-1

Allerdings wurde der Mustergedanke nicht konsequent, sondern eher als Leitfaden angewendet, was im Fehlen des in Definition III.3 beschriebenen Designziels begründet ist. Dies macht einen der Unterschiede der ersten Version („v1“) zur zweiten Version („v2“ in Tabelle V.1) der Methodik zur TBox-Entwicklung aus. Das Fehlen des Designziels resultierte darin, dass mehrere Informationsressourcen teilweise pragmatisch zusammengefasst wurden, um dem Fallbeispiel zu entsprechen. Beispielsweise ist die Darstellung aus dem Projekt „SemAnz40“ entsprechend [Hil+17*a] bereits eine Zusammenfassung mehrere Informationsressourcen und wurde als einzelnes ODM modelliert anstatt die zugrunde liegenden Informationsressourcen in seperate ODMs zu überführen. Auch wurde die [DIN EN 62264 -1] nicht vollständig in einen ODM überführt, sondern lediglich die für den Anwendungsfall notwendigen Bestandteile. Für die laufende Fallstudie stellte dies kein Problem dar, wirkte sich jedoch bereits auf die folgende Fallstudie negativ aus (s.u.). Basierend auf den erstellten Lightweight-Ontologien

wurde eine Heavyweight-Ontologie entsprechend Abschnitt III-3.3 erstellt, die in Summe 962 Axiome aufwies. Auf eine detaillierte Darstellung wird mangels weiterer Erkenntnisse verzichtet.

Da aus der Ontologie-Anforderungserhebung in der Fallstudie WA-2 wesentlich detaillierte Anforderungen hervorgingen, musste die in WA-1 erstellte TBox in der Fallstudie WA-2 überarbeitet werden. Dies war insbesondere darin begründet, dass das detaillierte Vorgehen zur Ontologie-Anforderungserhebung dazu führte, dass wesentlich tiefer gehende Anforderungen erhoben wurden. Zunächst wurde das in Definition III.3 beschriebene Designziel angewandt, weshalb die im ODM „SemAnz40“ entsprechend [Hil+17*a] zusammengefassten Informationsressourcen einzeln betrachtet wurden. Es fand eine Bewertung der funktionalen Eignung und der Nutzbarkeit aller bisher gefundenen Informationsressourcen statt, wobei die Kriterien für die Bestimmung der Implementierungskomplexität geschätzt wurden. Es wurde bewertet, dass die [DIN EN 62264 -1] im Vergleich zur [VDI/VDE 3682-1] bei gleicher funktionaler Eignung eine höhere Implementierungskomplexität aufweist. Dies ist im enormen Umfang des Standards [DIN EN 62264 -1] begründet. Darüber hinaus wurde bewertet, dass die [ISO 10303-44] einen im Vergleich zur [VDI 2206] höheren begrifflichen Overhead aufweist. Zwar lag die [ISO 10303-44] bereits als ODM vor, es wurde dennoch die [VDI 2206] bevorzugt, da der geringere begriffliche Overhead als wichtiger Einflussfaktor auf die Effizienz der Wartung erachtet wurde. Basierend auf dieser Bewertung wurden zwei ODMs aus der Fallstudie WA-1 nicht mehr benötigt und es wurden zwei weitere ODMs basierend auf [VDI/VDE 3682-1] und [VDI 2206] entwickelt. Die entwickelten Ontologie-Design-Muster wurden in Abschnitt III-4 dargestellt und wurden auch in den folgenden Fallstudien AF und PA angewandt. Die genutzte Alignment-Ontologie entspricht der in Abschnitt III-4.5 dargestellten Alignment-Ontologie mit Ausnahme einiger weniger projektspezifscher Anpassungen. Mit Hilfe des VDI 3682 ODM wurde der Schmierprozess modelliert, sowie die eingehenden Schmierstoffe. Mit Hilfe des ISA 88 ODM wurde der Zustandsautomat des Schmiersystems modelliert, sodass einerseits der aktuelle Füllstand erfragt werden kann, aber auch, ob ein Schmiersystem sich aufgrund eines Mangels an Schmierstoff in einem Fehlerzustand befindet. Mit Hilfe des VDI 2206 ODMs wurden die Bestandsbeziehungen modelliert, also in welcher Maschine sich ein bestimmtes Schmiersystem befindet. Die Merkmale, wie z.B. der Füllstand, wurden mit Hilfe des DIN EN 61360 ODM modelliert. Die Modellierung von Aufträgen wurde ebenfalls mit Hilfe des VDI 3682 ODM durchgeführt. Die Ontologie-Design-Muster sind, wie in Abschnitt III-4 erläutert, auf GitHub veröffentlicht [HiKö19@a]. Wäre in der ersten Fallstudie gleich eine konsequente Überführung der identifizierten Informationsressourcen in ODMs vorgenommen worden, wäre eine noch höhere Wiederverwendungsquote und ein geringerer Wartungsaufwand anzunehmen gewesen. Diese Erkenntnis wurde in allen folgenden Fallstudien aufgegriffen.

V-2.1.2.3. Ontologie-Entwicklung ABox

Zur Verifikation der Anforderungserfüllung in der Fallstudie WA-1 war die Definition einer exemplarischen ABox notwendig. Zu diesem Zeitpunkt existierte keine Version der im Rahmen dieser Arbeit entwickelten Methode zur ABox-Entwicklung. Daher wurde ein einfaches ad-hoc Vorgehen angewandt, wobei im Tool Protégé [Sta19@] manuell modelliert wurde. Zwar führte das Vorgehen zum Ziel, erwies sich jedoch als extrem aufwendig und fehleranfällig, da

auch inkonsistente Modelle erzeugt werden können, die erst nach der Modellierung auf Konsistenz geprüft werden können. Insgesamt wurden auf diese Weise 364 Tripel in der ABox modelliert.

Im Zuge der ABox-Entwicklung in der Fallstudie WA-2 wurde das in dieser Arbeit beschriebene Vorgehen in einer initialen Version genutzt. Die Version unterschied sich insofern vom in dieser Arbeit präsentierten Stand, als dass die für jeden ODM erstellten assertionalen Muster (siehe Listing C.1 bis C.7 im Anhang C) manuell, also ohne jegliche werkzeugtechnische Unterstützung, modifiziert und in der Vorgehensweise entsprechend Abbildung V.6 ausgeführt wurden. Das bedeutet, dass zunächst eine Menge von ausgefüllten assertionalen Mustern (SPARQL-Updates) des VDI 3682 ODMs durch einen Ontologie-Experten erstellt und anschließend ausgeführt wurde. Das ist darin begründet, dass ein assertionales Muster in der Regel mehrfach mit unterschiedlichen Variablen ausgeführt werden muss, um bspw. mehrere Prozesse anzulegen. Das gleiche wurde für Systemstrukturen, Zustandsautomaten sowie Merkmale durchgeführt. Auf diese immer noch recht aufwändige Weise wurden 752 Tripel erzeugt. Die Konsistenz der Ontologie wurde in diesem Ansatz durch die eingesetzte Graph-Datenbank sichergestellt, welche vor Ausführung jeder SPARQL-Update-Operation überprüfte, ob die Ausführung die Konsistenz gefährdet. Die Kohärenz der ABox hingegen musste manuell durch die angesprochene Sequenz der Ausführung der SPARQL-Update-Operationen sichergestellt werden.

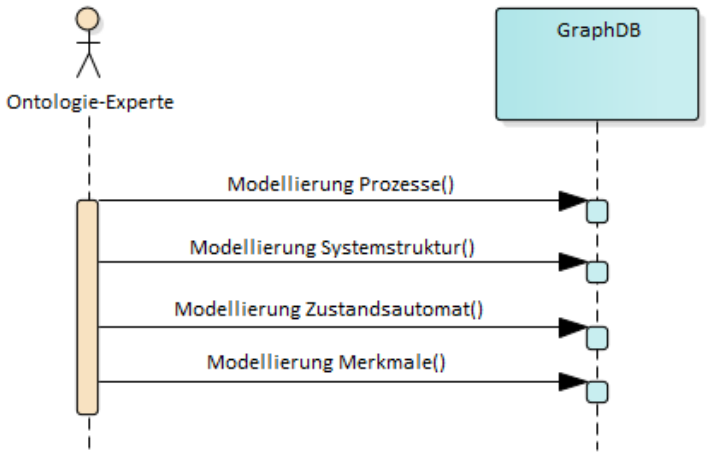

Abbildung V.6.: Ablauf der ABox-Modellierung in der Fallstudie WA-2

Im Rahmen der Verifikation der Fallstudie WA-2 entsprechend Abschnitt III-3.3.2 erwies sich die entwickelte Ontologie als vollständig und konsistent. Die Vollständigkeit wurde durch die Definition von SPARQL-Abfragen für die im Rahmen der Anforderungserhebung definierten Kompetenzfragen und Antworten ermittelt. Insgesamt wurden 27 SPARQL-Abfragen für 12 Kompetenzfragen erstellt. Listing V.1 stellt eine exemplarische SPARQL-Abfrage dar, welche für die Erfüllung von Anforderung 2 herangezogen wurde. Demnach wird im Beginn der Abfrage zunächst ein gesuchtes LRS definiert und die entsprechende Variable `?LRS_Instance` gebunden. Da der Füllstand als Merkmal des DIN EN 61360 ODM modelliert wurde, bezieht sich der Rest der Abfrage auf die Suche nach dem `?Value` der Instanz des Datenelements, welches für den Füllstand steht. Die Abfrage wurde an eine vom Unternehmen bereitgestellte Graph-Datenbank gesendet, welche Mappings ausführte, um die Daten aus der SQL-

Datenbank der speicherprogrammierbaren Steuerung zu beziehen und dann in der Rückgabe der SPARQL-Abfrage entsprechend zu berücksichtigen. Tabelle V.2 stellt eine exemplarische Rückgabe der Abfrage dar. Der `?Value` wurde hierbei durch ein R2RML-Mapping virtualisiert.

Listing V.1: SPARQL-Abfrage zur Erfüllung der Anforderung 2 aus Fallstudie WA

```
PREFIX DE6: <http://www.hsu-ifa.de/ontologies/DINEN61360#>
PREFIX a: <http://a_Knowledge_Base#>

# ID = 2: Was ist der aktuelle Fuellstand des LRS "LRS-1"?

SELECT ?LRS_Instance ?DataElementDefinition ?Value
{
    # Definiere ein bestimmtes LRS
  BIND(IRI("http://a_Knowledge_Base#LRS-1") AS ?LRS_Instance).

    # Suche den Fuellstand mit dem Aussageziel "Aktueller Wert"
    ?LRS_Instance DE6:has_Data_Element ?FillingLevelDE.
    ?FillingLevelDE DE6:has_Type_Description ?FillingLevelDET;
                    DE6:has_Instance_Description ?FillingLevelDEI.

    ?FillingLevelDET DE6:Code "DE-Code-DE-ABC123";
                    DE6:Version_Number "1.0";
                    DE6:Definition ?DataElementDefinition;
                    DE6:has_Data_Type ?anyDT;
                    DE6:Preferred_Name ?DataElementType.

    ?FillingLevelDEI DE6:has_Data_Type ?anyDT;
                DE6:Expression_Goal "Actual_Value".

    ?anyDT DE6:Value ?Value.
}
```

Tabelle V.2.: Exemplarische Rückgabe zu Listing V.1

?LRS_Instance	?DataElementDefinition	?Value
http://a_Knowledge_Base#LRS-1	Indicates how much of a certain volume is used or occupied, thus filled	90.123

In Anhang C sind zwei weitere exemplarische SPARQL-Abfragen (siehe Listing C.8 und C.9) dargestellt, welche sich auf die Anforderungen 1 und 3 aus Abbildung V.4 beziehen. Bei Abschluss der Fallstudie WA-2 wurde insbesondere die Weiterentwicklung der ABox-Entwicklungsmethode als größtes Optimierungspotential der Gesamtmethode betrachtet.

V-2.2. Industrielle Fallstudie „automatisierte Fähigkeitsabfrage“

Das industrielle Fallbeispiel, welches nachfolgend vorgestellt wird, liefert die Grundlage für die Fallstudie AF in Tabelle V.1, welche in Kooperation mit dem Unternehmen „B“ durchgeführt wurde.

V-2.2.1. Beschreibung des Fallbeispiels

Im Fokus des vorliegenden Fallbeispiels steht der Wunsch nach einer steigenden Flexibilität in Auftragsbearbeitungsprozessen. Hierzu soll die heute weitgehend manuelle Prüfung der Herstellbarkeit eines Produktes und die Zuordnung zu Ressourcen in Form eines Arbeitsplanes automatisiert werden. Die Automatisierung dieser Prüfung ist insbesondere dann von Interesse, wenn aufgrund wechselnder Produktanforderungen und kleinen Losen eine häufige Anzahl von Prüfungen durchgeführt werden muss, welche im Verhältnis zur eigentlichen Produktionszeit einen hohen Anteil der gesamten Auftragsbearbeitungszeit einnehmen.

Ansätze zum „fähigkeitsbasierten Engineering“ gelten für diesen Zweck als vielversprechend, da sie ein hohes Maß an Flexibilität durch situationsspezifische Verknüpfung von Produkten, Prozessen und Ressourcen bieten. Eine Fähigkeit ist in diesem Zusammenhang die Möglichkeit (einer Produktionsressource), eine Transformation bzw. einen Prozess auszuführen, wobei ein fähigkeitsbasierter Ansatz auf einer formalisierten Beschreibung der Funktionen von Produktionsressourcen und der Anforderungen von Produkten sowie Regeln zum automatisierten Abgleich von Produktanforderungen mit Ressourcenfunktionen basiert [Jär12]. Der Abgleich zwischen den von Ressourcen angebotenen und für die Produktion von Produkten erforderlichen Fähigkeiten erfolgt dann über den Abgleich der Fähigkeitsbeschreibung und ggf. deren Parametern mit den Anforderungen, welche das Produkt definiert.

V-2.2.2. Durchführung der Fallstudie

Im Rahmen der Fallstudie sollten die Wissensbereiche, die sich für einen Abgleich zwischen Produkten und Produktionsressourcen eignen, modelliert werden und somit eine TBox erstellt werden, welche die für Herstellbarkeitsprüfungen relevanten Wissensbereiche abdeckt. Des Weiteren sollte eine ABox im Rahmen der Fallstudie entwickelt werden, welche die Fähigkeiten zweier Produktionsressourcen (ein Bohr- und ein Fräsautomat) beinhaltet. Als besondere Herausforderung ist hier die Nicht-Existenz von Engineering-Artefakten der Produktionsressourcen zu nennen, wodurch sich eine hohe Anforderung an die Effizienz der ABox-Entwicklung ergibt. Darauf aufbauend sollte eine Vergleichsmethode entwickelt werden, welche zwischen Fähigkeiten von Produktionsressourcen und Produktanforderungen einen entsprechenden Abgleich durchführt und somit die Herstellbarkeit prüft. Dabei sollte unter anderem die Abfragesprache SPARQL genutzt werden, wobei die Entwicklung der Vergleichsmethode nicht Bestandteil der Fallstudie ist. Nachfolgend wird der Begriff „Capability“ für eine „abstrakte Fähigkeit“ und „Skill“ für eine „ausführbare Fähigkeit“ verwendet.

V-2.2.2.1. Ontologie-Anforderungserhebung

Hinsichtlich der Methodik zur Anforderungserhebung wurde zwar die im Rahmen dieser Arbeit vorgeschlagene Methodik angewandt, wurde jedoch nicht repräsentativ (unter Einbeziehung aller relevanten Stakeholder) durchgeführt. Die Ergebnisse der Durchführung der Anforderungserhebung gehen somit nicht in die Bewertung mit ein und werden daher auch nicht ausführlich dargestellt, siehe auch Tabelle V.1. Exemplarisch sei folgende User Story genannt: „Als ein Anlagenbetreiber möchte ich die Fähigkeit einer neuen Maschine in das Skill-basierte Plannungsprogramm aufnehmen können, sodass ich neue Skills anbieten kann und das Planungsprogramm weiterverwenden kann.“ Für das beschriebene Fallbeispiel wurde der Zweck „Spezifikation von Systembeschreibungen“ definiert, da die Ontologie für die Beschreibung von Fähigkeiten im Sinne einer Systembeschreibung eingesetzt werden sollte. Im Rahmen der Anforderungserhebung wurden 12 Kompetenzfragen und Antworten aufgenommen. Als exemplarisch für die Fallstudie können die Kompetenzfragen und Antworten in Tabelle V.3 herangezogen werden.

Tabelle V.3.: Exemplarische Kompetenzfragen und Antworten

ID	Kompetenzfrage	Antwort
1	Welche Capabilities besitzt Modul „X“?	Liste von Capabilities des Moduls „X“, optional mit In- und Outputs.
2	Wie kann der Skill „S“ von Modul „X“ gestartet werden?	Technische Informationen die für den Start des Skills notwendig sind.

V-2.2.2.2. Ontologie-Entwicklung TBox

Im Rahmen der TBox-Entwicklung wurden zunächst Wissensbereiche identifiziert, welche für die Modellierung von Capabilities und Skills relevant sind. Diese sind entsprechend [Köc+20*]:

- Maschinenstruktur: Dieser Wissensbereich beinhaltet Begriffe und Relationen zur strukturellen Beschreibung technischer Systeme, bspw. Baugruppenbeziehungen.
- Capabilities: Dieser Wissensbereich beinhaltet Begriffe und Relationen zur Modellierung von Capabilities, welche eine technologieunabhängige Darstellung der Fähigkeiten eines Systems darstelen.
- Skills: Dieser Wissensbereich beinhaltet Begriffe und Relationen zur Modellierung der notwendigen Informationen zur Ausführung einer bestimmten Capability.

Für den Wissensbereich der Maschinenstruktur wurde der ODM der [VDI 2206] herangezogen. Für den Wissensbereich der Capabilities wurde der ODM der [VDI/VDE 3682-1] herangezogen und um den Standard [DIN 8580] zur Beschreibung von Fertigungsverfahren ergänzt. Für den Wissensbereich der Skills wurde die W3C Submission [Had09@] herangezogen, welche die „Web Application Description Language“ darstellt und zur Beschreibung von RESTful-Webservices genutzt werden kann. Als weiterer ODM wurde der ODM zur Beschreibung von Merkmalen basierend auf der [DIN EN 61360-1] herangezogen. Damit erfolgte die

Wiederverwendung von drei in der Fallstudie WA-2 entwickelten ODMs. Die Informationsressourcen wurden durch eine Expertenschätzung als funktional geeignet und nutzbar bewertet. Die Modellierung einer Lightweight-Ontologie war lediglich für die auf [Had09@] beruhende „Web Application Description Language“ notwendig, welche in [Köc+20*] dargestellt ist. Die [DIN 8580] wurde nicht als Lightweight-Ontologie dargestellt, da dies im Rahmen der Fallstudie aufgrund der umfangreichen Hierarchie von Fertigungsverfahren des Standards als nicht zielführend erachtet wurde. Die resultierende Lightweight-Alignment-Ontologie weicht von dem Ergebnis aus der Fallstudie WA-2 ab und ist in Abbildung V.7 dargestellt. Zur Steigerung der Übersichtlichkeit wurden die UML-Klassen und AML-Assoziationen um die jeweils genutzten Namensräume ergänzt.

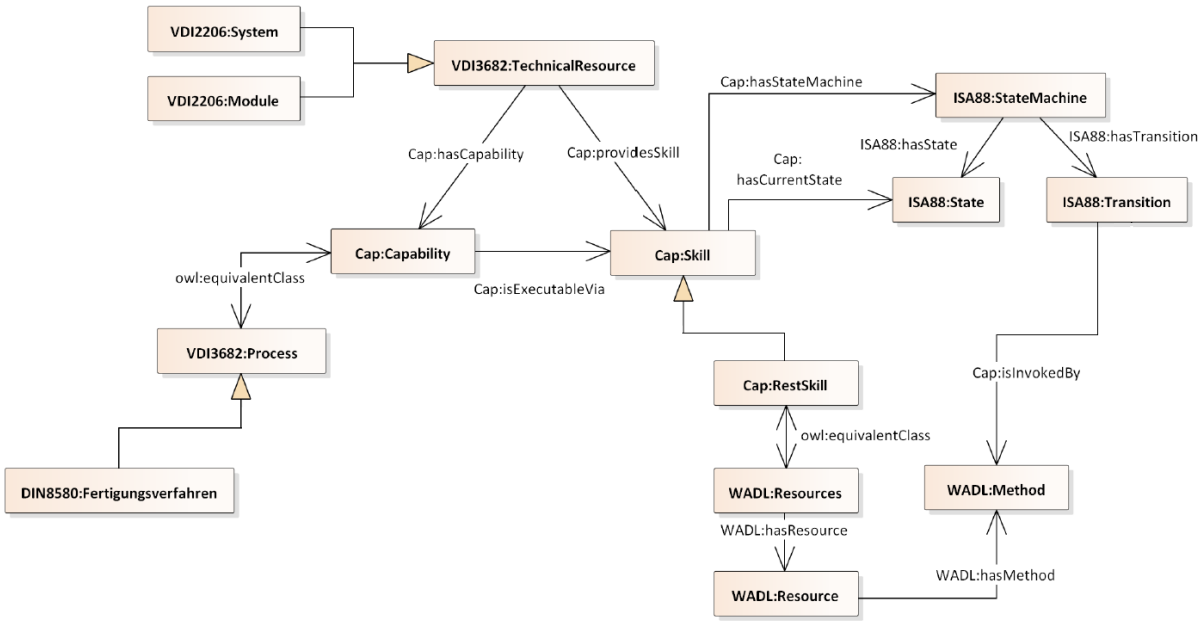

Abbildung V.7.: Lightweight-Alignment-Ontologie aus Fallstudie AF in Anlehnung an [Köc+20*]

Entsprechend [Köc+20*] nutzt die Lightweight-Alignment-Ontologie folgende Begriffe und Relationen. Zur Beschreibung der Tatsache, dass Capabilities durch Prozesse des VDI 3682 ODMs beschrieben werden, wird der Begriff `Cap:Capability` eingeführt und als Äquivalent zu `VDI3682:Process` definiert. Um eine nähere Beschreibung dieser Capabilities zu erzielen, werden die Fertigungsverfahren der [DIN 8580] als Spezialisierung des `VDI3682:Process` definiert. Der Begriff `Cap:Capability` und `VDI3682:TechnicalResource` wurden über die Relation `Cap:hasCapability` miteinander in Beziehung gesetzt, um anzuzeigen, dass eine `VDI3682:TechnicalResource` eine Capability bereitstellt. Die Begriffe `VDI2206:System` und `VDI2206:Module` werden als Spezialisierungen der `VDI3682:TechnicalResource` aufgefasst. Darüber hinaus stellt die Relation `Cap:providesSkill` dar, dass eine `VDI3682:TechnicalResource` einen Skill (`Cap:Skill`) bereitstellt. Der Begriff `Cap:RestSkill` wird als Spezialisierung eines `Cap:Skill` eingeführt und stellt die technologische Implementierung des Skill dar. Der aktuelle Zustand eines Skills wird mit den Begriffen des ISA 88 ODMs beschrieben, ebenso wie die Möglichkeit die Zustände über eine Transition zu ändern. Alle genannten ODMs sind Online veröffentlicht [HiKö19@a], ebenso wie die vollständige Heavyweight-Ontologie [Köc20@]. Hinsichtlich der

Wiederverwendung erstellter ODMs aus denn Fallstudien WA-1 und WA-2 konnten 446 Axiome direkt wiederverwendet werden (DIN EN 61360 ODM, VDI 3682 ODM, VDI 2206 ODM). Der auf [DIN 8580] basierende ODM (635 Axiome) wurde neu entwickelt. Dies gilt auch für den auf [Had09@] basierenden ODM (138 Axiome) für die „Web Application Description Language“. Die Methodik wurde durch die beteiligten Stakeholder als zielführend erachtet und es ergaben sich keine weiteren, wesentlichen Erkenntnisse hinsichtlich der Notwendigkeit einer Weiterentwicklung.

V-2.2.2.3. Ontologie-Entwicklung ABox

Wie in der Beschreibung des Fallbeispiel erwähnt, kam der Entwicklung der ABox eine wesentliche Rolle im Rahmen der Fallstudie zu. Für die ABox-Entwicklung wurde die im Rahmen der Arbeit vorgestellte Methodik verwendet, jedoch ohne Berücksichtigung der Metrik zur Strategieauswahl entsprechend IV-3.2, da im Rahmen der Fallstudie keinerlei Artefakte für Mappingansätze verfügbar waren. Demnach war lediglich eine semi-automatisierte ABox-Erstellung möglich. Der Ansatz für die semi-automatisierte Erstellung der ABox basierte auf den Erkenntnissen der Fallstudien WA-1 und WA-2, weshalb eine werkzeugtechnische Unterstützung entsprechend dem in dieser Arbeit vorgestellten Ansatz entwickelt wurde.

Im Rahmen der Fallstudie wurde für die Modellierung das in Abbildung V.8 gezeigte Vorgehen genutzt. Demnach wurde zunächst eine Modellierung der Capabilities vorgenommen. Dies wurde mittels des Werkzeugs „LIOnS“ vorgenommen, welches entsprechend Abschnitt IV-3.3.5 die assertionalen Muster der ODMs nutzt, um die Nutzereingaben bestimmten variablen Entitäten zuzuweisen und die assertionalen Muster anschließend in einer Graph-Datenbank auszuführen. Für die Modellierung der Capabilities wurde eine tabellarische Eingabemaske genutzt, welche in Abbildung B.7 im Anhang B exemplarisch dargestellt ist. Die Modellierung einer Capability erfolgte hierbei lediglich durch die Modellierung des VDI 3682 ODM. Durch die genutzte Alignment-Ontologie, welche eine Capability (`Cap:Capability`) als Äquivalent zu einem Prozess (`VDI3682:Prozess`) definiert, wird diese Verknüpfung durch die TBox sichergestellt und gilt für alle angelegten Prozesse. Die Modellierung der Systemstrukturen, des Zustandsautomaten und der Merkmale erfolgte analog in LIOnS durch eine tabellarische Eingabemaske. Bis zu diesem Schritt war die gesamte Modellierung durch einen Domänenexperten möglich, da lediglich die Kenntnis der zugrundeliegenden Standards und das Modellierungsziel (Capabilities) bekannt sein mussten. Die Modellierung der Skills wurde durch einen Software-Entwickler vorgenommen, welcher das notwendige Wissen über die Implementierung der Capability besitzt. Auch hierfür wurde das Werkzeug „LIOnS“ herangezogen, wobei auch hier eine tabellarische Eingabe erfolgte, womit die Variablen in den assertionalen Mustern des WADL-ODM befüllt wurden. Die Konsistenz der modellierten Ontologie wurde, wie in Abschnitt IV-3.3.1 erläutert, durch die verwendete Graph-Datenbank sichergestellt. Die Kohärenz der ABox wurde durch die Benutzerführung des Werkzeugs „LIonS“ sichergestellt, welche die in Abbildung V.8 gezeigte Sequenz einforderte und notwendige Eingaben einforderte.

Auf Basis der erstellten Heavyweight-Ontologie und den im Rahmen der Ontologie-Anforderungserhebung genannten Kompetenzfragen wurden SPARQL-Abfragen entwickelt, um die Erfüllung der Anforderungen zu überprüfen und somit die Vollständigkeit der On-

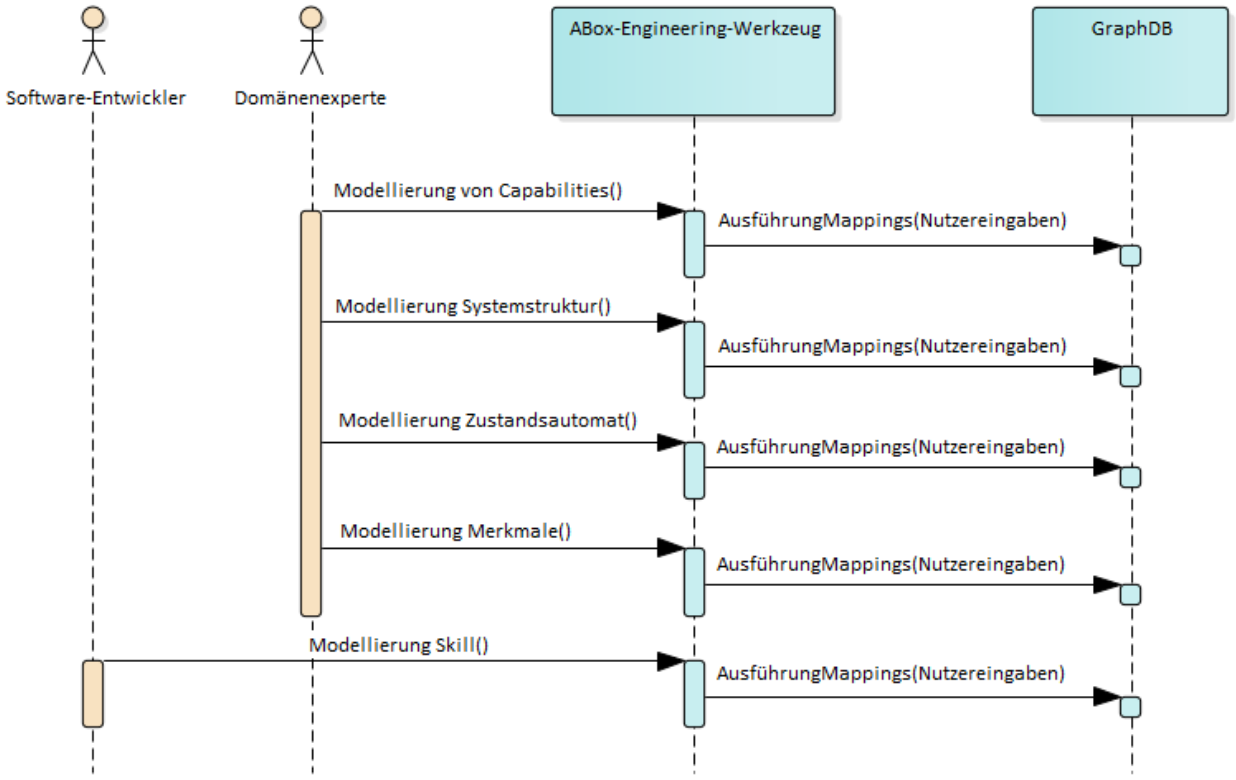

Abbildung V.8.: Ablauf der ABox-Modellierung in der Fallstudie AF

tologie zu bewerten. Eine exemplarische SPARQL-Abfrage, zur Erfüllung von Anforderung 2 aus Tabelle V.3, ist in Listing V.2 dargestellt. Demnach werden die in der Anforderungen enthaltenen Variablen `?skill`, `?command` und `?module` zunächst gebunden. Im Anschluss wird die Skillimplementierung entsprechend den in der Alignment-Ontologie definierten Begriff `Cap:RestSkill` und Relation `Cap:provideSkill` gesucht. Anschließend werden weitere Informationen über die Skillimplementierung gesucht, welche im vorliegenden Beispiel die `WADL:Method` ist. Tabelle V.4 stellt eine exemplarische Rückgabe für die SPARQL-Abfrage in Listing V.2 dar. Eine SPARQL-Abfrage, welche die Anforderung 2 aus Tabelle V.3 erfüllt, ist in Listing V.2 im Anhang C dargestellt. Die erstellte Ontologie wurde als vollständig und konsistent bewertet.

Listing V.2: SPARQL-Abfrage zur Erfüllung der Anforderung 2 aus Fallstudie AF

```
PREFIX VDI3682: <http://www.hsu-ifa.de/ontologies/VDI3682#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX WADL: <http://www.hsu-ifa.de/ontologies/WADL#>
PREFIX Cap: <http://www.hsu-ifa.de/ontologies/capability-model#>

# ID = 5: Wie kann der Skill "Bohren-1" von Modul "X" gestarted werden?

SELECT ?basePath ?path ?httpMethod WHERE {
    # Definiere betrachteten Skill, Kommando und Modul
    BIND(IRI("http://www.hsu-ifa.de/ontologies/capability-model#Bohren-1") AS ?
```

```
        skill).
    BIND(IRI("http://www.hsu-ifa.de/ontologies/capability-model#Start") AS ?
        command).
    BIND(IRI("http://www.hsu-ifa.de/ontologies/capability-model#X") AS ?module).

    # Finde die Skillimplementierung
    ?module a VDI2206:Module.
    ?module Cap:provideSkill ?skill.
    ?skill a Cap:RestSkill;
        WADL:hasBase ?basePath;
        WADL:hasResource ?resource.

    # Finde zusaetzliche Informationen ueber die Skillimplementierung
    ?resource WADL:hasPath ?path;
        WADL:hasMethod ?skillMethod.
    ?skillMethod rdf:type ?httpMethod.
    ?httpMethod sesame:directSubClassOf WADL:Method.

    # Finde die Transition, welche mit dem Kommando "Start" verbunden ist
    ?command a ISA88:Transition;
        Cap:invokedBy ?skillMethod.

}
```

Tabelle V.4.: Exemplarische Rückgabe zu Listing V.2

?basePath	?path	?httpMethod
http://192.168.2.100	http://192.168.2.1/Bohren-1/start	WADL:POST

PREFIX WADL: http://www.hsu-ifa.de/ontologies/WADL

Im Rahmen der ABox-Modellierung der Fallstudie wurden 2604 Tripel modelliert. Der Ansatz für die semi-automatisierte Erstellung der ABox wurde dabei durch die Stakeholder als äußerst positiv aufgefasst, was insbesondere darin begründet war, dass für die Nutzung des Werkzeugs lediglich Wissen über die zugrundeliegenden Informationsressourcen notwendig ist und alle Ontologie-spezifischen Aspekte durch das Software-Werkzeug abgedeckt werden. Als Optimierungspotential wurde insbesondere eine stärkere Betrachtung der Einbindung des Werkzeugs in ein Engineering-Vorgehen genannt, um das Werkzeug an die Bedürfnisse des Engineerings-Vorgehens anpassen zu können und somit eine noch höhere Effizienz in der ABox-Entwicklung zu ermöglichen.

V-2.3. Industrielle Fallstudie „Prozessanalyse“

Das industrielle Fallbeispiel, welches nachfolgend vorgestellt wird, liefert die Grundlage für die Fallstudien PA-1 und PA-2 in Tabelle V.1, welche in Kooperation mit dem Unternehmen „C“ durchgeführt wurden. Das Unternehmen „C“ ist in der Luftfahrtbranche tätig und befasst

sich mit der Entwicklung von Produktionssystemen für Kompositbauteile.

V-2.3.1. Beschreibung des Fallbeispiels

Die Fertigung von Kompositbauteilen in der Luftfahrtbranche ist ein zunehmend automatisierter Prozess, bei welchem mit Hilfe des Harzinjektionsverfahrens (englisch: „Resin Transfer Moulding“, kurz RTM) beispielsweise Teile des Tragwerks hergestellt werden. Die Herstellung von Bauteilen erfolgt dabei in der Regel in sehr kleinen Losen, wobei die für den Prozess notwendigen Sollparameter, wie zum Beispiel Drücke und Temperaturen sowie deren zeitlicher Verlauf, meist basierend auf Expertenwissen für ein Los eingestellt werden. Die zunehmende Automatisierung führt zu einer steigenden Verfügbarkeit von Daten über den Herstellungsprozess der Bauteile, welche für Analysezwecke direkt nach der Fertigung, oder nach einem Schadensfall auch lange Zeit nach der Fertigung, herangezogen werden können. Darüber hinaus können die Sollparameter die im Rahmen der Fertigung eingestellt wurden, bei Wiederkehren der Produktanfrage als Vorlage für die Fertigung weiterer Bauteile herangezogen werden.

Bei einer RTM-Anlage des Unternehmens „C“ handelt sich um ein komplexes, automatisierungstechnisches System mit einer Vielzahl an speicherprogrammierbaren Steuerungen (SPS) sowie einer Vielzahl an SQL-Datenbanken. Während die SPSen in der Regel einen OPC-UA Server aufweisen, über den aktuelle Variablen abgefragt werden können, dienen die SQL-Datenbaken der Speicherung von historischen Soll- und Istparametern, welche im Rahmen der Fertigung von Bauteilen anfallen. Des Weiteren befinden sich Auftragsdaten, wie z.B. Losgröße und Bauteilidentifikationsnummer, in einem Manufacturing Execution System, welches ebenfalls Daten in einer SQL-Datenbank speichert. Als wesentlicher Problemfall ist die Heterogenität verschiedener SQL-Schemata und OPC-UA-Informationsmodelle zu nennen, welche als einzelne "Datensilos" zu verstehen sind, die lediglich durch menschliche Interpretation eine informationstechnische Verknüpfung erfahren. Dies führt dazu, dass sich insbesondere eine nachträgliche Analyse von Soll- und Istparametern als aufwendiger Prozess darstellt, welcher stark durch die Suche nach den richtigen Daten in der richtigen Datenquelle und das Verständnis der darin enthaltenen Informationen geprägt ist.

Das Unternehmen „C“ möchte daher eine Prozessanalyse-Applikation entwickeln. Diese Applikation soll die im Rahmen der Fertigung eines Bauteils anfallenden Daten einer Maschine mit einem Auftrag und Los verknüpfen können und bei Bedarf einem Techniker anzeigen. Auf diese Weise soll der Techniker in die Lage versetzt werden, die Daten eines Prozesses auch nachträglich analysieren zu können. Hierfür soll eine Ontologie entwickelt werden, welche einerseits die verschiedenen Datenquellen miteinander in Verbindung setzt und andererseits auch die Semantik der in den Daten enthaltenen Informationen darstellt. Die Prozessanalyse-Applikation soll dann nicht über das Wissen zur Interpretation der verschiedenen Datenquellen verfügen müssen, sondern lediglich das Wissen über die Inhalte der Ontologie. Für den Zugriff auf dynamische Daten in SQL- oder OPC-UA-Datenquellen soll eine Middleware sorgen, welche Anfragen der Prozessanalyse-Applikation in Anfragen an Datenquellen übersetzt. Auf diese Weise soll die Entwicklung der Prozessanalyse-Applikation von der heterogenen Semantik der Datenquellen in der Produktion entkoppelt werden und auch eine Ergänzung von

Datenquellen ohne Anpassung der Applikation erfolgen können.

V-2.3.2. Durchführung der Fallstudien

In Bezug auf das genannte Fallbeispiel, wurden zwei Fallstudien (PA-1 und PA-2) durchgeführt. Dabei waren die Fallstudien so konzipiert, dass Erkenntnisse aus der ersten Fallstudie in die zweite Fallstudie einfließen konnten. Das Ziel beider Fallstudien war einerseits die Entwicklung einer TBox, welche die für den Anwendungsfall notwendigen Informationen abbilden kann. Darüber hinaus sollte die ABox alle quasi-statischen Informationen über ein real existierendes Produktionssystem enthalten und eine Menge von Mappings für die dynamischen Informationen (z.B. Temperaturen oder Drücke) bereitstellen. Die Entwicklung der Middleware, welche die Mappings ausführt, war nicht Bestandteil der Fallstudie. Nachfolgend sind die wesentlichen Erkenntnisse der Fallstudien PA-1 und PA-2 dargestellt. Nachfolgend wird das in den Fallstudien betrachtete Produktionssystem als „Anlage“ abgekürzt.

V-2.3.2.1. Ontologie-Anforderungserhebung

In der Fallstudie PA-1 wurde die Anforderungserhebung entsprechend der in dieser Arbeit vorgestellten Methodik durchgeführt. Dabei ergaben sich keine weiteren, wesentlichen Erkenntnisse hinsichtlich der Notwendigkeit einer Weiterentwicklung der Methode. Im Rahmen der Fallstudie PA-2 wurden die Anforderungen aus der vorhergehenden Fallstudie PA-1 aufgegriffen und überprüft, woraus sich keine Änderungen ergaben. Ein beiden Fällen wurde ein eintägiger Workshop zur Anforderungserhebung bzw. Anforderungsüberprüfung durchgeführt. Nachfolgend sind die Ergebnisse der Ontologie-Anforderungserhebung dargestellt.

Im Rahmen der Durchführung der Ontologie-Anforderungserhebung wurden zunächst diverse User Stories definiert. Folgender User Story wurde die höchste Priorität zugewiesen: „Als Prozessmanager möchte ich einen Prozessüberblick über gelaufene Prozesse erhalten sowie einen Soll-Ist-Abgleich zwischen Produktanforderungen (Soll) und Prozessdaten (Ist) vornehmen können, um sicherzustellen, dass gefertigte Produkte den gestellten Qualitätsanforderungen entsprechen.“ Als Zweck der Ontologie wurde die Spezifikation von produktionsrelevanten Produkt- Prozess- und Ressourceninformationen genannt, sowie die Integration heterogener Informationsquellen. Für die genannte User Story wurde ein Anwendungsfalldiagramm entworfen, welches in Abbildung V.9 zu finden ist. Die sogenannte „Qualitätsanalyseplattform“ hat demnach die Funktion, eine Prozessübersicht für den Prozessmanager bereitzustellen. Dabei inkludiert die Prozessübersicht die Bereitstellung aktueller Prozessdaten wie Drücke und Temperaturen („Prozess-KPIs“) sowie die Bereitstellung von Informationen über hergestellte Produkte. Letztere Funktion wird erweitert um eine Funktion die Produktanforderungen (z.B. Sollwerte für Prozesse oder Reihenfolgen von Prozessschritten) bezieht sowie eine Funktion die historische Prozessdaten (z.B. Druckzeitreihen) und eine Funktion die einen Soll-Ist-Abgleich zwischen Produktanforderungen und historischen Prozessdaten vornimmt. Aus dem Anwendungfall wurde eine Funktionsstruktur der zu entwerfenden Systemgruppe definiert, welche in Abbildung V.10 zu finden ist und nachfolgend erläutert wird.

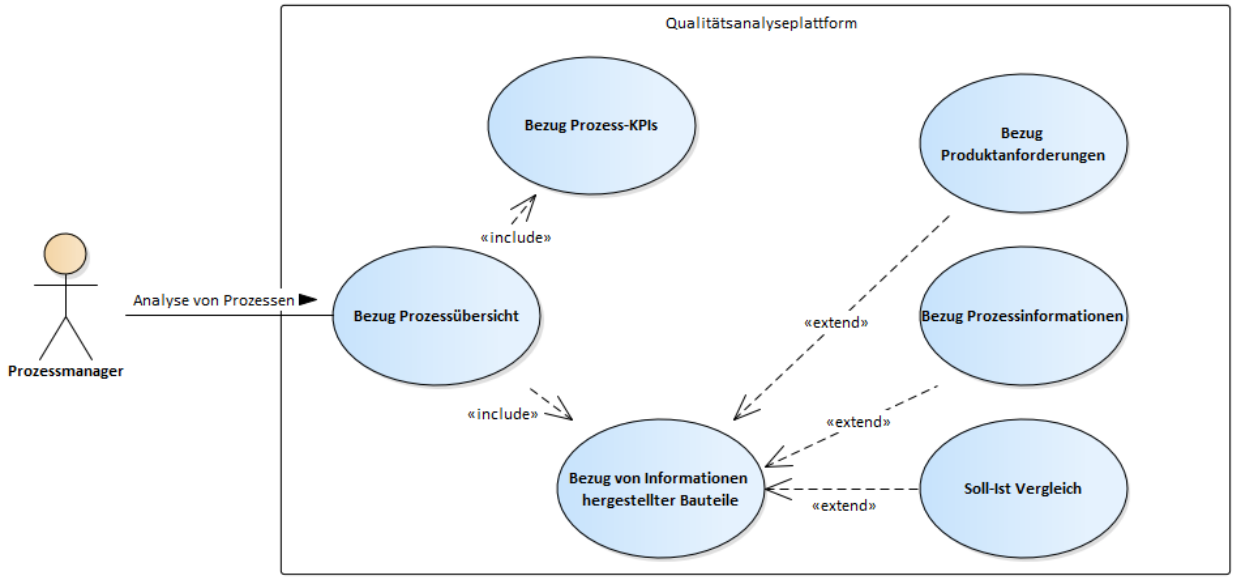

Abbildung V.9.: Modellierung des Anwendungsfalls der Fallstudie PA-1&2

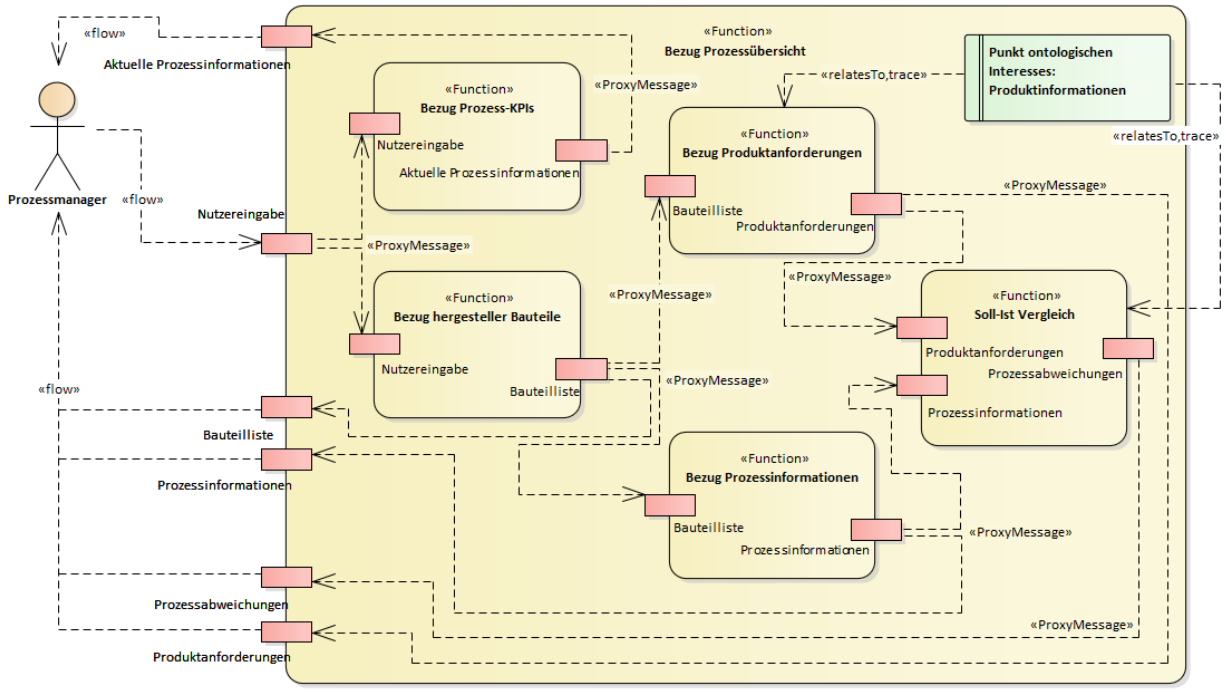

Abbildung V.10.: Modellierung der Funktionsstruktur der Systemgruppe der Fallstudie PA-1&2

Im Funktionsstrukturdiagramm wurden alle Funktionen aus der Abbildung V.9 überführt und um einen Informations- bzw. Nachrichtenfluss ergänzt. Dabei wird ersichtlich, welche Informationen für die Ausführung einer Funktion notwendig sind und welche Informationen bei der Ausführung der Funktion entstehen. Beispielsweise erfordert die Funktion „Bezug Produktanforderungen“ eine Bauteilliste als Input, wobei für alle die Bauteile der Liste dann die jeweiligen Produktanforderungen zur Herstellung der Bauteile als Output generiert werden.

Wie in Abbildung V.10 ersichtlich, wurden auch Punkte ontologischen Interesses ergänzt. Der Übersicht halber wurde lediglich ein Punkt ontologischen Interesses dargestellt und mit den Funktionen „Bezug Produktanforderungen“ und „Soll-Ist-Vergleich“ verbunden.

Insgesamt wurden drei Punkte ontologischen Interesses definiert und in einem Anforderungsdiagramm zusammengefasst. Letzteres ist in Abbildung V.11 exemplarisch dargestellt. Hier zu erkennen ist die in Abschnitt II-4.4 vorgeschlagene Modellierung. Demnach komponieren die in Abbildung V.10 dargestellten Punkte ontologischen Interesses eine Menge an funktionalen, ontologischen Anforderungen. Eine funktionale, ontologische Anforderung (z.B. „Anforderungen an Prozess“) beinhaltet nun eine Menge von UML-Operationen. Jede UML-Operation steht für eine Kompetenzfrage und Antwort, welche später in eine SPARQL-Abfrage überführt werden kann. Rechts in der Abbildung ist dargestellt, dass für die Ausführung der SPARQL-Abfrage ein „Produkt“ als Parameter übergeben werden muss und wie die Kompetenzfrage und Antwort aussehen. Auf diese Weise wurden 24 Kompetenzfragen und Antworten modelliert. Tabelle V.5 stellt die vier in Abbildung V.11 genannten funktionalen ontologischen Anforderungen dar, wobei die tabellarische Darstellung lediglich aus Platzgründen vorgenommen wird. Die in den Kompetenzfragen der V.5 angesprochenen Informationen wurden als quasi-statisch bewertet. Unter den insgesamt aufgenommenen Kompetenzfragen befanden sich jedoch auch Informationen mit einer Dynamik im Sekundenbereich.

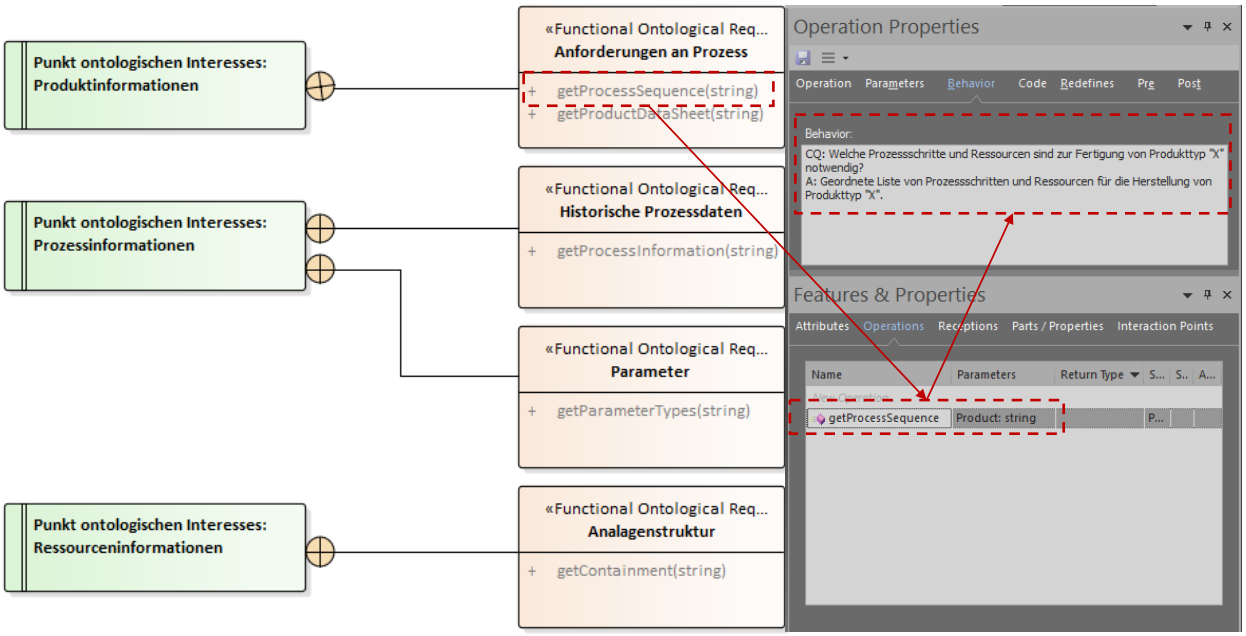

Abbildung V.11.: Modellierung der ontologischen Anforderungen aus der Fallstudie PA-1&2

Zur Darstellung der zu entwerfenden Systemgruppe wurde ein Systemgruppenstrukturdiagramm erstellt, welches in Abbildung V.12 dargestellt ist. Demnach besteht die Systemgruppe aus einer Frontend-Applikation, einem Backend sowie einer Middleware. Das Frontend bezieht die darzustellenden Daten vom Backend. Das Backend nutzt SPARQL-Abfragen um Informationen aus der Ontologie zu beziehen. Dabei sendet es die SPARQL-Abfragen an eine Middleware. Letztere leitet eingehende SPARQL-Abfragen an die Graph-Datenbank weiter und

Tabelle V.5.: Exemplarische Kompetenzfragen und Antworten

ID	Kompetenzfrage	Antwort
1	Welche Module besitzt die Anlage „X“?	Liste von Modulen und Virtual-Reality-Links.
2	Welche Prozessschritte und Ressourcen sind zur Fertigung von Produkttyp „X“ notwendig?	Geordnete Liste von Prozessschritten und Ressourcen für die Herstellung von Produkttyp „X“.
3	Welche Prozesswerte (Parameter) müssen für die Ausführung von Prozess „X“ definiert werden?	Liste mit Typen von Prozesswerten (Prozessparameter) die vor Prozessstart definiert werden müssen.
4	Welche historischen Prozesswerte (Zeitreihe) für Druck und Temperatur liegen für den ausgeführten Prozess „X“ vor?	Darstellung der Zeitreihe für Prozess „X“.

übernimmt im Falle von vorliegenden Mappings auch deren Ausführung, sodass eine Virtualisierung bestimmter Teile der ABox, bspw. für aktuelle Druckwerte eines Prozesses, erfolgen kann. Die Middleware übersetzt demnach auch eingehende SPARQL-Abfragen in Anfragen an einen OPC-UA-Server bzw. eine SQL-Datenbank. Ist in der ABox ein Verweis auf eine RESTful Schnittstelle gegeben, so wird dieser Verweis an das Backend zurückgegeben, welches diese dann auflöst um bspw. Produktdatenblätter oder Visualisierungen einer Virtual-Reality-Umgebung zu beziehen. Die betrachtete Systemgruppe wurde als geschlossen definiert.

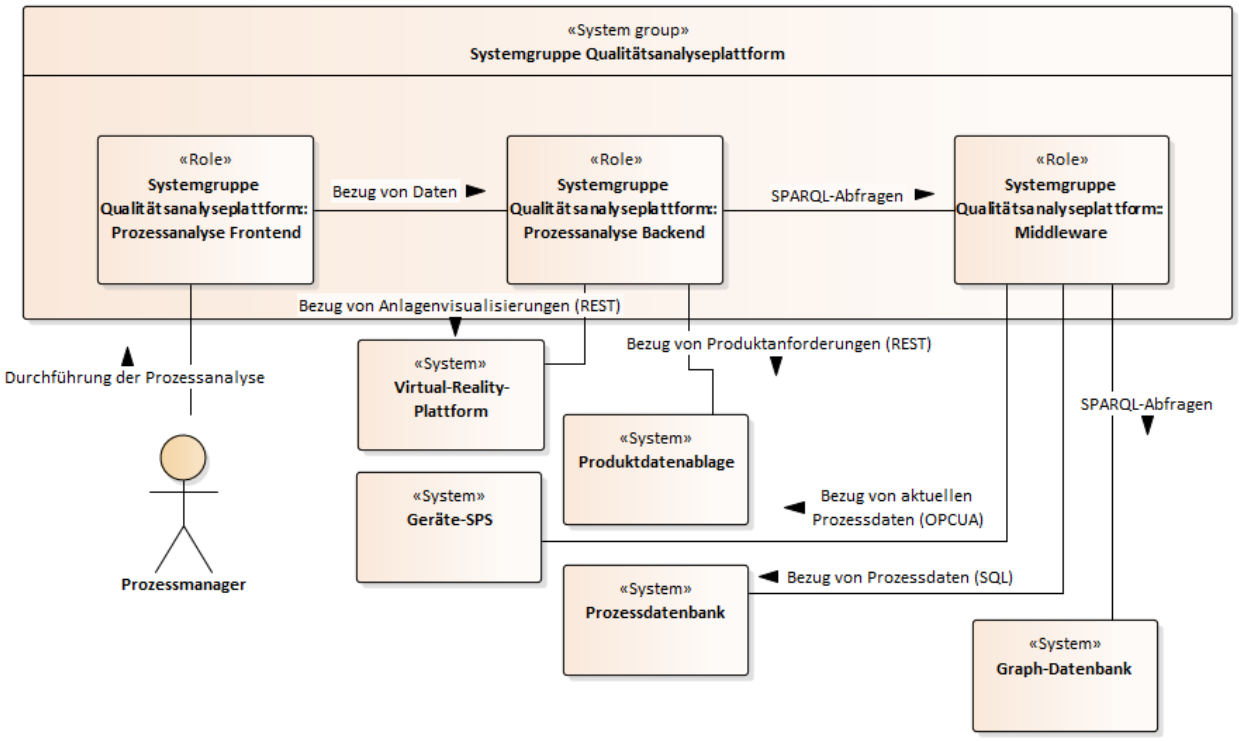

Abbildung V.12.: Modellierung der Systemgruppenstruktur in der Fallstudie PA-1&2

Darüber hinaus wurde das Funktionsverhalten der in Abbildung V.10 enthaltenen Funktionen modelliert. Dies ist in Abbildung V.13 exemplarisch für die Funktion „Bezug von Produktanforderungen“ dargestellt. Dort ist dargestellt, dass wenn der Prozessmanager die „Produktinformationsseite“ im Frontend öffnet, über das Backend zwei Anfragen gesendet

werden, welche eine der in Abbildung V.11 dargestellten funktionalen, ontologischen Anforderungen ansprechen. In diesem Falle ist es der Aufruf der für die Herstellung notwendigen Prozesssequenz sowie ein Produktdatenblatt. Zum Produktdatenblatt soll die Ontologie jedoch lediglich eine URL zurückliefern, weshalb das Backend im Anschluss das Produktdatenblatt selbst über die Produktdatenablage beziehen kann. Wie der Abbildung zu entnehmen ist, sind die dargestellten Objekte jeweils Instanzen der in Abbildung V.12 dargestellten Rollen bzw. Systeme. Das Funktionsverhalten in Abbildung V.10 beschreibt somit Anforderungen an das zu entwickelnde System bzw. die zu entwickelnde Systemgruppe und setzt diese in Beziehung zu den funktionalen ontologischen Anforderungen. Die Methodik wurde durch die beteiligten Stakeholder als zielführend erachtet und es ergaben sich keine weiteren, wesentlichen Erkenntnisse hinsichtlich der Notwendigkeit einer Weiterentwicklung.

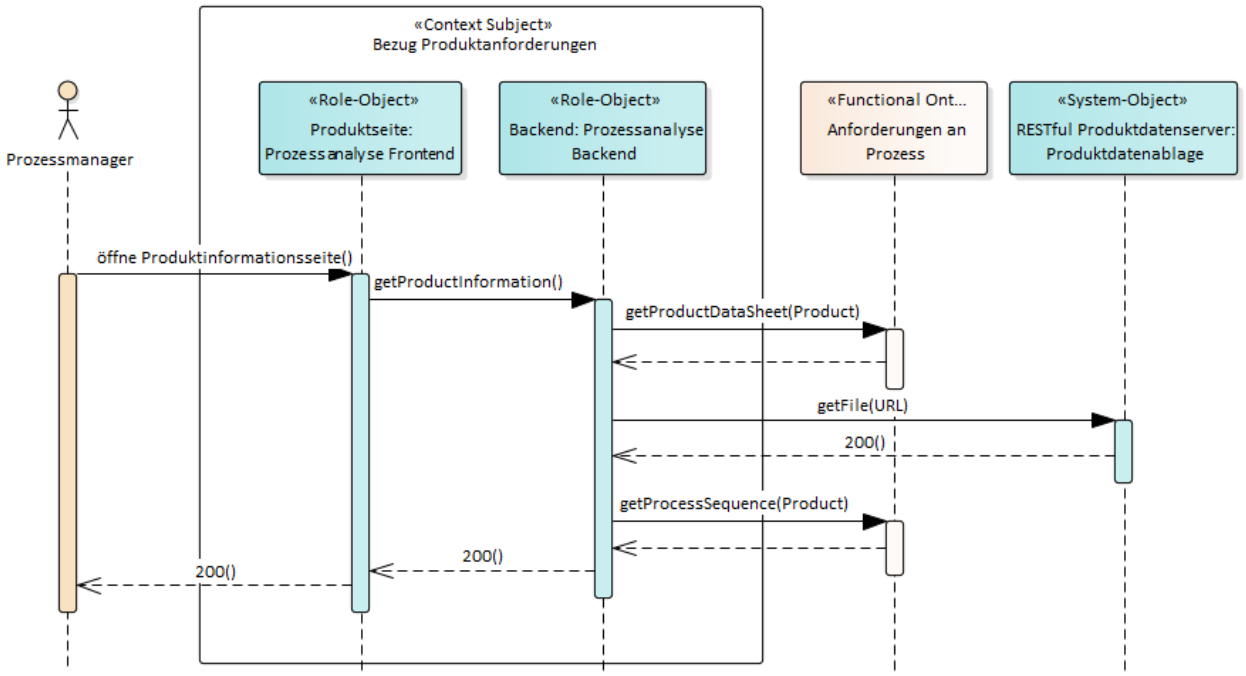

Abbildung V.13.: Modellierung des Funktionsverhaltens der Funktion „Bezug Produktanforderungen“ der Fallstudie PA-1&2

V-2.3.2.2. Ontologie-Entwicklung TBox

Bezüglich der TBox-Entwicklung wurde ebenfalls die Methode der Arbeit aufgegriffen. Es wurden die im Rahmen der Fallstudie WA-2 entwickelten DIN EN 61360 ODM, VDI 3682 ODM und VDI 2206 ODM wiederverwendet. Darüber hinaus wurde im Rahmen der Fallstudie PA-2 eine OPC-UA-TBox entsprechend [Per+19] wiederverwendet, welche bereits als Heavyweight-Ontologie vorlag und keinerlei Anpassung bedurfte. Die den ODMs zugrunde liegenden Informationsressourcen wurden durch eine Experteneinschätzung als funktional geeignet und nutzbar bewertet. Durch die Integration der OPC-UA-TBox war die Anpassung der in WA-2 entwickelten Alignment-Ontologie notwendig. Die Ergänzung der Alignment-Ontologie aus der Fallstudie WA-2 (siehe Abbildung III.18) ist in Abbildung V.14 zu finden und wird nachfolgend erläutert. Zunächst wurde ein neues ObjectProperty

`c:hasOntologicalValue` definiert, welches zwischen der Instanzbeschreibung der DIN EN 61360 `DINEN61360:InstanceDescription` und einer OPC-UA-Variable `OpcUa:UAVariable` angelegt wird. Auf diese Weise kann der Wert einer Instanzbeschreibung mit einer OPC-UA-Variable referenziert werden. Darüber hinaus kann ein `VDI2206:System` mit einem OPC-UA-Server `OpcUa:UAServer` über `c:hasOpcUaServer` in Verbindung gesetzt werden. Auf diese Weise kann angezeigt werden, welches System über einen OPC-UA-Server verfügt. Im Rahmen der Fallstudien PA-1 und PA-2 konnten somit 446 Axiome aus ODMs bisheriger Fallstudien wiederverwendet werden (DIN EN 61360 ODM, VDI 3682 ODM, VDI 2206 ODM) und 5628 Axiome aus der OPC-UA-Ontologie entsprechend [Per+19]. Es wurden durch die Anpassungen der Alignment-Ontologie lediglich 43 Axiome ergänzt.

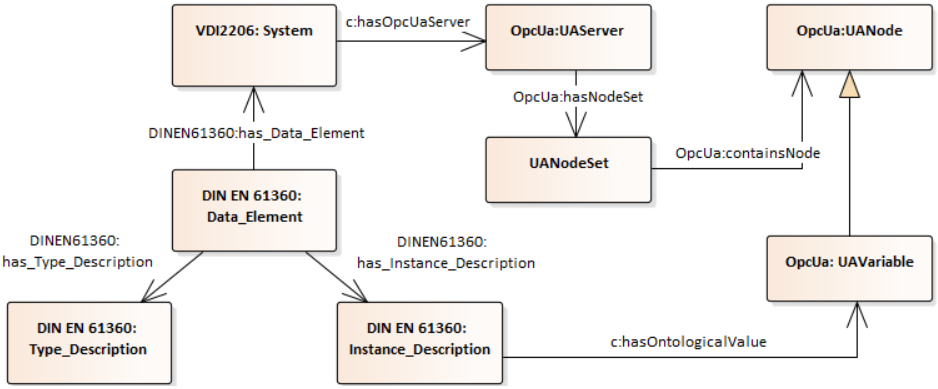

Abbildung V.14.: Vorgenommene Erweiterungen der Aligment-Ontologie aus Fallstudie WA-2 in den Fallstudien PA-1&2

V-2.3.2.3. Ontologie-Entwicklung ABox - Herleitung der Modellierungsstrategie

Im Rahmen der Fallstudie PA-1 wurde keine Herleitung der Modellierungsstrategie durchgeführt. Die verfolgte Modellierungsstrategie war, bedingt durch den Wunsch der Experten des Unternehmens, eine vollautomatische Strategie, bei der mit Hilfe von Mappings die gesamte ABox erzeugt werden sollte. Dies erwies sich insofern als problematisch, als dass bestimmte Engineering-Artefakte bei der Erstellung von Mappings einen sehr hohen Aufwand in Erstellung und Wartung erzeugten. Die daraus gewonnene Erkenntnis war, dass eine tiefer gehende Betrachtung der Engineering-Artefakte in die Strategieauswahl mit einzubeziehen ist, da ein höherer Automatisierungsgrad (wie in der Fallstudie erkannt) nicht immer zu einem geringeren Gesamtaufwand führen muss und eine Kombination von semi-automatischen und automatischen Strategien im Sinne einer hybriden Strategie als zielführend betrachtet werden kann. Dies wurde in der Fallstudie PA-2 aufgegriffen, wobei das Vorgehen nachfolgend dargestellt ist. Die im Rahmen der Fallstudie PA-2 berücksichtigten Artefakte sind in Tabelle A.4 im Anhang A beschrieben.

Die Analyse relevanter RDF-Strukturen wurde für die ODMs der VDI 3682, VDI 2206, der DIN EN 61360 und OPC-UA durchgeführt. Die zugrundeliegenden assertionalen Muster sind in Listing C.1 bis C.7 dargestellt. Basierend auf den in Abschnitt IV-3.2.1 definierten Berechnungen, kann Tabelle V.6 hergeleitet werden, welche nachfolgend erläutert wird.

Tabelle V.6.: Ergebnisse der Analyse relevanter RDF-Strukturen

ODM	Entitäten					Aufwände	
	$\|E_{var}(\lambda)\|$	v	τ_{Brutto}	η	τ_{Netto}	z_u in sec	Z_s in min
VDI 3682	48,5	100	4850	0,82	869,24	15	217,31
DIN 61360 (T)	57	16	912	0,73	245,66	15	61,41
DIN 61360 (I)	14	500	7000	0,64	2500	15	n.a.
VDI 2206	21	3000	63000	0,81	12000	15	3000
OPC UA	14	500	7000	0,214	5500	15	1375

Der Tabelle V.6 ist zu entnehmen, dass die Prozessmodellierung (VDI 3682) und die Modellierung von Typbeschreibungen von Merkmalen (DIN 61360 (T)) verhältnismäßig geringe Nettomodellierungsbedarfe (τ_{Netto}) erzeugen. Dies war in der geringen Anzahl an Varianten (v) von Prozessen (VDI 3682) sowie Typbeschreibungen von Merkmalen (DIN 61360 (T)) begründet. Die Nettomodellierungsbedarfe (τ_{Netto}), die für die Modellierung von Systemstrukturen (VDI 2206) und Instanzbeschreibungen von Merkmalen (DIN 61360 (I)) anfielen, waren jedoch wesentlich höher, was in einer höheren Anzahl von Varianten (v) begründet war. Die Anzahl an Varianten (v) von Prozessen wurde basierend auf der in der ersten Fallstudie PA-1 erstellten Prozessbeschreibungen geschätzt. Für die Anzahl an Varianten (v) der Systemstruktur wurden die Baugruppen und Komponenten im 3D-CAD-System ausgewertet. Die Anzahl der Typbeschreibungen von Merkmalen basierte auf der Beschreibung von Merkmalen durch einen Domänenexperten der Fallstudie PA-1. Die Anzahl der Instazbeschreibungen wurde mit fünf Instanzbeschreibungen pro Prozess geschätzt, was ebenfalls aus der Prozessbeschreibung der Fallstudie PA-1 entnommen wurde. Da eine große Anzahl von Instanzbeschreibungen auf die Beschreibung von OPC-UA-Variablen entfällt, wurde eine Variantenanzahl (v) von fünf pro Prozess (Summe 500) angesetzt. Basierend auf dem näherungsweise bestimmten Nettomodellierungsbedarf (τ_{Netto}) und einem Aufwand pro Nutzereingabe (z_u) von 15 Sekunden, ließ sich für eine semi-automatische Strategie ein zeitlicher Aufwand (Z_s) für jeden einzelnen ODM bestimmen. Für die Instanzbeschreibungen der DIN 61360 wurde dieser Wert nicht bestimmt, da dies lediglich durch ein automatisiertes Verfahren möglich war. Die Summe der Aufwände ergab den Gesamtaufwand von 4653,72 Minuten (ca. 78 Stunden) bei Nutzung einer semi-automatischen Modellierungsstrategie, ohne Berücksichtigung der notwendigen Werkzeugentwicklung.

Im Rahmen der Bewertung existierender Artefakte wurde zunächst die technologieorientierte Bewertung der Artefakte vorgenommen. Hierzu wurden die in Abschnitt IV-3.2.2.1 angeführten Kriterien auf die Artefakte in Tabelle A.4 angewandt. Das Ergebnis dieser Bewertung ist in Tabelle V.7 zu finden und wird nachfolgend erläutert. Erkennbar ist, dass mit Ausnahme des Artefakts „Meister Task“ alle Artefakte über eine standardisierte Form verfügten. Das Artefakt „Meister Task“ verfügte zwar über ein Schema, allerdings ist dieses auf die Darstellung von Projektmanagementaspekten (Aktivität, Start, Ende etc.) fokussiert. Die für die Fallstudie relevanten Informationen (z.B. Prozssbeschreibungen) waren somit nicht zwingend an einem spezifischen Ort im Schema zu finden, sondern variierten je nach Präferenz des Nutzers. Aus diesem Grund verfügte das Artefakt über keine standardisierte Form zu Repräsentation von Prozessbeschreibungen.

Die eindeutige Auslegbarkeit war lediglich bei Artefakten nicht gegeben, für welche anhand

Tabelle V.7.: Ergebnisse der technologieorientierten Bewertung

Name	Format	Referenz	Kriterium										
			Menschliche Lesbarkeit	Standardisierte Form	Eindeutige Auslegbarkeit	Maschinenlesbarkeit	Auswertbarkeit t.f. Werte	Identifizierbarkeit	Typisierung	Objektivität	Anpassbarkeit	Fehlerfreiheit	Summe / Maxpunktzahl
FPB.js	.json	VDI 3682	1	1	1	1	0	1	1	1	0	0	0,7
3D CAD	.stp	VDI 2206	1	1	1	0	0	1	0	1	0	1	0,6
Meister Task	.csv	VDI 3682	1	0	0	0	0	1	0	0	0	0	0,2
Anlagen-prozess-tabelle	.xls	VDI 3682	1	1	1	0	0	1	1	0	1	0	0,6
Prozess-daten	.sql	DIN 61360 (I)	1	1	0	1	0	1	0	1	0	1	0,6
UPC UA Server	.json	OPC UA	1	1	1	1	0	1	1	1	0	0	0,7
MS Visio Beschrei-bung	.xml	VDI 3682	1	1	1	1	0	1	0	1	0	1	0,7

der jeweiligen Datenobjekte nicht einwandfrei erkannt werden konnte, was sie aussagen. Dies war bspw. beim Artefakt „Meister Task“ der Fall, da bei der Eintragung eines Prozesses bspw. nicht einwandfrei erkennbar war, welche In- und Outputs er hatte. Des Weiteren galt dies für die SQL-Prozessdaten, da sie nicht alle der für eine DIN EN 61360 Instanzbeschreibung notwendigen Attribute beinhalteten (Aussagelogik, Aussageziel etc.), sondern lediglich den Wert. Hinsichtlich der Maschinenlesbarkeit wurde bewertet, inwiefern eine existierende Software-Bibliothek zur Auswertung des Artefakts genutzt werden konnte, ohne dass die Entwicklung eines eigenen Datenmodells zwingend nötig gewesen wäre. Da keine Bibliothek speziell für die Schemata der einzelnen Artefakte existierte, mit deren Hilfe auch eine Auswertbarkeit trotz fehlender Werte gewährleistet werden könnte, erhielt keines der Artefakt einen Punktwert für dieses Kriterium.

Hinsichtlich der Typisierung wurden die Artefakte dahingehend bewertet, ob eine Typisierung der für den referenzierten ODM notwendigen Typen existiert. Dies ist bspw. für „FPB.js“ der Fall, da es eine richtlinienkonforme Typisierung vorsieht. Bezüglich der Objektivität wurde unterschieden, ob die Typisierung objektiv vorgenommen wurde. Dies war bei einem Werkzeug wie „FPB.js“ der Fall, da es eine Eingabe von dem Schema widersprechenden Inhalten verbietet (bspw. Verbindung zwischen Produkten). Eine einfache Anpassbarkeit des Schemas war lediglich für die „Anlagenprozesstabelle“ gegeben, da diese das einzige Artefakt war, dass seine Typisierung direkt mitlieferte und angepasst werden konnte. Für das Kriterium der Fehlerfreiheit wurde untersucht, inwiefern die Werkzeuge u.U. dazu in der Lage waren, die korrekte Einhaltung des Schemas zu überprüfen. Dies war lediglich für „FPB.js“, sowie „Meister Task“, die „Anlagenprozesstabelle“ und die „UPC UA Server“ Spezifikation der Fall. Aus den durchgeführten Bewertungen ließ sich die individuelle technologische Eignung der Artefakte berechnen, welche in der letzten Spalte der Tabelle V.7 dargestellt ist.

Für die Bewertung der Anwendbarkeit und Skalierbarkeit möglicher Mappings wurden die Berechnungen in Abschnitt IV-3.2.2.2 auf die Artefakte in Tabelle A.4 angewandt. Die Ergebnisse sind in Tabelle V.8 zu finden und werden nachfolgend erläutert.

Tabelle V.8.: Ergebnisse der Bewertung der Anwendbarkeit und Skalierbarkeit möglicher Mappings

Name	**Format**	**Referenz**	**Existierende Verfahren**	**Skalierbarkeit**		
			Punktwert Mapping-Verfahren	Anzahl gleichzeitig existierender Schemata	Anzahl von Schema-Versionen in relevantem Zeitraum	Skalierbarkeits-faktor
FPB.js	.json	VDI 3682	0,5	1	1	1
3D CAD	.stp	VDI 2206	0,17	1	1	1
Meister Task	.csv	VDI 3682	1	1	1	1
Anlagen-prozess-tabelle	.xls	VDI 3682	1	1	3	0,33
Prozess-daten	.sql	DIN 61360 (I)	1	1	1	1
UPC UA Server	.json	OPC UA	0,5	1	1	1
MS Visio Beschrei-bung	.xml	VDI 3682	0,5	1	2	0,5

Hinsichtlich der Bewertung existierender Verfahren, wurde zunächst die Tabelle IV.4 um die jeweilige Expertise des Projektteams ergänzt. Hierzu wurden Punkte der Bewertung wie folgt hinzugefügt:

- 2: Sehr gute Kenntnisse in der Anwendung des Ansatzes
- 1: Erste bis gute Kenntnisse in der Anwendung des Ansatzes
- 0: Keine Kenntnisse in der Anwendung des Ansatzes

Die resultierende Bewertung ist in Tabelle V.9 dargestellt. Für das Format der Artefakte in Tabelle A.4 wurde dann das Verfahren gewählt, welches den höchsten Punktwert besitzt und mit Hilfe der maximal zu erreichenden Punktzahl (6) normiert. Das Ergebnis ist in Tabelle V.8 (Punktwert Mappingverfahren) dargestellt. Hinsichtlich der Anzahl gleichzeitig existierender Schemata wurde untersucht, inwiefern evtl. mehrere Schemata des gleichen Artefakts existierten, welche zu berücksichtigen waren. Dies war bei keinem Artefakt der Fall. Bezüglich der Versionierung wurde basierend auf den Erfahrungen aus der Fallstudie PA-1 angenommen, dass das Schema der „Anlagenprozesstabelle“ sich noch mindestens drei mal ändern wird, bevor eine entsprechende Reife des Artefakts angenommen werden kann. Dies war insbesondere in der Tatsache begründet, dass der Informationsbedarf der durch die „Anlagenprozesstabelle“ gedeckt werden soll, sich auch im letzten Projekt sukzessive erweiterte. Hinsichtlich des Artefaktes „MS Visio Beschreibung“ wurde festgestellt, dass das Artefakt

nicht vollständig richtlinienkonform war. Es war daher davon auszugehen, dass eine richtlinienkonforme Version erstellt werden wird. Unter Berücksichtigung dieser Annahmen ließen sich die Skalierbarkeitsfaktoren berechnen, welche in Tabelle V.8 dargestellt sind.

Tabelle V.9.: Bewertung automatisierter Mappingverfahren unter Berücksichtigung der Erfahrung des Projekt-Teams

Referenz	Format						
	CSV	XLS	SQL	XML	JSON	CAEX, AML	3D CAD
[OHM10]	4	4	-	-	-	-	-
[Blf+17]	2	2	-	-	-	-	-
[W3C12@b]*	6	6	6	-	-	-	-
[Run10]	-	-	-	-	-	0	-
[HuHe19]	-	-	-	-	-	1	-
[Dim+14]	3	-	-	3	3	-	-
[LZB17]	2	-	-	2	2	-	-
[Gla+15]	-	-	-	-	-	1	1

„*“ = Nutzbar für Virtualisierung;

Für eine modellspezifische Bewertung der Artefakte in Tabelle A.4 wurden die Formeln in Abschnitt IV-3.2.2.3 herangezogen. Die Ergebnisse sind in Tabelle V.10 dargestellt und werden nachfolgend erläutert. Die angenommene Anzahl an Varianten v, wurde aus der Analyse relevanter RDF-Strukturen übernommen, um eine Vergleichbarkeit der Berechnungen des automatisierten bzw. semi-automatisierten Verfahrens zu ermöglichen. Anschließend wurden die im Artefakt enthaltenen und für die Erstellung der ABox des referenzierten ODMs relevanten Datenobjekte $|D_{r,A_k}|$ bestimmt. Basierend darauf konnte der Anteil R am Nettomodellierungsbedarf aus Tabelle V.6 berechnet werden. Dieser Anteil drückt die Relevanz für das Projekt aus, je geringer der Anteil am Modellierungsbedarf, desto weniger relevant ist ein Artefakt für die Modellerstellung und damit das Projektziel. Hinsichtlich des Overheads wurden die für die Fallstudie relevanten Typen von Datenobjekten $|D_{r,A_k}|$ ins Verhältnis gesetzt zu den insgesamt verfügbaren Typen von Datenobjekten $|D_{A_k}|$. Dies wird als Overhead D_{o,A_k} der Datenobjekte bezeichnet.

Tabelle V.10.: Ergebnisse der modellspezifischen Bewertung

Name	Format	Referenz	Relevanz			Overhead	
			v	$\|D_{r,A_k}\|$	R	$\|D_{A_k}\|$	D_{o,A_k}
FPB.js	.json	VDI 3682	100	14	0,07	14	0,00
3D-CAD	.stp	VDI 2206	3000	3	0,43	10	0,70
Meister Task	.csv	VDI 3682	100	5	0,02	14	0,64
Anlagenprozess-tabelle	.xls	VDI 3682	100	16	0,08	34	0,53
Prozessdaten	.sql	DIN 61360 (I)	500	1	0,02	1	0,00
UPC UA Server	.json	OPC UA	500	9	0,21	20	0,45
MS Visio Beschreibung	.xml	VDI 3682	100	14	0,07	56	0,75

Basierend auf den bisher ermittelten Kennzahlen, ließ sich eine zusammenfassende Bewertung der Artefakte entsprechend der Formeln in Abschnitt IV-3.2.2.4 vornehmen. Die Ergeb-

nisse dieser Bewertung sind in Tabelle V.11 zu finden und werden nachfolgend erläutert. Der Bewertungsfaktor γ, welcher im Optimalfall „1“ ist, ist geeignet um einen Vergleich zwischen Artefakten anzustreben, die die gleichen Inhalte beschreiben und somit als Alternativen zu betrachten sind. Dieser Fall ist lediglich für die Prozesbeschreibungen entsprechend VDI 3682 zutreffend. Demnach ist das Artefakt „FPB.js“ das am besten geeignete Artefakt, welches nachfolgend weiter betrachtet wurde. Darüber hinaus wurde an dieser Stelle ein Aufwand für die Erstellung des Mappings eines einzelnen Datenpunktes in Höhe von 10 Minuten für alle Formate angenommen, um einen Aufwand für die Erstellung des Mappings abzuschätzen. Die verbliebenen Artefakte wurden nachfolgend für die Strategieauswahl herangezogen.

Tabelle V.11.: Zusammenfassende Bewertung der Artefakte

Name	Format	Referenz	Aufwände		
			γ	z_{a_k}	Z_M
FPB.js	.json	VDI 3682	1,36	600,00	190,91
3D-CAD	.stp	VDI 2206	2,81	600,00	281,25
Meister Task	.csv	VDI 3682	1,93	600,00	269,72
Anlagenprozesstabelle	.xls	VDI 3682	4,35	600,00	1477,84
Prozessdaten	.sql	DIN 61360 (I)	1,15	600,00	11,54
UPC UA Server	.json	OPC UA	1,71	600,00	342,86
MS Visio Beschreibung	.xml	VDI 3682	4,14	600,00	2317,24

Basierend auf den Ergebnissen konnte eine Auswahl der Modellierungsstrategie durch Vergleich der zu erwartenden Aufwände erfolgen, welcher in Tabelle V.12 dargestellt ist. Dabei sei an dieser Stelle darauf hingewiesen, dass es sich bei der Schätzung der zu erwartenden Aufwände keineswegs um exakte Zahlen handelt. Vielmehr diente die Bestimmung der Aufwände zwischen automatisierter und semi-automatisierter Verfahrensweise lediglich der Aufdeckung großer Aufwandsunterschiede.

Tabelle V.12.: Vergleich zu erwartender Aufwände

ODM	Artefakt	Aufwände in Minuten	
		semi-automatisch	automatisch
VDI 3682	FPB.js	217,31	190,91
DIN 61360 (T)	-	61,41	-
DIN 61360 (I)	Prozessdaten	-	11,54
VDI 2206	3D-CAD	3000,00	281,25
OPC UA	UPC UA Server	1375,00	342,86

Betrachtet man den Modellbereich des VDI 3682 ODM, so ist der Aufwand im automatischen Fall nur unwesentlich geringer als im semi-automatischen Fall. Dennoch konnte klar für den automatischen Fall entschieden werden, da die Aufwände für die Werkzeug-Entwicklung noch unberücksichtigt sind und die erstellten Mappings potentiell auch in anderen Projekten eingesetzt werden konnten. Für die Erstellung der Typbeschreibungen der des DIN 61360 ODM hat sich herausgestellt, dass keines der Artefakte eine derartige Beschreibung beinhaltet, weshalb diese semi-automatisch erstellt werden musste. Für die Instanzbeschreibungen des DIN 61360 ODM hingegen trifft das Gegenteil zu: Sie mussten automatisiert erstellt werden. Im Modellbereich des VDI 2206 ODM hingegen war ein erheblicher Aufwandsunterschied

erkennbar, der klar gegen eine semi-automatische Strategie sprach. Dies war ebenso für das OPC-UA-ODM der Fall. Die resultierende Modellierunsstrategie war somit eine hybride, da sowohl semi-automatische als auch automatische Bestandteile enthalten waren.

V-2.3.2.4. Ontologie-Entwicklung ABox - ABox-Erstellung

Der für die Fallstudie PA-2 entwickelte Workflow ist in Abbildung V.15 zu finden und wird nachfolgend erläutert. Vor der Durchführung wurden Mappings für die automatisch zu erstellenden ABox-Bestandteile (3D-CAD, Prozessdaten, OPC-UA-Server und FPB.js) erstellt. Darüber hinaus wurde das Werkzeug „LIOnS“ dahingehend erweitert, dass es auch eine Benutzerschnittstelle zum Hochladen der Artefakte anbietet und anschließend die Ausführung von Mappings zum Zwecke der Materialisierung der ABox übernimmt. Im ersten Schritt der in Abbildung V.15 dargestellten Sequenz wurde die Formalisierte Prozessbeschreibung der Anlage durch den Maschinenbau als führendes Gewerk erstellt. Dies entspricht dem Vorschlag aus [Sch18], wobei das verwendete Werkzeug hier FPB.js entsprechend [NHF20*] ist.

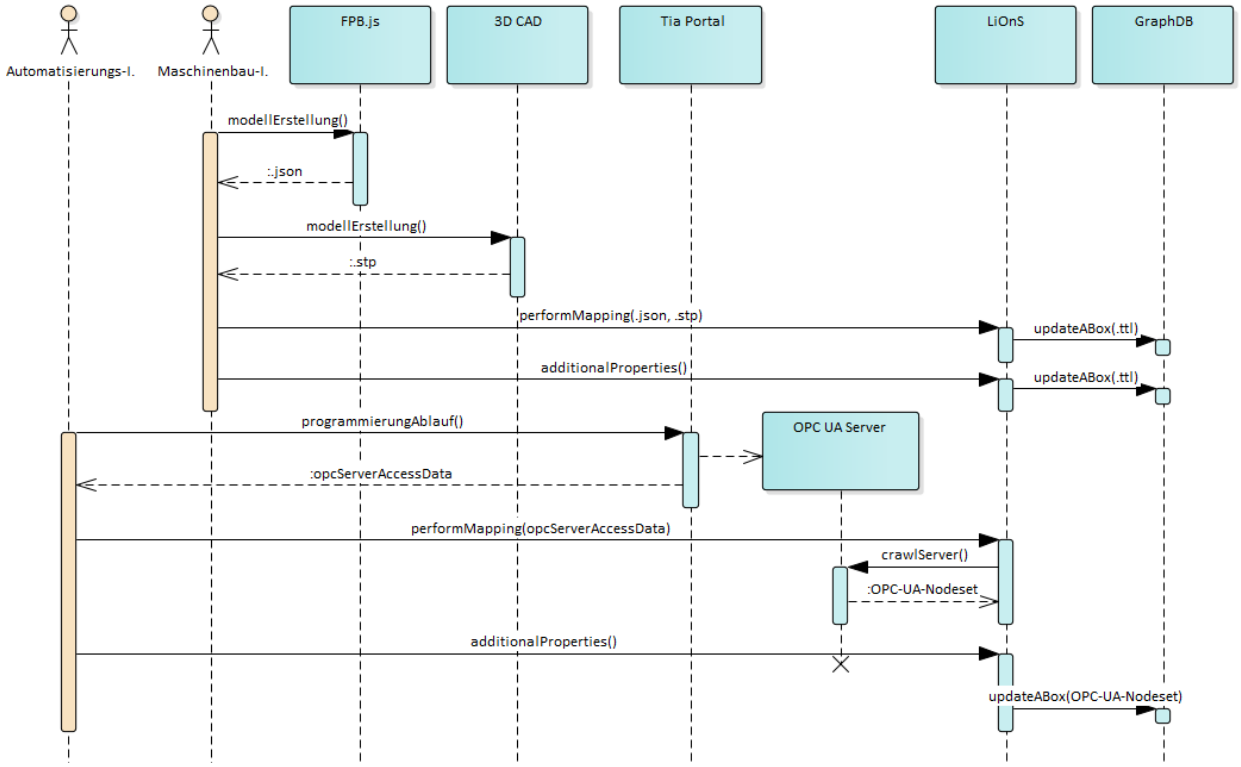

Abbildung V.15.: Workflow zur Erstellung der ABox in der Fallstudie PA-2

Anschließend begann der Maschinenbau mit der Erstellung der Systemstruktur und 3D-Modellierung in einem entsprechenden Werkzeug, wobei ein .stp (STEP) Export erfolgt. Das Artefakt von FPB.js und dem 3D-CAD-Werkzeug werden im Werkzeug LIOnS hochgeladen. Dort erfolgt ein Mapping, wobei die Ergebnisse der Mappings der Graph-Datenbank und somit der ABox des Projektes hinzugefügt werden. Anschließend nimmt der Maschinenbau einige zusätzliche Eingaben vor. Dies umfasst einerseits die Definition von Typbeschreibungen des DIN 61360 ODM im Werkzeug „LIOnS“, als auch ergänzende Modellverknüpfungen

zwischen dem Artefakt von FPB.js und dem 3D-CAD-Artefakt (siehe Abschnitt IV-3.4.2). Anschließend kann das automatisierungstechnische Gewerk mit der Erstellung von Programmabläufen der einzelnen Anlagenteile beginnen. Im Werkzeug LIOnS werden dann die erstellten OPC-UA-Server automatisch erkundet („crawling“) und ein Mapping des OPC-Nodesets wird durchgeführt und in die ABox überführt. Darüber hinaus können zusätzliche Typisierungen der OPC-UA-Variablen vorgenommen werden, um bspw. bestimmte Messwerte mit Hilfe von DIN EN 61360 Beschreibungen zu kennzeichnen. Nicht zuletzt wird eine Verknüpfung zwischen dem OPC-UA-Server und einer in der Strukturbeschreibung (3D-CAD) aufgeführten Baugruppe bzw. Komponente (im Regelfall eine Steuerung) hergestellt. Die Konsistenz der Ontologie wurde durch die eingesetzte Graph-Datenbank sichergestellt, welche vor der Ausführung jeder SPARQL-Update-Operation überprüfte, ob diese die Konsistenz gefährdet. Die Kohärenz wurde der ABox wurde durch die Benutzerführung des Werkzeug „LIOnS“ sichergestellt, welche die in Abbildung V.15 dargestellte Sequenz und notwendige Eingaben einforderte.

Auf diese Weise wurden 21324 Tripel quasi-statischer Natur und 407913 dynamischer Natur (unter der Annahme der gleichzeitigen Ausführung aller dynamischen Mappings und der minimalen Anzahl beteiligter Systeme) erzeugt. Die erstellte Ontologie wurde als vollständig und konsistent bewertet. Listing V.3 stellt eine exemplarische SPARQL-Abfrage dar, welche für die Erfüllung von Anforderung 2 herangezogen wurde. Hierbei wird zunächst der betrachtete Prozess als Variable gebunden. Da Prozesswerte im Rahmen der Fallstudie mit Hilfe des DIN EN 61360 ODM modelliert wurden, werden die gesuchten Prozessdaten über die jeweilige Typbeschreibung identifiziert. Es werden dann die Soll- und Istwerte gesucht und dargestellt. Tabelle V.13 stellt eine exemplarische Rückgabe dieser Anfrage dar.

Listing V.3: SPARQL-Abfrage zur Erfüllung der Anforderung 4 aus Fallstudie PA-1&2

```
PREFIX VDI3682: <http://www.hsu-ifa.de/ontologies/VDI3682#>
PREFIX DE6: <http://www.hsu-ifa.de/ontologies/DINEN61360#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX c: <http://C_Knowledge_Base#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

# ID = 4: Welche historischen Prozesswerte fuer Druck und Temperatur liegen fuer
    den ausgefuehrten Prozess "ON456789_O_2-21" vor?

SELECT ?Index
?NominalTempValue ?ActualTempValue
?NominalPressValue ?ActualPressValue

WHERE {
    # Definiere den betrachteten Prozess
    BIND(IRI("http://C_Knowledge_Base#ON456789_O_2-21") AS ?Process)

    # Suche die Prozessdaten fuer den Prozess
    ?Process DE6:has_Data_Element ?DEI.
    ?DEI DE6:Instance_Description_has_Type c:HistoricalProcessData.
```

```
    ?DEI DE6:complex_Data_Type_has_Member ?NominalTemp.
    ?DEI DE6:complex_Data_Type_has_Member ?ActualTemp.

    ?DEI DE6:complex_Data_Type_has_Member ?NominalPressure.
    ?DEI DE6:complex_Data_Type_has_Member ?ActualPressure.

    # Suche die Sollwerte (Requirement) und Ist-Werte (Actual_Value)
    ?NominalTemp DE6:Instance_Description_has_Type c:E_2-226_TD;
        DE6:Expression_Goal "Requirement";
          DE6:Array_Member_Position ?Index;
          DE6:Value ?NominalTempValue.
    ?ActualTemp DE6:Instance_Description_has_Type c:E_2-226_TD;
          DE6:Expression_Goal "Actual_Value";
          DE6:Array_Member_Position ?Index;
          DE6:Value ?ActualTempValue.


    ?NominalPressure DE6:Instance_Description_has_Type c:E_2-229_TD;
          DE6:Expression_Goal "Requirement";
          DE6:Array_Member_Position ?Index;
          DE6:Value ?NominalPressValue.
    ?ActualPressure DE6:Instance_Description_has_Type c:E_2-229_TD;
          DE6:Expression_Goal "Actual_Value";
          DE6:Array_Member_Position ?Index;
          DE6:Value ?ActualPressValue.



} ORDER BY ( xsd:long ( ( STR ( ?Index ) ) ))
```

Tabelle V.13.: Exemplarische Rückgabe zu Listing V.3

?Index	?Nominal-TempValue	?Actual-TempValue	?NominalPressure Value	?ActualPressure-Value
1	0	27	20	0
2	0	27	20	0
3	20	27	20	2
4	30	30	20	4
5	40	35	20	8
...	...	...	...	...
2718	0	27	0	0

Die SPARQL-Abfragen, welche die anderen in Tabelle V.5 genannten ontologischen Anforderungen erfüllen, sind in Listing C.11 bis C.13 dargestellt. Die Methodik wurde durch die beteiligten Stakeholder als zielführend erachtet und es ergaben sich keine weiteren, wesentlichen Erkenntnisse hinsichtlich der Notwendigkeit einer Weiterentwicklung.

V-3. Übergreifende Bewertung der Ergebnisse

Der vorliegende Abschnitt widmet sich der Überprüfung der Erreichung des in Abschnitt I-1.3 definierten Forschungsziels. Hierzu wird der Abschnitt V-3.1 zunächst auf die Erfüllung der Forschungsteilziele eingehen, indem die Anforderungserfüllung fallstudienübergreifend diskutiert wird. Anschließend diskutiert Abschnitt V-3.2 die Erreichung des Forschungsziels.

V-3.1. Forschungsteilzielbezogene Bewertung

Der vorliegende Abschnitt untersucht die Erreichung der einzelnen Forschungsteilziele in den Abschnitten V-3.1.1 bis V-3.1.3 anhand einer Diskussion der Anforderungserfüllung der einzelnen Forschungsteilziele.

V-3.1.1. Forschungsteilziel I

Das in Definition I.2 definierte Forschungsziel bezieht sich auf die Entwicklung einer Methodik zur Anforderungserhebung, wobei der Abschnitt I-4.2 die für die Zielerreichung notwendigerweise zu erfüllenden Anforderungen darstellt. Letztere sind unter Berücksichtigung der Erläuterungen des Abschnitts V-2 den Ergebnissen der Fallstudien in Tabelle V.14 gegenübergestellt. Wie der Tabelle V.14 zu entnehmen ist, wurde die Methodik zur Ontologie-Anforderungserhebung in den Fallstudien WA-1 und AF nicht angewandt. In der ersten Fallstudie WA-2, in welcher die Methodik zur Ontologie-Anforderungserhebung angewandt wurde, ist nur eine teilweise Zielerreichung ersichtlich. Die Verknüpfung der Entwicklungsziele von Ontologie und Applikation (A1.FA1) erfolgte über die Aufnahme von User Stories und auch eine Erweiterung von Artefakten der Systementwicklung (A1.FA2) in Form von UML- bzw. SysML-Modellen fand, wie im Rahmen der Arbeit vorgeschlagen, statt und erzielte den entsprechend der Anforderungen gewünschten Effekt. Hinsichtlich der Modellierung der Funktionsstruktur wurde der im Rahmen der Arbeit vorgeschlagene Ansatz verfolgt, jedoch erfolgte keine Modellierung des Funktionsverhaltens wie im Rahmen der Arbeit vorgeschlagen, weswegen die Anforderung A1.FA3 als nur teilweise erfüllt betrachtet wurde. Das Resultat dessen war, dass die ontologischen Anforderungen nur auf Ebene der Funktionsstrukturdiagramme integriert werden konnten. Die Dokumentation der funktionalen, ontologischen Anforderungen wurde mit Hilfe von SysML-Anforderungsdiagrammen durchgeführt. Daraus ergibt sich zwar grundsätzlich die Möglichkeit einer Nachverfolgbarkeit (A1.FA4), allerdings nur auf Ebene der Punkte ontologischen Interesses und nicht auf Ebene der funktionalen ontologischen Anforderungen. Die Anforderung A1.FA4 galt daher nur als teilweise erfüllt. Die Anforderung zur Nutzung eines leichtgewichtigen Ansatzes (A1.NFA1) war in der Fallstudie WA-2 erfüllt, da einerseits UML-Modelle genutzt wurden und anderseits der Umfang der Anforderungserhebung auf Workshops begrenzt werden konnte. Eine Abstraktion von realen Systemen (A1.NFA2) gelang durch die Nutzung der UML-Modelle, welche Anforderungen an das zu entwickelnde System darstellen und die ontologischen Anforderungen lediglich in diese Darstellungen integrieren. Die Entwicklung der Applikation des Fallbeispiels begann kurz vor Ende der Fallstudie WA-2. Die Methodik wurde nach der Fallstudie WA-2 überarbeitet

und wurde als nächstes in der Fallstudie PA-1 angewandt. In der Fallstudie PA-1 wurde die Methodik zur Ontologie-Anforderungserhebung so angewandt, wie im Rahmen der Arbeit vorgeschlagen. Durch die vorgenommenen Änderungen erfolgte eine effiziente Anforderungserhebung, wobei eine vollständige Nachverfolgbarkeit von ontologischen Anforderungen erzielt wurde. Dies wurde durch die Integration der funktionalen, ontologischen Anforderungen in das Funktionsverhaltensdiagramm erzielt. Wie Tabelle V.14 weiter entnommen werden kann, führten weitere Anwendungen zu keiner veränderten Bewertung der Anforderungserfüllung. Basierend auf den durch die Fallstudien nachweislich erfüllten Anforderungen, ist das Forschungsteilziel I als erfüllt anzusehen.

Tabelle V.14.: Evaluation Forschungsteilziel I

Anforderungen	**Fallstudien**				
	WA - 1	WA - 2	PA - 1	AF	PA - 2
Verknüpfung der Entwicklungsziele von Ontologie und Applikation (A1.FA1)	n.a.	X	X	n.a.	X
Nutzung oder Erweiterung von Artefakten der Systementwicklung (A1.FA2)	n.a.	X	X	n.a.	X
Modellierung von Funktionsstrukturen und Funktionsverhalten (A1.FA3)	n.a.	(X)	X	n.a.	X
Nachverfolgbarkeit ontologischer Anforderungen (A1.FA4)	n.a.	(X)	X	n.a.	X
Leichtgewichtiger Ansatz, Nutzung einer domänenspezifischen Sprache (A1.NFA1)	n.a.	X	X	n.a.	X
Abstraktion von realen Systemen (A1.NFA2)	n.a.	X	X	n.a.	X

„X“ - erfüllt; „(X)“ - teilweise erfüllt;
„-“ - nicht erfüllt; „n.a“ - nicht angewendet

V-3.1.2. Forschungsteilziel II

Das in Definition I.3 definierte Forschungsziel bezieht sich auf die Entwicklung einer Methodik zur TBox-Entwicklung, wobei der Abschnitt I-4.3 die für die Zielerreichung notwendigerweise zu erfüllenden Anforderungen darstellt. Letztere sind unter Berücksichtigung der Erläuterungen des Abschnitts V-2 den Ergebnissen der Fallstudien in Tabelle V.15 gegenübergestellt. Wie der Tabelle V.15 zu entnehmen ist, erfüllte die in der ersten Fallstudie (WA-1) angewandte Variante der Methodik zur TBox-Entwicklung die an sie gestellten Anforderungen nur teilweise. Nennenswert ist insbesondere die entgegen der Anforderung A2.FA1 stehende, nur teilweise verfolgte Modularisierung der TBox in der Fallstudie WA-1, was bereits in der folgenden Fallstudie WA-2 zum gleichen Fallbeispiel zu vermeidbaren Aufwänden führte. Wie in der Fallstudie WA-1 ersichtlich, wurden die Ontologie-Design-Muster vornehmlich mit dem Ziel erstellt, die Anforderungen des aktuellen Projektes zu erfüllen und nicht mit dem Ziel, die zugrunde liegende Informationsressource bestmöglich als Ontologie abzubilden. Die „Module“ waren somit zu stark auf den Anwendungsfall bezogen und ihre Wie-

derverwendbarkeit war dadurch eingeschränkt. Darüber hinaus wurde im Rahmen der ersten Fallstudie die Erstellung von assertionalen Mustern (AF2.FA2) nicht berücksichtigt. In der folgenden Fallstudie (WA-2) erfolgte eine Berücksichtigung der assertionalen Muster, was sich positiv auf die Erstellung der ABox auswirkte. Allerdings berücksichtigten die Fallstudie WA-1 die Dokumentation der Ontologie-Design-Muster (A2.FA3) nur unzureichend. Die Methodik wurde nach der Fallstudie WA-1 entsprechend der gewonnen Erkenntnisse überarbeitet und als nächstes in der Fallstudie WA-2 angewandt. Im Rahmen aller Fallstudien erfolgte eine explizite Berücksichtigung von Informationsressourcen der Automatisierungstechnik (A2.FA4). Dies erfolgte einerseits durch die Suche nach Informationsressourcen in Form von Industriestandards und andererseits durch die Suche nach existierenden Ontologien aus wissenschaftlichen Veröffentlichungen die als Ontologie-Design-Muster aufgefasst und im Rahmen der Methodik verwendet werden können. Die Anforderung zur Domänenexperten-zentrischen Entwicklung (A2.NFA1) kann in allen Fallstudien als erfüllt angesehen werden. Dies ist darin begründet, dass einerseits das Wissen der Domänenexperten in den Vordergrund gestellt wird und andererseits Domänenexperten als aktiv an der Modellierung und der Verifikation der Modellierungsergebnisse teilgenommen haben. Die Anforderung zur Nutzung von Beschreibungsmitteln, welche für alle Stakeholder verständlich sind (A2.NFA2), ist ebenfalls als erfüllt anzusehen. Die genutzten UML-Klassendiagramme waren in allen Fallstudien ein angemessener Kompromiss aus allgemeiner Verständlichkeit und Ausdrucksstärke des Modells. Wie Tabelle V.15 weiter entnommen werden kann, führten nach der Fallstudie WA-2 weitere Anwendungen zu keiner veränderten Bewertung der Anforderungserfüllung. Basierend auf den durch die Fallstudien nachweislich erfüllten Anforderungen, ist das Forschungsteilziel II als erfüllt anzusehen.

Tabelle V.15.: Evaluation Forschungsteilziel II

Anforderungen	**Fallstudien**				
	WA - 1	WA - 2	PA - 1	AF	PA - 2
Modularisierung erstellter TBoxen (A2.FA1)	(X)	X	X	X	X
Modularisierung erstellter ABoxen (A2.FA2)	-	X	X	X	X
Dokumentation (A2.FA3)	(X)	X	X	X	X
Nutzung von Informationsressourcen der Automatisierungstechnik (A2.FA4)	X	X	X	X	X
Domänenexperten-zentrische Entwicklung (A2.NFA1)	X	X	X	X	X
Nutzung von Beschreibungsmitteln, welche für alle Stakeholder verständlich sind (A2.NFA2)	X	X	X	X	X

„X“ - erfüllt; „(X)“ - teilweise erfüllt;
„-“ - nicht erfüllt; „n.a“ - nicht angewendet

V-3.1.3. Forschungsteilziel III

Das in Definition I.4 definierte Forschungsziel bezieht sich auf die Entwicklung einer Methodik zur ABox-Entwicklung, wobei der Abschnitt I-4.4 die für die Zielerreichung notwendigerweise zu erfüllenden Anforderungen darstellt. Letztere sind unter Berücksichtigung der Erläuterungen des Abschnitts V-2 den Ergebnissen der Fallstudien in Tabelle V.16 gegenübergestellt. Wie der Tabelle V.16 zu entnehmen ist, fand die Methodik zur ABox-Entwicklung erstmals in der Fallstudie WA-2 Anwendung. Hier jedoch in einer initialen Version, welche keine der aufgeführten Anforderungen voll erfüllte und erhebliches Optimierungspotential besaß, wie Abschnitt V-2.1.2.3 darstellte. Die Formalisierung von Daten, Informationen und Wissen aus Planung und Betrieb (A3.FA1) erfolgte bereits in Fallstudie WA-2, wobei als Informationen aus der Planung insbesondere die Zustandsmaschine und die Maschinenstruktur hervorzuheben sind und für die Informationen des Betriebs die aktuellen Zustände der betrachteten Systeme. Die Einbindung automatisierter Ansätze (A3.FA2) erfolgte in der Fallstudie WA-2 nur teilweise. Es wurden zwar die Zustände der betrachteten Systeme virtualisiert, jedoch fand die Auswahl der Ansätze nicht strukturiert statt. Dies war erst in Fallstudie PA-2 möglich, da erst dort eine ausreichende Anzahl an Artefakten zur Verfügung stand, um die entwickelte Methodik zur Strategieauswahl sinnvoll anwenden zu können. Zwar standen die Artefakte bereits in Fallstudie PA-1 zur Verfügung, jedoch wurde hier eine vollständig automatisierte ABox-Erstellung durchgeführt, ohne die betrachteten Artefakte hinsichtlich ihrer Eignung zu analysieren. Im Rahmen der Fallstudie WA-2 wurde das erste Mal eine semi-automatisierte ABox-Erstellung (A3.FA3) auf Basis von assertionalen Mustern angewandt. Jedoch war diese Anwendung ohne eine werkzeugtechnische Unterstützung und daher nur eingeschränkt nutzbar. Das im Rahmen der Arbeit vorgestellte Werkzeug „LIOnS“ wurde im Rahmen der Fallstudie AF zum ersten Mal erfolgreich angewandt, da dort keinerlei Artefakte für die ABox-Erstellung zur Verfügung standen. Die Nutzung des Werkzeugs „LIOnS“ führte zur Erfüllung von Anforderung A3.FA3. Durch die Visualisierung der ABox mit Hilfe der werkzeugtechnischen Unterstützung („LIOnS“) in der Fallstudie AF war darüber hinaus das Verständnis (A3.NFA2) über die modellierte ABox sichergestellt, da das Werkzeug stets den aktuellen Modellierungszustand darstellt. Des Weiteren wurde durch den Ansatz der Nutzung einer Graph-Datenbank im Rahmen der ABox-Modellierung in Fallstudie WA-2 eine Serialisierung in unterschiedlichen Formaten (A3.FA4) realisiert, da eine Graph-Datenbank einen Export erstellter ABoxen in diverse Formate erlaubt. Unter anderem aus diesem Grund berücksichtigt auch das Werkzeug „LIOnS“ in seiner Architektur die Nutzung einer Graph-Datenbank. Eine Integration in das Engineering von fertigungstechnischen Produktionssystemen (A3.FA5) erfolgte erst im Rahmen der Fallstudie PA-2. Hierbei wurde der Engineering-Workflow, wie im Rahmen der Arbeit angeführt, analysiert und die ABox-Modellierung darin integriert, wodurch ein zeitlicher Bezug zu relevanten Inhalten der ABox hergestellt wurde. Durch die erstellte Sequenz ließ sich die Kohärenz (A3.NFA1 für Kohärenz) der ABox mit Hilfe des Werkzeugs „LIOnS“ sicherstellen, da die zu verknüpfenden Fakten der ABox (z.B. Maschinenstruktur und Prozessbeschreibung) nun sequentiell, entsprechend des Engineering-Workflows, erstellt und verknüpft werden konnten. Die Anforderung war in den vorangegangenen Fallstudien (WA-2, PA-1 und AF) stets teilweise erfüllt, da die Konsistenz (A3.NFA1 für Konsistenz) seit Fallstudie WA-2 durch eine Graph-Datenbank sichergestellt wurde, welche die Konsistenz der ABox automatisiert prüft. Die strukturierte Modellierung struktureller, funktionaler und verhaltensori-

entierter Fakten des betrachteten Produktionssystems (A3.FA6) wurde zwar bereits in der Fallstudie WA-2 betrachtet, jedoch erfolgte dort, wie auch in den Fallstudien PA-1 und AF, keine Berücksichtigung des Engineering-Workflows. Eine strukturierte Betrachtung struktureller, funktionaler und verhaltensorientierter Fakten erfolgte demnach erst in Fallstudie PA-2 durch Analyse des Engineering-Workflows und der Integration der ABox-Modellierung in diesen Workflow. Wie Tabelle V.16 zu entnehmen ist, erfüllte die Methodik zur ABox-Entwicklung in Fallstudie PA-2 letztlich alle Anforderungen vollständig. Basierend auf den durch die Fallstudien nachweislich erfüllten Anforderungen, ist das Forschungsteilziel III als erfüllt anzusehen.

Tabelle V.16.: Evaluation Forschungsteilziel III

Anforderungen	**Fallstudien**				
	WA - 1	WA - 2	PA - 1	AF	PA - 2
Formalisierung von Daten, Informationen und Wissen aus Planung und Betrieb (A3.FA1)	n.a	X	X	X	X
Einbindung automatisierter Ansätze (A3.FA2)	n.a	(X)	(X)	n.a.	X
Semi-automatisierte ABox-Erstellung (A3.FA3)	n.a	(X)	n.a.	X	X
Serialisierung der ABox in unterschiedlichen Formaten (A3.FA4)	n.a	X	X	X	X
Integration in das Engineering von fertigungstechnischen Produktionssystemen (A3.FA5)	n.a	-	-	-	X
Modellierung struktureller, funktionaler und verhaltensorientierter Fakten eines Produktionssystems (A3.FA6)	n.a	(X)	(X)	(X)	X
Erzeugung konsistenter und kohärenter ABox-Inhalte (A3.NFA1)	n.a	(X)	(X)	(X)	X
Erzeugung verständlicher ABox-Inhalte (A3.NFA2)	n.a	-	-	X	X

„X“ - erfüllt; „(X)“ - teilweise erfüllt;
„-“ - nicht erfüllt; „n.a“ - nicht angewendet

V-3.2. Forschungszielbezogene Bewertung

Der vorliegende Abschnitt diskutiert, inwiefern das in Definition I.1 dargestellte Forschungsziel erreicht werden konnte. Wie in den vorangegangenen Abschnitten V-3.1.1 bis V-3.1.3 dargestellt, wurden die Forschungsteilziele basierend auf den definierten Anforderungen erreicht. Dies kann als Indiz für die Erreichung des Forschungsziels gewertet werden. Zusätzlich soll nachfolgend diskutiert werden, inwiefern durch die Nutzung der im Rahmen dieser Arbeit entwickelten Artefakte eine Steigerung der Wiederverwendung bzw. Effizienzsteigerung im Entwicklungsprozess zu beobachten ist. Hierzu wird als Referenz die Arbeit von [Pau18] herangezogen, in welcher die Kosten für die Erstellung von Tripeln in Knowledge Graphs in der Domäne des Semantic Web untersucht wurden. Demnach entstehen je nach verwendetem Werkzeug Kosten zwischen 0,008 und 6 USD pro Tripel. Die Bandbreite der Kosten beinhaltet sowohl Personal- als auch Entwicklungskosten und ist tendenziell umso niedriger, je höher der

Automatisierungsgrad ist. Es sei jedoch auf den Abschnitt VI-2 verwiesen, da automatisierte Ansätze zur Erstellung von Knowledge Graphs i.d.R. bewusst fehlerbehaftet sind, zugunsten der Effizienz der Erstellung. Dies gilt nicht für die vorliegende Methode, da sie Qualität über Kosten stellt und somit einen deterministischen Ansatz verfolgt. Um die Wiederverwendung bewerten zu können, wurde die Wiederverwendungsquote von TBox-Axiomen über den zeitlichen Verlauf der Durchführung der Fallstudien berechnet. Darüber hinaus wurden die Kosten für die Modellierung der TBox in Euro pro Axiom anhand der Personalkosten und der Gesamtzahl der genutzten Axiome berechnet. Um ebenfalls eine Aussage über eine mögliche Effizienzsteigerung der ABox-Entwicklung treffen zu können, wurden die Kosten für die Erstellung der fallstudienspezifischen Tripel der ABox in Euro pro Tripel berechnet, wobei auch hier die in der Fallstudie angefallenen Personalkosten berücksichtigt wurden. Hierbei wurde eine Trennung in dynamische und quasi-statische Tripel vorgenommen, da die dynamischen Tripel, welche in großer Zahl zur Laufzeit durch Mappings erzeugt werden, das Ergebnis u.U. verzerren können. Als Grundlage der Berechnung wurden die tatsächlich angefallenen Personenmonate (bzw. Arbeitsstunden) für die Erstellung der TBox und ABox in den jeweiligen Fallstudien herangezogen, sowie ein Kostensatz von 49,35 € pro Stunde für einen wissenschaftlichen Mitarbeiter. In der Berechnung wurde die Erstellung des Werkzeugs „LIOnS“ nicht mit berücksichtigt, welche für einen prototypischen Zustand in etwa Kosten von 26000€ erzeugte. Der Grund hierfür ist, dass die Kosten keinem Projekt direkt zugeordnet werden können, da das Werkzeug nicht innerhalb der Projekte, oder speziell für diese Projekte, entwickelt wurde. Das Ergebnis dieser Berechnungen ist in Tabelle V.17 zu finden und wird nachfolgend erläutert.

Tabelle V.17.: Auswertung der Modellierungskosten in den Fallstudien

Fallstudie	WV ODMs	WV Axiome	€/Axiom	€/Tripel	€/Tripel (o.d.)
WA-1	0	0	9,234	18,302	18,302
WA-2	0,667	0,648	4,463	11,673	11,813
AF	0,6	0,357	3,556	1,705	1,705
PA-1	0,75	0,993	0,291	0,019	2,903
PA-2	1	1	0	0,010	0,208

„WV“ - Wiederverwendungsquote; „o.d.“ - ohne dynamische Tripel;

Wie der Tabelle zu entnehmen ist, konnten Ontologie-Design-Muster mit Ausnahme der ersten Fallstudie WA-1, stets wiederverwendet werden (WV ODMs). Dies traf auch zu, obwohl die jeweiligen Fallbeispiele in den abzubildenden Inhalten variierten, oder wie in Fallstudie PA-1 fallspezifische Ergänzungen notwendig waren. Betrachtet man die Wiederverwendungsquote auf Ebene der TBox-Axiome und nicht auf Ebene der einzelnen ODMs so ergibt sich ein etwas anderes Bild. Durch die Wiederverwendbarkeit der ODMs ergibt sich stets auch eine Wiederverwendungsquote der TBox-Axiome (WV Axiome) größer null in den Fallstudien. Ist in einer Fallstudie jedoch eine Informationsressource zu formalisieren, welche eine hohe Menge an Begriffen und Relationen aufweist (z.B. DIN 8580 [DIN 8580] oder OPC-UA nach [Per+19]), so senkt dies die Wiederverwendungsquote der TBox-Axiome erheblich (siehe AF) oder steigert sie erheblich (siehe PA-1). Ebenfalls erkennbar ist, dass die Wiederverwendung grundsätzlich einen Einfluss auf die Kosten pro TBox-Axiom (€/Axiom) hat und somit die Effizienz steigert. Basierend auf der ersten und der vorletzten industri-

ellen Fallstudie (WA-1 bzw. PA-1) ist eine Effizienzsteigerung von über 97% in den Kosten pro entwickeltem TBox-Axiom erkennbar. Selbst bei ausschließlicher Berücksichtigung der im Rahmen der Fallstudien entwickelten Ontologien, also dem Vergleich der Kosten pro TBox-Axiom (€/Axiom) zwischen Fallstudie WA-1 und AF liegt eine Effizienzsteigerung von ungefähr 60% vor. Auch hinsichtlich der Methodik zur ABox-Entwicklung ist eine Effizienzsteigerung ersichtlich. Während in der Fallstudie WA-1 die im Rahmen dieser Arbeit entwickelte Methodik keinen Einsatz fand und eine rein manuelle Modellierung durchgeführt wurde, konnte im Rahmen der Fallstudie WA-2 bereits mit den assertionalen Mustern der TBox-Entwicklung eine Effizienzsteigerung erzielt werden. Dies ist durch die geringeren Kosten pro Tripel (€/Tripel) ersichtlich. Eine erhebliche Verbesserung stellte sich jedoch erst durch Nutzung von automatisierten Verfahren (PA-1) und semi-automatisierten Verfahren entsprechend der Methode dieser Arbeit (AF) oder hybriden Verfahren entsprechend der Methode dieser Arbeit (PA-2) ein. Basierend auf der ersten und letzten Fallstudie (WA-1 bzw. PA-2) ergibt sich ohne Berücksichtigung dynamischer Tripel eine Effizienzsteigerung von über 98%.

Anhand der dargestellten Berechnungen ist somit erkennbar, dass einerseits die Wiederverwendung ontologischer Artefakte mit der im Rahmen dieser Arbeit beschriebenen Methode gefördert wird und andererseits eine effiziente Entwicklung ontologischer Artefakte ermöglicht wird. Dies wird auch durch einen Vergleich mit den eingangs erwähnten Studie nach [Pau18] untermauert, wonach die Erstellung von Tripeln in der Semantic Web Domäne Kosten zwischen 0,008 und 6 USD erzeugt. Entsprechend der oben dargestellten Berechnung entstehen durch die Anwendung der im Rahmen dieser Arbeit entwickelten Methodik vergleichbare Kosten. Es ist somit anzunehmen, dass eine im Vergleich zu Ansätzen aus der Semantic Web Domäne vergleichbar effiziente Methodik für die Domäne der Automatisierungstechnik entworfen wurde. Es sei allerdings nochmals darauf hingewiesen, dass die vorliegende Methodik dies ohne Inkaufnahme von fehlerbehafteten ontologischen Artefakten leistet.

VI. Zusammenfassung und Ausblick

Im Rahmen der vorliegenden Arbeit wurde eine Methodik für das Engineering ontologischer Modelle in der Automatisierungstechnik entworfen. Der nachfolgende Abschnitt VI-1 fasst die wesentlichen Aspekte der vorliegenden Arbeit zusammen. Abschnitt VI-2 beendet die vorliegende Arbeit mit einem Ausblick mittels zukünftiger Forschungsfragen, welche sich durch die Erkenntnisse der vorliegenden Arbeit ergeben.

VI-1. Zusammenfassung

Mit einer steigenden Anzahl an Software-Funktionalitäten im Kontext eines Produktionssystems und einer zunehmenden Menge an Daten, Informationen und Wissen über Produktionssysteme steigt der Druck hin zu einer effizienten Nutzung dieser Daten, Informationen und Wissensbestandteile. Ontologien sind ein geeignetes Mittel, um unterschiedlichste Aspekte im Kontext eines Produktionssystems zu formalisieren und somit für entsprechende Systeme aufzubereiten. Sei es um semantische Interoperabilität zwischen Systemen und deren ausgetauschten Daten bzw. Informationen zu gewährleisten, oder um, basierend auf dem spezifizierten Wissen der Systeme, Schlussfolgerungen auszuführen, oder mittels der formalisierten Modelle eine Suche nach Informationen durchzuführen. Die Entwicklung einer Ontologie ist jedoch eine nicht-triviale Aufgabe, welche einen hohen Aufwand unter Einbeziehung unterschiedlicher Stakeholder erfordert. Es ist daher von essentieller Bedeutung, dass diese hohen Aufwände über mehrere Projekte verteilt werden können, sodass sich langfristig eine gewisse Effizienz im Umgang mit den relevanten Technologien einstellen kann und deren Nutzung auf diese Weise einen Mehrwert im industriellen Umfeld schaffen kann. Die vorliegende Arbeit befasste sich daher mit dem Ziel, eine Engineering-Methode zu entwickeln, welche durch geeignete Automatisierung und Strukturierung von Modellierungstätigkeiten sowie ein entsprechendes Wiederverwendungskonzept die notwendige Effizienz für eine Anwendung im industriellen Umfeld schafft. Zur Bestimmung der notwendigen Gestalt einer derartigen Methode wurde der relevante Stand der Wissenschaft in der Domäne der Automatisierungstechnik untersucht, um ein detaillierteres Problemverständnis hinsichtlich der Entwicklung von Ontologien zu gewinnen. Basierend darauf wurden drei Methoden entwickelt, welche für die Anforderungserhebung an Ontologien, die Entwicklung der terminologischen Box und die Entwicklung der assertionalen Box einer Ontologie genutzt werden können.

Im Rahmen der Methodik zur Anforderungserhebung erfolgte im Rahmen dieser Arbeit eine Untersuchung des relevanten Stands der Wissenschaft zur Anforderungserhebung an Ontologien. Basierend darauf wurde eine modellbasierte Methodik unter Nutzung von SysML-/UML-Modellen entwickelt, welche die Anforderungserhebung an Ontologien mit der Anfor-

derungserhebung der sie nutzenden Systeme kombiniert. Auf diese Weise werden ontologische Anforderungen erhoben, welche durch ihr enges Zusammenspiel mit den Anforderungen die Systeme, welche die Ontologie nutzen, einen hohen Detailgrad und Fokus auf das Entwicklungsziel besitzen. Die Methodik ist derart schlank gestaltet, dass sie im Rahmen von entwicklungsbegleitenden Workshops durchführbar ist. Im Rahmen dessen werden die relevanten Entwicklungsziele von Systemen und Ontologien aufgenommen, Systemfunktionen definiert, die eine Ontologie benötigen, und Systemstrukturen sowie Systeminteraktionen dargestellt. Auf der Basis dieser Spezifikationen werden die Anforderungen an eine Ontologie in Form von Kompetenzfragen und Antworten erhoben und ebenfalls modellbasiert aufgenommen. Auf diese Weise ist eine Nachverfolgbarkeit der ontologischen Anforderungen möglich, was insbesondere in größeren Entwicklungsprojekten von Relevanz ist.

Basierend auf den somit erhobenen Anforderungen an die zu erstellende Ontologie kann die terminologische Box der Ontologie entworfen werden. Für den Entwurf der hierfür bestimmten Methode wurde im Rahmen der vorliegenden Arbeit der relevante Stand der Wissenschaft zur Ontologie-Entwicklung analysiert und hinsichtlich der Nutzbarkeit in der Domäne der Automatisierungstechnik bewertet. Die entwickelte Methode führt eine Formalisierung des für den Anwendungsfall relevanten Wissens durch, wobei hierfür auf Informationsressourcen im Sinne von Industriestandards zurückgegriffen wird. Letztere werden in sogenannte Ontologie-Design-Muster überführt, welche den Kernbestandteil des Wiederverwendungskonzeptes der Arbeit darstellen. Die im Rahmen der Methodik definierten Aspekte zur Dokumentation und Dissemination der Ontologie-Design-Muster zielen auf eine hohe Wiederverwendung der entwickelten ontologischen Artefakte ab, sodass die entstehenden Aufwände potentiell über eine Vielzahl an Projekten verteilt werden können.

Für die Beschreibung eines Produktionssystems mit Hilfe einer Ontologie ist die Definition von Fakten über dieses System notwendig, was in der assertionalen Box erfolgt. Wie die Darstellung des Stands der Wissenschaft gezeigt hat, existiert bereits eine Vielzahl an Methoden für die automatisierte Erstellung der assertionalen Box, welche im Rahmen der Methodik aufgegriffen werden. Die im Rahmen dieser Arbeit vorgestellte Methodik ergänzt ein auf Nutzereingaben basierendes semi-automatisches Verfahren zur Modellierung der assertionalen Box und bietet mit Hilfe verschiedener Metriken die Möglichkeit, projektspezifisch die effizientesten Ansätze zu erkennen, bevor Aufwände entstehen. Darüber hinaus bietet die vorliegende Methodik eine Einbindung in das Engineering fertigungstechnischer Produktionssysteme, mit dessen Hilfe eine strukturiertes Vorgehen zur Modellierung der ABox für ein Ontologie-Entwicklungsprojekt geschaffen wird.

Zur Überprüfung der Zielerreichung wurde im Rahmen der vorliegenden Arbeit ein Evaluationsvorgehen basierend auf Fallstudien gewählt, wobei industrielle Fallstudien durchgeführt wurden. Im Rahmen der fünf industriellen Fallstudien konnte einerseits die Anforderungserfüllung überprüft werden, aber auch die Effizienzsteigerung in der Entwicklung der terminologischen und assertionalen Box mit Hilfe der entwickelten Methoden. Als wesentliche Treiber für die Effizienzsteigerung haben sich im Rahmen der Evaluation die Wiederverwendbarkeit der Ontologie-Design-Muster im Rahmen der Entwicklung der terminologischen Box herausgestellt, als auch die strukturierte Verwendung automatisierter und semi-automatisierter Verfahren zur Erstellung der assertionalen Box.

VI-2. Ausblick

Im Rahmen der Evaluation in Kapitel V wurde aufgezeigt, dass die in Kapitel I definierten Ziele zur Verbesserung des Stands der Wissenschaft erreicht werden konnten und daher mit Hilfe der vorliegenden Methodik eine effizientere Entwicklung ontologischer Modelle in der Automatisierungstechnik erreicht werden kann. Dennoch zeichnen sich einige Aspekte ab, die entweder eine weitere Effizienzsteigerung in der Entwicklung erhoffen lassen oder den Methoden zugrunde liegende Annahmen relativieren könnten. Diese Aspekte sollen nachfolgend als potentieller Forschungsbedarf vorgestellt werden.

Der Methodik zur Entwicklung der terminologischen Box entsprechend Kapitel III liegt die Annahme zu Grunde, dass zur Laufzeit der Systeme nicht zwei Systeme mit ähnlichen aber unterschiedlichen terminologischen Boxen interagieren müssen. Wird eine Ontologie jedoch für ein System bzw. eine Systemgruppe entwickelt, deren Zusammensetzung zur Laufzeit vollständig oder zumindest teilweise unbekannt ist (siehe z.B. [Hil+19*]), muss diese Annahme relativiert werden. In diesem Falle sind bereits in der Ontologie-Entwicklung nicht nur Ontologie-Design-Muster und Alignment-Ontologien für ein spezifisches Projekt zu erstellen, sondern auch Alignment-Ontologien zwischen unterschiedlichen, aber inhaltlich ähnlichen, Ontologie-Design-Mustern. Soll durch eine Gruppe von Systemen bspw. eine Ontologie zur Beschreibung von Prozessen genutzt werden, so ist zusätzlich zum Ontologie-Design-Muster der Formalisierten Prozessbeschreibung (siehe Abschnitt III-4.1) eine Alignment-Ontologie zu entwickeln, welche die Konzepte der Formalisierten Prozessbeschreibung mit den Konzepten anderer Prozessbeschreibungen vereint. Das hierbei zu beachtende Wissensgebiet ist das Ontology-Matching (siehe z.B. [EuSh13]).

Hinsichtlich der Effizienz der Methodik zur Entwicklung der terminologischen Box entsprechend Kapitel III fällt die stark manuelle Prägung der vorliegenden Methodenteile auf. Mögliche Ansatzpunkte für eine Automatisierung von Modellierungsaktivitäten sind vor allem in frühen Methodenteilen erkennbar. Die Methodik sieht beispielsweise keine automatisierte Exploration von existierenden Ontologien oder Dokumenten von Standardisierungsgremien vor, um deren Eignung zur Erfüllung der Anforderungen eines Projektes zu überprüfen. Eine Möglichkeit zur automatisierten Exploration von existierenden Ontologien ist im Ansatz von [Ren+14] erkennbar. Durch die Definition von Archetypen von Kompetenzfragen ist auch die Exploration existierender Ontologien möglich. Die in einem Projekt auftretenden Kompetenzfragen müssen hierzu in entsprechende Archetypen überführt werden, indem die Variablen der Archtypen auf die Anforderungen des Projektes angepasst werden. Aus den Archetypen lassen sich dann entsprechende SPARQL-Abfragen erstellen, welche gegen existierende Ontologien ausgeführt werden können. Hinsichtlich der stärker automatisierten Einbindung von Dokumenten von Standardisierungsgremien ist natürlich darauf zu hoffen, dass zukünftige Standards ihr Vokabular bereits entsprechend typisieren, bspw. mit Hilfe von Semantic Web Technologien, sodass deren Eignung auch automatisiert erkannt werden kann. Alternativ ist auch der Einsatz von „Natural-Language-Processing“ Verfahren denkbar, in welchen die Dokumente automatisiert ausgelesen werden und somit zumindest eine Extraktion der relevanten Konzepte ermöglicht wird [YCY19a]. Ein weiterer Ansatzpunkt ist die automatisierte Erstellung von OWL-Ontologien basierend auf der Darstellung der UML-Klassendiagramme. Wie in Abschnitt III-3.3 dargestellt, existiert bereits eine Logik zur Überführung von UML

nach OWL basierend auf den gewählten Modellelementen, welche die Grundlage für eine entsprechende Automatisierung liefern kann.

Bezüglich der in Kapitel IV dargestellten Methodik zur Entwicklung der assertionalen Box der Ontologie ist insbesondere Forschungsbedarf hinsichtlich automatisierter Verfahren erkennbar, welche auf die Domäne der Automatisierungstechnik angepasst sind. Wie die Analyse existierender automatisierter Verfahren gezeigt hat, existieren nur wenige Verfahren, welche auf Formate im Kontext des Engineerings angepasst sind. So musste im Rahmen der Anwendung der Methodik im Kontext der Evaluation stets ein nicht unerheblicher Aufwand betrieben werden, um Verfahren der informationstechnischen Domäne auf die vorliegenden Artefakte anzupassen. Beispielsweise mussten bestimmte Artefakte erst in ein für die jeweilige Mappingsprache nutzbares Format überführt werden. Es hat sich gezeigt, dass die Verfügbarkeit einer Mapping-Sprache für Artefakte der Gewerke des Engineerings (Maschinenbau, Elektrotechnik, Automatisierungstechnik) und die Verfügbarkeit einer entsprechenden Mapping-Engine eine erhebliche Aufwandsreduktion bewirkt hätte. Des Weiteren liegt der Methodik die Annahme zu Grunde, dass die erstellte Ontologie als komplementär und nicht als substitutiv zu existierenden Informationsmodellen, wie z.B. AutomationML, genutzt wird. Im Rahmen der verfügbaren Fallstudien ergab sich jedoch nicht die Gelegenheit die jeweiligen Stärken und Schwächen methodisch zu kombinieren, da keines der betrachteten Industrieunternehmen AutomationML als Austauschformat nutzte und daher eine Evaluation nicht möglich gewesen wäre. Das Potential einer komplementären Nutzung ist jedoch, wie in [HuHe19] dargestellt, nicht von der Hand zu weisen und bedarf weiter Untersuchung, auch im Hinblick auf eventuell redundante Engineering-Tätigkeiten der vorliegenden Methodik und der Erstellung von Informationsmodellen wie z.B. AutomationML. Eine weitere Annahme die dem Ansatz zugrunde liegt ist die Forderung nach einem deterministischen Ergebnis eines automatisierten Verfahrens. Die automatisierte Erzeugung einer ABox darf demnach, wie im Engineering üblich, keine Unsicherheit bezüglich des Ergebnisses erzeugen, da die resultierende ABox möglicherweise für Applikationen genutzt wird, die auf die Verlässlichkeit der Daten setzt. Trifft diese Annahme nicht zu, eröffnen sich andere methodische Möglichkeiten. Beispielsweise wäre die Definition von generellen Mappings von Artefakten in eine ABox möglich, die dann nachträglich manuell (mit Hilfe von Nutzereingaben) oder automatisiert (z.B. durch Kombination mit anderen Artefakten) um weitergehende Semantiken ergänzt werden.

Anhang

A. Tabellen

Die in [Ham17] entwickelten Qualitätsmerkmale für ein Ontologie-Design-Muster sind in Tabelle A.1 zu finden. Indikatoren die auf die Erfüllung der Merkmale hinwiesen, sind in Tabelle A.3 und Tabelle A.2 zu finden.

Tabelle A.1.: Qualitätsmerkmale eines Ontologie-Design-Musters, in Anlehnung an [Ham17]

Qualitätsmerkmal	Sub-Merkmal	Beschreibung
Funktionale Eignung	Vollständigkeit	Grad zu dem ein ODM die Anforderungen erfüllt (z.B. Kompetenzfragen).
	Angemessenheit	Grad zu dem ein ODM mit Hilfe einfacher SPARQL Anfragen genutzt werden kann.
	Konsistenz	Grad zu dem ein ODM für sich konsistent ist.
	Genauigkeit	Grad zu dem ein ODM die reale Welt akkurat beschreibt.
Nutzbarkeit	Erkennbarkeit der Angemessenheit	Grad zu dem potentielle Nutzer die Eignung eines ODM für ein Projekt erkennen können.
	Lernbarkeit	Grad zu dem die Struktur und die vorgesehene Nutzung eines ODM durch Nutzer erlernt werden kann.
	Bedienbarkeit	Grad zu dem ein ODM Attribute besitzt, die seine Anwendung und Nutzung erleichtern.
	Nutzerfehlervermeidung	Grad zu welchem ein ODM eine falsche Nutzung verhindet.
	Nutzerschnittstellenästhetik	Grad zu dem die textuelle oder grafische Dokumentation den Nutzern zusagt.
	Zugänglichkeit	Grad zu dem die Dokumentation des ODM für Nutzer unterschiedlicher Fähigkeitsstände zugänglich ist.
Wartbarkeit	Modularität	Grad zu dem die Definitionen eines ODM unabhängig zu denen in anderen Ontologien sind.
	Analysierbarkeit	Grad der Effektivität und Effizienz, mit der es möglich ist, den Einfluss einer Änderung von Teilen eines ODMs zu bestimmen, oder einen ODM auf Fehler zu überprüfen, oder auf zu modifizierende Bestandteile zu prüfen.
	Modifzierbarkeit	Grad zu dem ein ODM effektiv und effizient modifiziert werden kann, ohne die Absenkung seiner Qualität.
	Testfähigkeit	Grad der Effektivität und Effizienz, mit der Test-Kriterien für einen ODM festgelegt und Tests durchgeführt werden können.

	Stabilität	Erwartung der Änderungen des ODM in der Zukunft.
Kompatibilität	Wiederverwendbarkeit	Grad zu der ein ODM in mehr als einem Projekt, oder mehr als einer Ontologie wiederverwendet werden kann.
	Ko-Existenz	Grad zu dem ein ODM mit anderen ODMs als Modul in einer Ontologie koexistieren kann, ohne andere Teile oder ODMs negativ zu beeinträchtigen.
	Interoperabilität	Grad zu dem ODM Konzepte mit anderen ODMs teilt.
Performance-Effizienz		Kann in Form von durchschnittlicher Zeit für Konsistenzprüfen, Abfragen oder Inferenzen diskutiert werden.

Tabelle A.2.: Auswahl von Modellindikatoren für die Erfüllung von Qualitätsmerkmalen, in Anlehnung an [Ham17]

Indikator	Beschreibung	Betroffenes Merkmal
Annotationsverhältnis	Verhältnis aus der Anzahl von Annotationen und der Anzahl aller Klassen und Properties geht gegen 1	Nutzbarkeit (positiv), Wartbarkeit (positiv)
Average Class In-Degree	Der Durchschnitt der eingehenden RDF-Kanten, die eine Klasse des ODM hat, ist hoch	Performance-Effizienz (negativ)
Average Class Out-Degree	Der Durchschnitt der ausgehenden RDF-Kanten, die eine Klasse des ODM hat, ist hoch	Performance-Effizienz (negativ)
Class/Axiom Verhältnis	Verhältnis der Anzahl der Klassen zur Anzahl der Axiome geht gegen null	Analysierbarkeit (positiv)
Class/Property Verhältnis	Verhätlnis zwischen der Anzahl der Klassen und der Summe aller Properties geht gegen null	Nutzbarkeit (positiv)
Benennungsstruktur	Die Benennung von Klassen und Properties folgt einem Schema und ist konsistent vorgenommen.	Modifizierbarkeit (negativ), Nutzbarkeit (positiv)
Menschenlesbare Benennungen	IRI-Elemente des ODM sind aus Begriffen zusammengesetzt, die im menschlichen Vokabular existieren	Modifizierbarkeit (negativ), Testbarkeit (positiv)
Property-Domain Restriktionsverhältnis	Das Verhätlnis von Properties mit Domain-Restriktion zu der Summe aller Properties geht gegen 1	Lernbarkeit (positiv), Wiederverwendbarkeit (negativ), Performance-Effizienz (negativ)
Property-Range Restriktionsverhältnis	Das Verhätlnis von Properties mit Domain-Restriktion zu der Summe aller Properties geht gegen 1	Lernbarkeit (positiv), Wiederverwendbarkeit (negativ), Performance-Effizienz (negativ)
Größe	Die Summe aller Klassen und Properties ist hoch	Lernbarkeit (negativ), Analysierbarkeit (negativ)
Breite der Klassenstruktur	Die durchschnittliche Anzahl der Klassen auf einer Ebene (z.B. unter owl:Thing) ist hoch	Nutzbarkeit (negativ)

Tiefe der Klassenstruktur	Die durchschnittliche Anzahl der Spezialisierungen ausgehend von einer Klasse auf der Ebene unter owl:Thing ist hoch	Nutzbarkeit (negativ)
Verwirrung	Das Verhältnis von Klassen die mehr als eine Super-Klasse haben zu der Summe aller Klassen geht gegen 1	Nutzbarkeit (negativ), Kompatibilität (negativ), Performance-Effizienz (negativ)

Tabelle A.3.: Dokumentationsrelevante Indikatoren für die Erfüllung von Qualitätsmerkmalen, in Anlehnung an [Ham17]

Indikator	Beschreibung	Betroffenes Merkmal
Zugehörige textuelle Spezifikation	Existenz einer textuellen Spezifikation für den ODM	Erkennbarkeit der Angemessenheit (positiv)
Gängige Problemfallbeschreibungen	Textuelle Spezifikation enthält eine Beschreibung üblicher Fehler in der Anwendung	Nutzerfehlervermeidung (positiv)
Kompetenzfragenanzahl	Das Verhältnis aus Anzahl der Kompetenzfragen und Größe des ODM (siehe Größe) geht gegen 1	Erkennbarkeit der Angemessenheit (positiv), Lernbarkeit (positiv)
Vollständigkeit der Dokumentation	Jede Klasse oder Property besitzt eine textuelle Spezifikation.	Nutzbarkeit (positiv)
Minimalismus der Dokumentation	Die Dokumentation enthält keine nicht notwendigen Bestandteile	Lernbarkeit (positiv)
Illustration der Struktur	Die Dokumentation enthält eine Darstellung der Klassen und Properties und wie diese miteinander in Verbindung stehen	Erkennbarkeit der Angemessenheit (positiv), Lernbarkeit (positiv)
Anzahl der Nutzungsbeispiele	Anzahl der geschriebenen Beispiele wie der ODM genutzt werden soll	Erkennbarkeit der Angemessenheit (positiv), Lernbarkeit (positiv)
Illustration der Nutzungsbeispiele	Anzahl der Illustrationen, wie ein ODM genutzt werden soll	Lernbarkeit (positiv)
Qualitätskontrolle	Es ist ersichtlich, dass der ODM einen Prozess der Qualitätskontrolle durchlaufen hat	Erkennbarkeit der Angemessenheit (positiv)

Für die Artefaktanalyse der Fallstudie in Abschnitt V-2.3 wurden die in Tabelle A.4 beschriebenen Artefakte herangezogen.

Tabelle A.4.: Beschreibung verfügbarer Artefakte

Name	Format	Referenz	Herkunft	Kurzbeschreibung
FPB.js	.json	VDI 3682	Erstellt mit Hilfe des Werkzeugs FPB.js [Nab19@].	Das Artefakt enthält Informationen entsprechend der Norm [VDI/VDE 3682-1] über die Anlage.

3D-CAD	.stp	VDI 2206	Erstellt im Rahmen des Engineerings der Anlage.	Das Artefakt enthält eine geometrische Darstellung der Anlage, sowie strukturelle Bestandbeziehungen (Baugruppen und Komponenten) der Anlage.
Meister Task	.csv	VDI 3682	Erstellt im Rahmen von Projekt-Managementtätigkeiten.	Enthält Informationen über Teile der Anlage sowie Prozesse und Relationen zwischen diesen.
Anlagen-prozess-tabelle	.xls	VDI 3682	Erstellt im Rahmen der Fallstudie PA-1.	Enthält Informationen entsprechend der Norm [VDI/VDE 3682-1] über die Anlage.
Prozess-daten	.sql	DIN 61360 (I)	Erstellt durch einen OPC-UA-Server im laufenden Betrieb der Anlage.	Ein OPC-UA-Server schreibt Soll- und Istwerte eines laufenden Prozesses in eine SQL-Datenbank.
UPC-UA-Server	.json	OPC UA	Erstellt im Rahmen des Engineerings der Anlage.	Der Server enthält OPC Variablen, welche Soll- und Istwerte eines laufenden Prozesses darstellen.
MS Visio Beschreibung	.xml	VDI 3682	Das Artefakt wurde im Rahmen eines anderen Projektes entworfen.	Das Artefakt enthält Informationen entsprechend der Norm [VDI/VDE 3682-1] über die Anlage.

B. Abbildungen

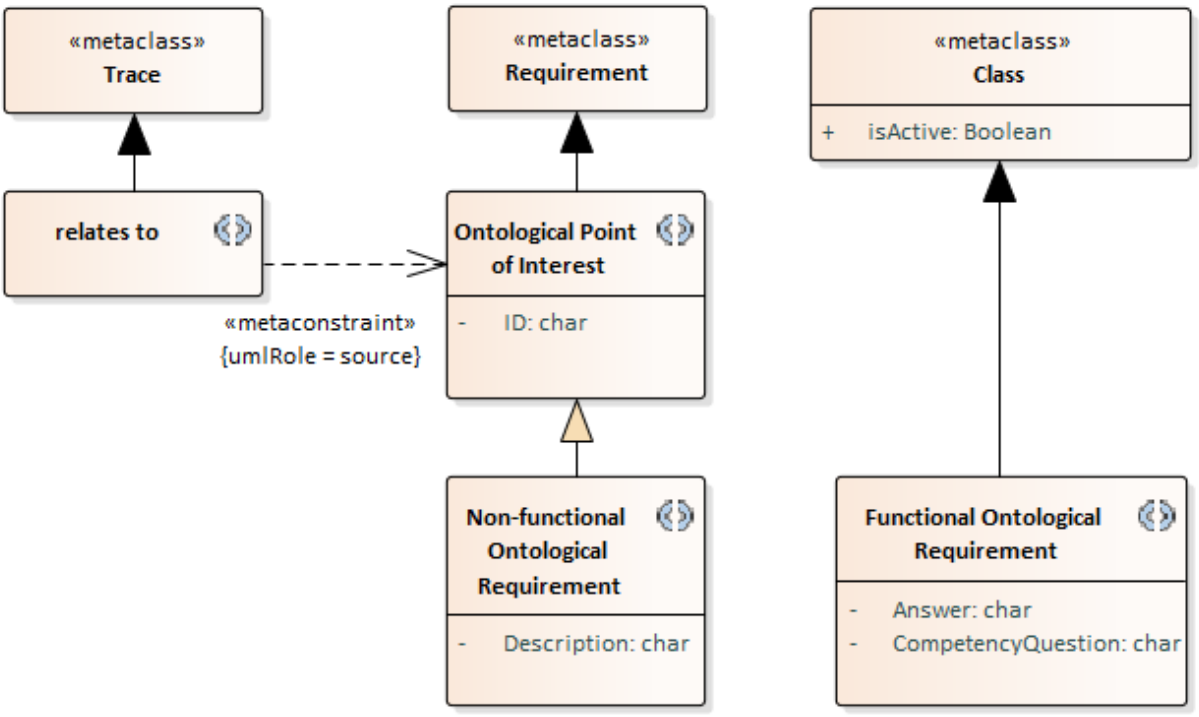

Abbildung B.1.: Profil zur Modellierung ontologischer Anforderungen

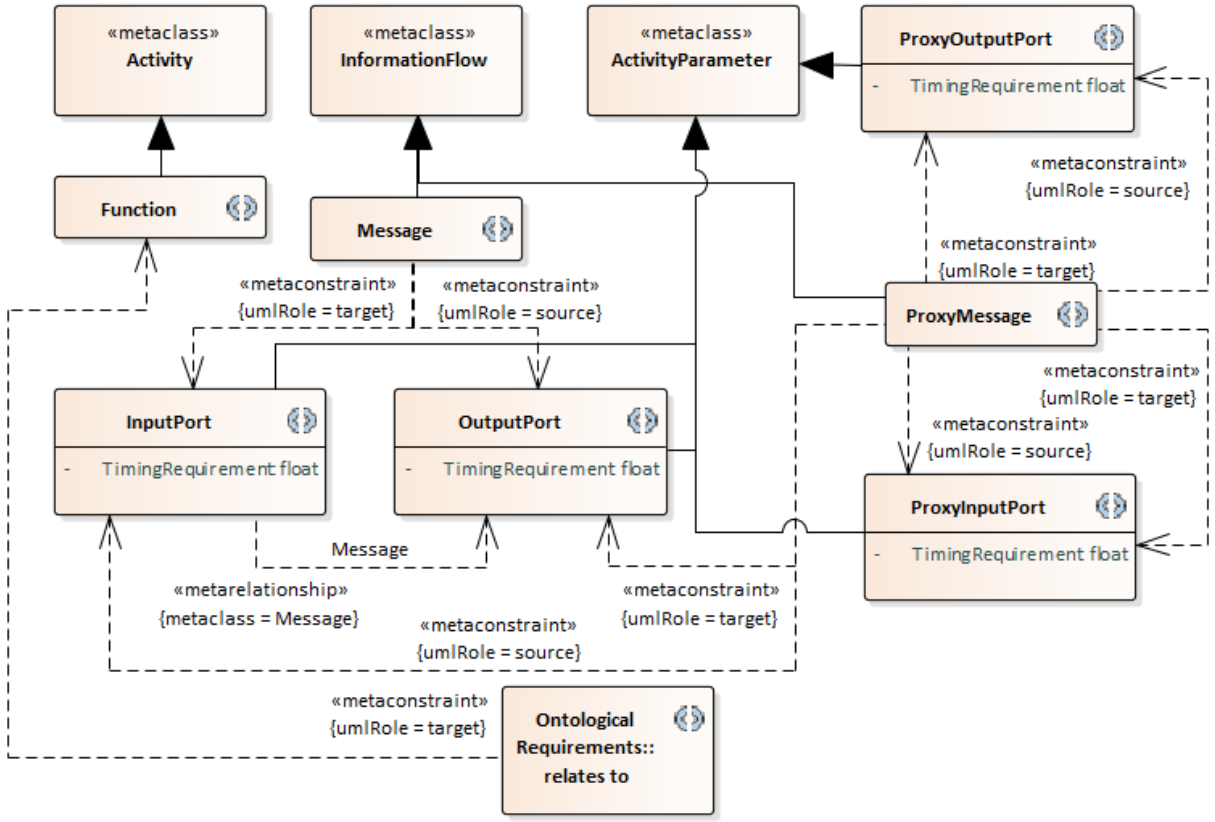

Abbildung B.2.: Profil zur Modellierung von Funktionsstrukturen

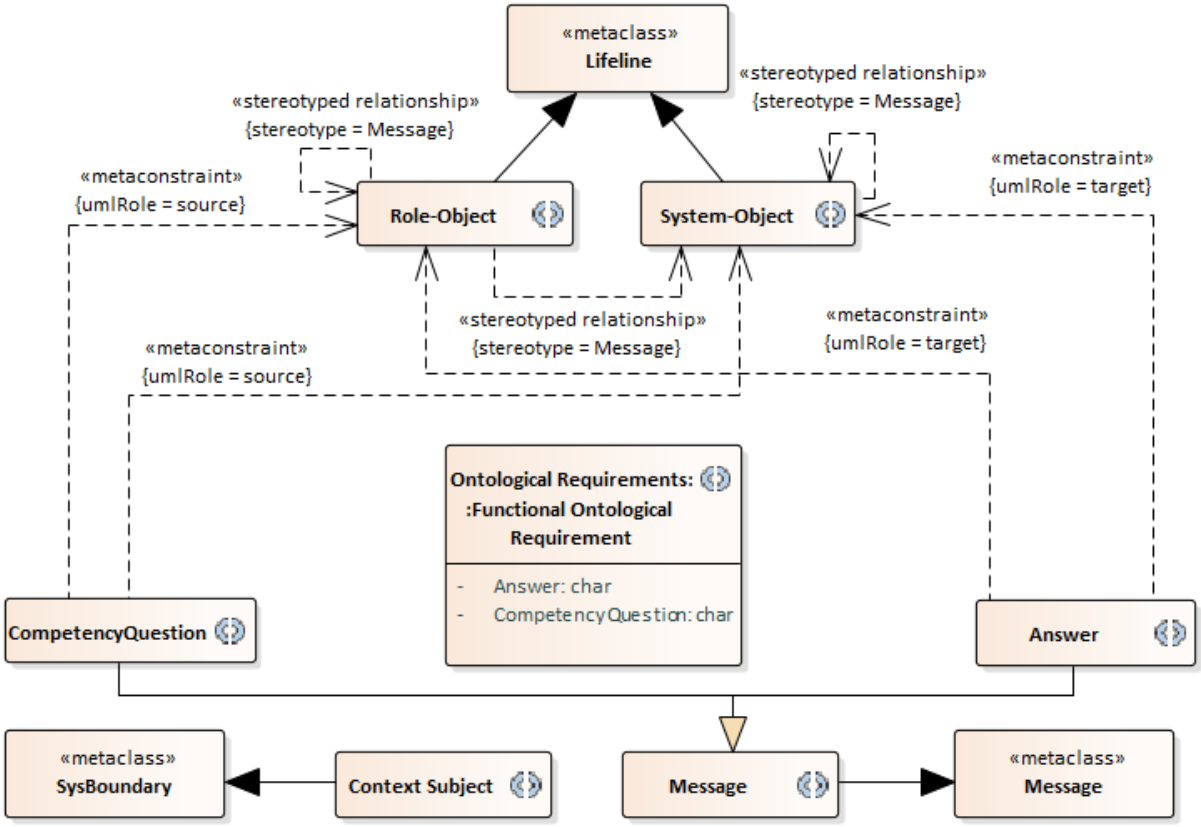

Abbildung B.3.: Profil zur Modellierung des Funktionsverhaltens

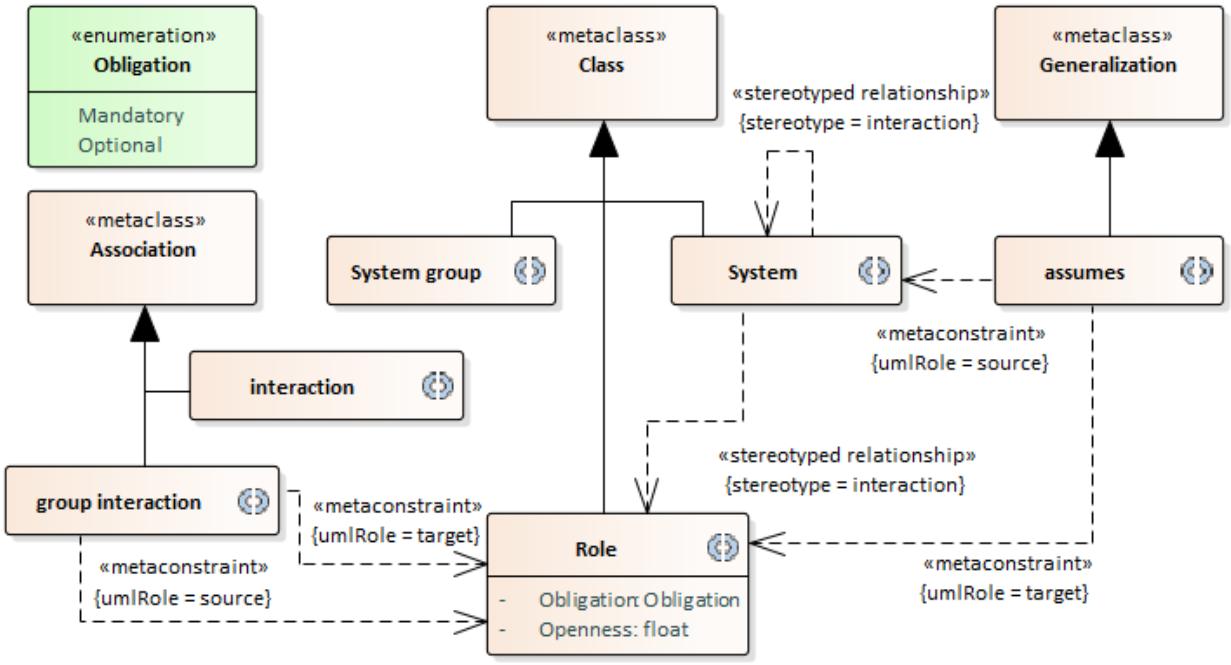

Abbildung B.4.: Profil zur Modellierung der Systemgruppenstruktur

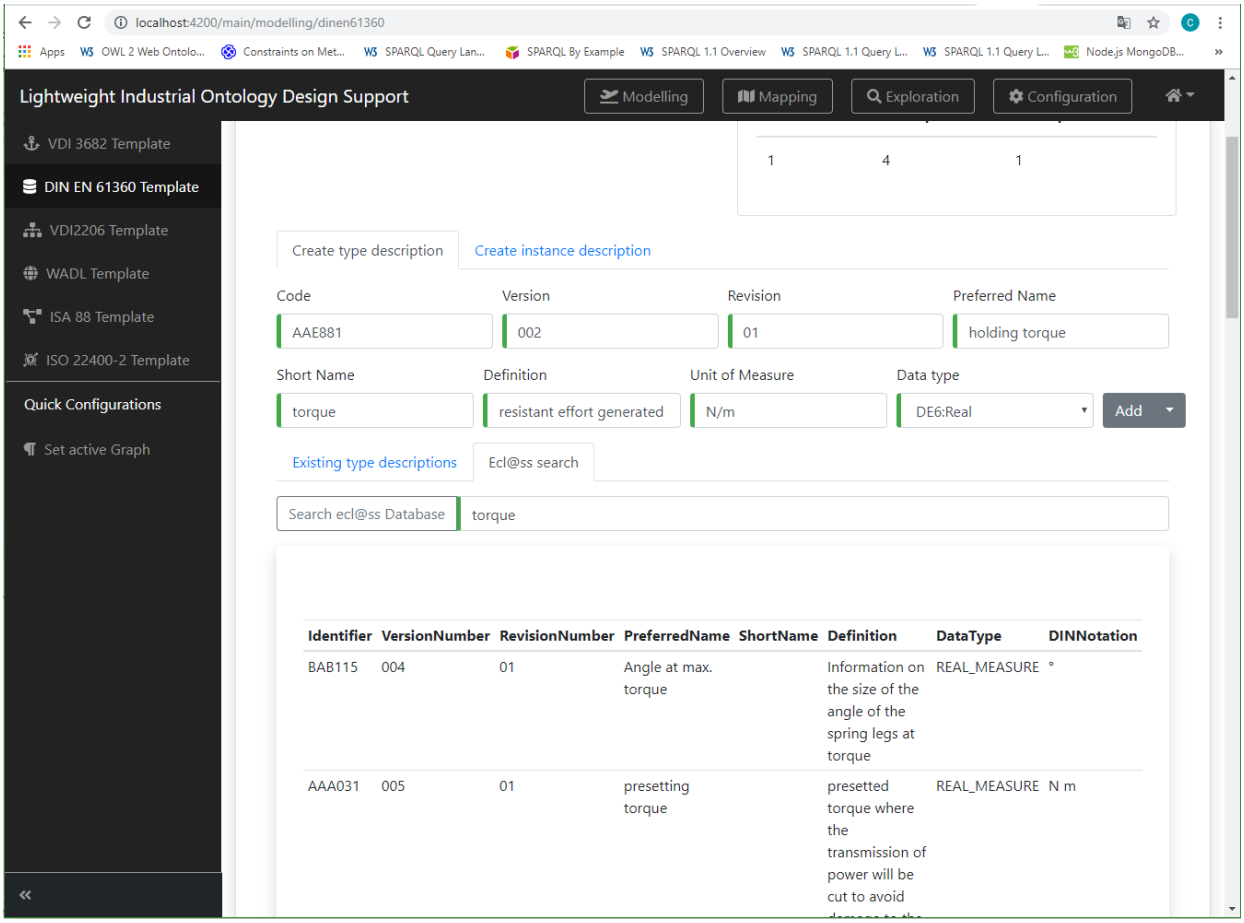

Abbildung B.5.: Merkmalsmodellierung in LIOnS - Typbeschreibungen

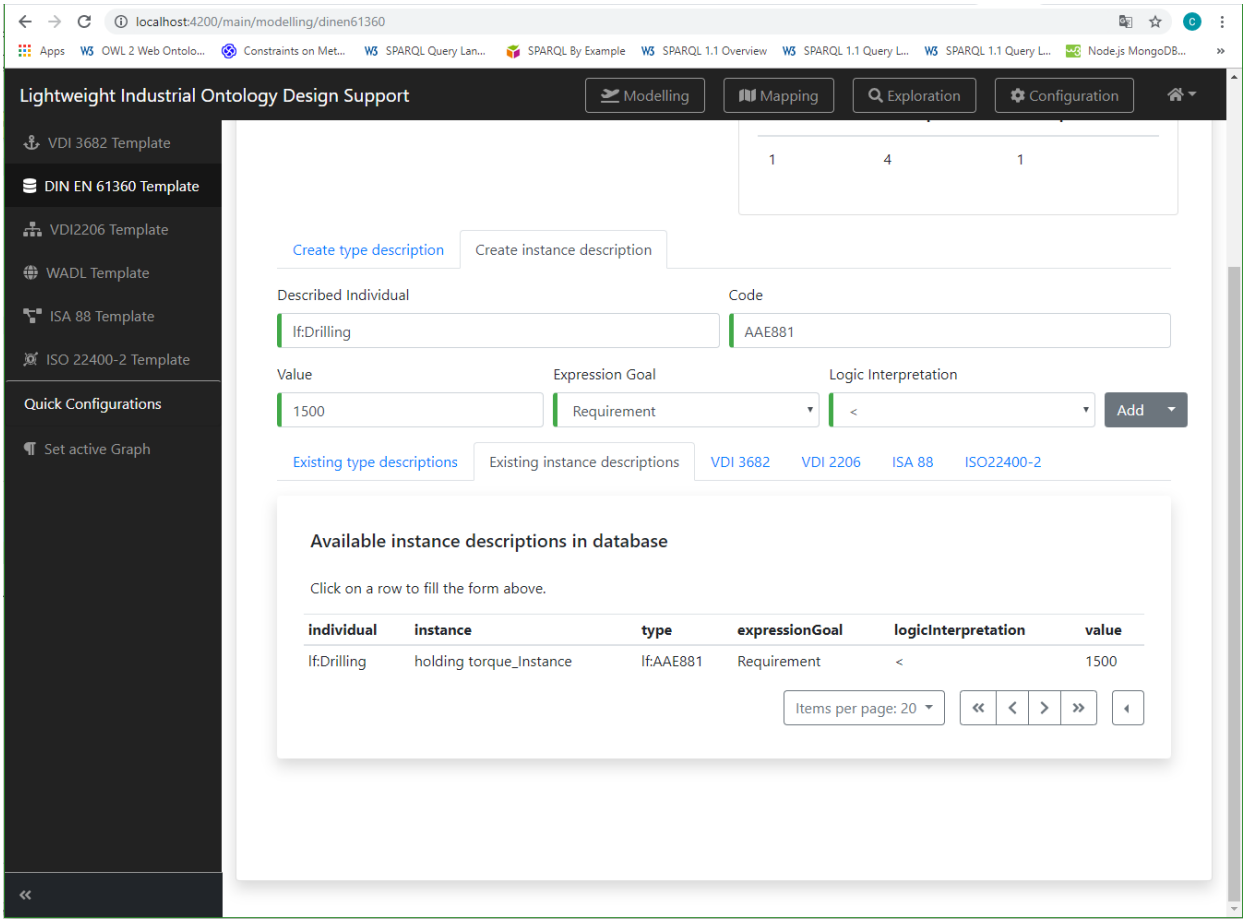

Abbildung B.6.: Merkmalsmodellierung in LIOnS - Instanzbeschreibungen

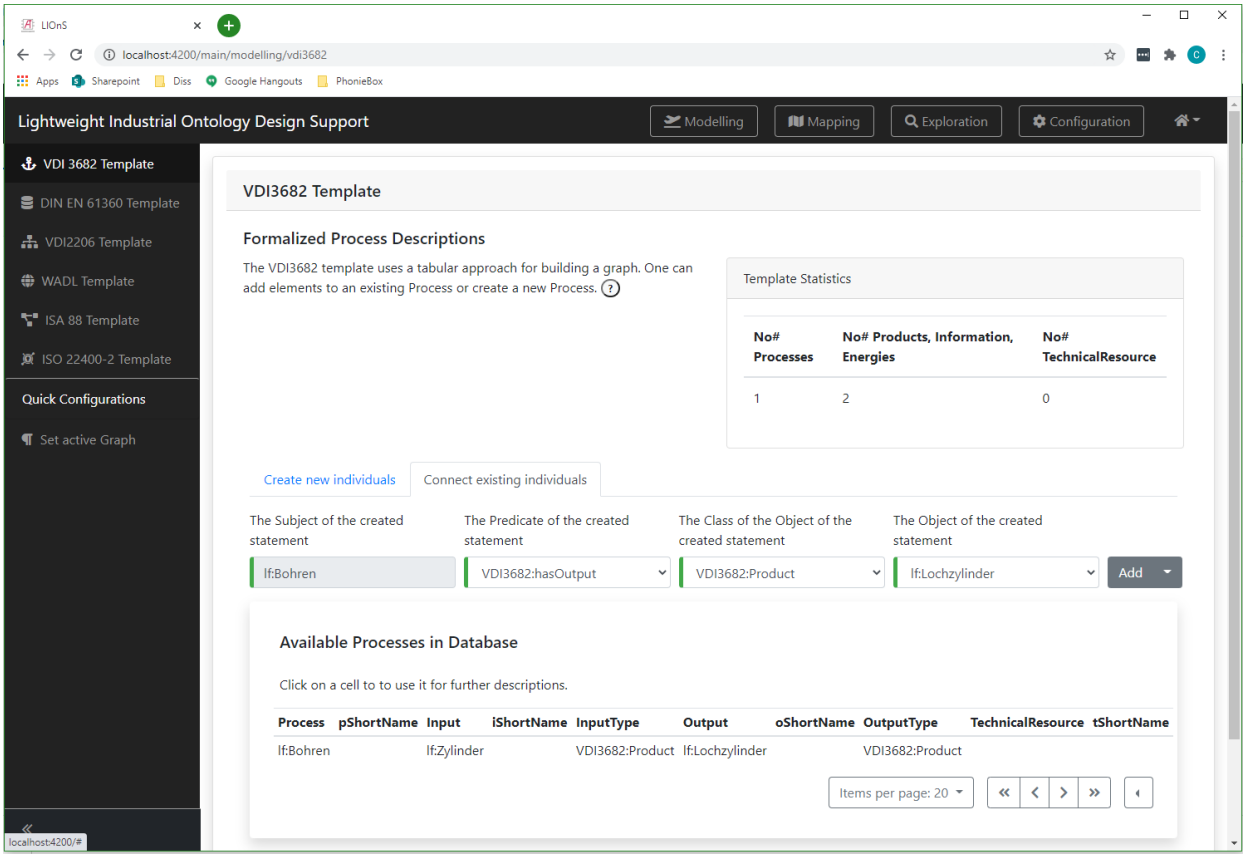

Abbildung B.7.: Tabellarische Modellierung der VDI 3682 in LIOnS

C. Listings

Listing C.1: Minimales assertionales Muster des VDI3682 ODM

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX VDI3682: <http://www.hsu-ifa.de/ontologies/VDI3682#>
INSERT {
    ?ProcessIndividual rdf:type VDI3682:Process; a owl:NamedIndividual.
    ?ProcessIndividual VDI3682:isWithinSystemBorder ?SystemBorder.
    ?ProcessIndividual VDI3682:has_Input ?InputProduct.
    ?ProcessIndividual VDI3682:has_Output ?OutputProduct.
    ?InputProduct rdf:type VDI3682:Product; a owl:NamedIndividual.
    ?OutputProduct rdf:type VDI3682:Product; a owl:NamedIndividual.
} WHERE {
    BIND(IRI(STR("http://ExampleProcess")) AS ?ProcessIndividual).
    BIND(IRI(STR("http://ExampleSystemBorder")) AS ?SystemBorder).
    BIND(IRI(STR("http://ExampleProductState1")) AS ?InputProduct).
    BIND(IRI(STR("http://ExampleProductState2")) AS ?OutputProduct).
}
```

Listing C.2: Maximales assertionales Muster des VDI3682 ODM

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX VDI3682: <http://www.hsu-ifa.de/ontologies/VDI3682#>
INSERT {
    ?ProcessIndividual rdf:type VDI3682:Process; a owl:NamedIndividual.
    ?ProcessIndividual VDI3682:isWithinSystemBorder ?SystemBorder.
    ?ProcessIndividual VDI3682:has_Input ?InputProduct.
    ?InputProduct VDI3682:isOnSystemBorder ?SystemBorder;
        rdf:type VDI3682:Product; a owl:NamedIndividual.
    ?ProcessIndividual VDI3682:has_Input ?InputEnergy.
    ?InputEnergy VDI3682:isOnSystemBorder ?SystemBorder;
        rdf:type VDI3682:Energy; a owl:NamedIndividual.
    ?ProcessIndividual VDI3682:has_Input ?InputInformation.
    ?InputInformation VDI3682:isOnSystemBorder ?SystemBorder;
        rdf:type VDI3682:Information; a owl:NamedIndividual.

    ?ProcessIndividual VDI3682:has_Output ?OutputProduct.
    ?OutputProduct VDI3682:isOnSystemBorder ?SystemBorder;
        rdf:type VDI3682:Product; a owl:NamedIndividual.
    ?ProcessIndividual VDI3682:has_Output ?OutputEnergy.
    ?OutputEnergy VDI3682:isOnSystemBorder ?SystemBorder;
        rdf:type VDI3682:Energy; a owl:NamedIndividual.
    ?ProcessIndividual VDI3682:has_Output ?OutputInformation.
    ?OutputInformation VDI3682:isOnSystemBorder ?SystemBorder;
        rdf:type VDI3682:Information; a owl:NamedIndividual.

    ?TechnicalResource VDI3682:TechnicalResourceIsAssignedToProcessOperator ?
        ProcessIndividual; a owl:NamedIndividual.
} WHERE {
    BIND(IRI(STR("http://ExampleProcess")) AS ?ProcessIndividual).
    BIND(IRI(STR("http://ExampleSystemBorder")) AS ?SystemBorder).
    BIND(IRI(STR("http://ExampleProductState1")) AS ?InputProduct).
    BIND(IRI(STR("http://ExampleEnergyState1")) AS ?InputEnergy).
    BIND(IRI(STR("http://ExampleInformationState1")) AS ?InputInformation).
    BIND(IRI(STR("http://ExampleProductState2")) AS ?OutputProduct).
    BIND(IRI(STR("http://ExampleEnergyState2")) AS ?OutputEnergy).
    BIND(IRI(STR("http://ExampleInformationState2")) AS ?OutputInformation).
    BIND(IRI(STR("http://ExampleTechnicalResource")) AS ?TechnicalResource).
}
```

Listing C.3: Assertionales Muster für einen Zustandsautomaten des ISA 88 ODM

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX ISA88: <http://www.hsu-ifa.de/ontologies/ISA-TR88#>
```

```
INSERT {

# Declaration of individuals for states
  ?Aborting rdf:type ISA88:Aborting;
      a owl:NamedIndividual.
  ?Clearing rdf:type ISA88:Clearing;
      a owl:NamedIndividual.
  ?Complete rdf:type ISA88:Complete;
      a owl:NamedIndividual.
  ?Completing rdf:type ISA88:Completing;
      a owl:NamedIndividual.
  ?Execute rdf:type ISA88:Execute;
      a owl:NamedIndividual.
  ?Holding rdf:type ISA88:Holding;
      a owl:NamedIndividual.
  ?Resetting rdf:type ISA88:Resetting;
      a owl:NamedIndividual.
  ?Starting rdf:type ISA88:Starting;
      a owl:NamedIndividual.
  ?Stopping rdf:type ISA88:Stopping;
      a owl:NamedIndividual.
  ?Suspending rdf:type ISA88:Suspending;
      a owl:NamedIndividual.
  ?Unholding rdf:type ISA88:Unholding;
      a owl:NamedIndividual.
  ?Unsuspending rdf:type ISA88:Unsuspending;
      a owl:NamedIndividual.
  ?Aborted rdf:type ISA88:Aborted;
      a owl:NamedIndividual.
  ?Held rdf:type ISA88:Held;
      a owl:NamedIndividual.
  ?Idle rdf:type ISA88:Idle;
      a owl:NamedIndividual.
  ?Stopped rdf:type ISA88:Stopped;
      a owl:NamedIndividual.
  ?Suspended rdf:type ISA88:Suspended;
      a owl:NamedIndividual.


# Declaration of individuals for transitions
  ?Abort_Command rdf:type ISA88:Abort_Command;
      a owl:NamedIndividual.
  ?Clear_Command rdf:type ISA88:Clear_Command;
      a owl:NamedIndividual.
  ?Hold_Command rdf:type ISA88:Hold_Command;
      a owl:NamedIndividual.
  ?Reset_Command rdf:type ISA88:Reset_Command;
      a owl:NamedIndividual.
  ?Start_Command rdf:type ISA88:Start_Command;
```

```
      a owl:NamedIndividual.
  ?Stop_Command rdf:type ISA88:Stop_Command;
      a owl:NamedIndividual.
  ?Suspend_Command rdf:type ISA88:Suspend_Command;
      a owl:NamedIndividual.
  ?Un_Hold_Command rdf:type ISA88:Un-Hold_Command;
      a owl:NamedIndividual.
  ?Unsuspend_Command rdf:type ISA88:Unsuspend_Command;
      a owl:NamedIndividual.
  ?Aborting_State_Complete rdf:type ISA88:Aborting_State_Complete;
      a owl:NamedIndividual.
  ?Clearing_State_Complete rdf:type ISA88:Clearing_State_Complete;
      a owl:NamedIndividual.
  ?Completing_State_Complete rdf:type ISA88:Completing_State_Complete;
      a owl:NamedIndividual.
  ?Execute_State_Complete rdf:type ISA88:Execute_State_Complete;
      a owl:NamedIndividual.
  ?Holding_State_Complete rdf:type ISA88:Holding_State_Complete;
      a owl:NamedIndividual.
  ?Resetting_State_Complete rdf:type ISA88:Resetting_State_Complete;
      a owl:NamedIndividual.
  ?Starting_State_Complete rdf:type ISA88:Starting_State_Complete;
      a owl:NamedIndividual.
  ?Stopping_State_Complete rdf:type ISA88:Stopping_State_Complete;
      a owl:NamedIndividual.
  ?Suspending_State_Complete rdf:type ISA88:Suspending_State_Complete;
      a owl:NamedIndividual.
  ?Unholding_State_Complete rdf:type ISA88:Unholding_State_Complete;
      a owl:NamedIndividual.
  ?Unsuspending_State_Complete rdf:type ISA88:Unsuspending_State_Complete;
      a owl:NamedIndividual.

# Declaration of ObjectProperties for states
      #state to transition
      ?Aborting ISA88:State_is_connected_with_Transition ?
          Aborting_State_Complete.
      ?Clearing ISA88:State_is_connected_with_Transition ?Abort_Command;
          ISA88:State_is_connected_with_Transition ?Clearing_State_Complete.
      ?Complete ISA88:State_is_connected_with_Transition ?Abort_Command;
          ISA88:State_is_connected_with_Transition ?Reset_Command;
          ISA88:State_is_connected_with_Transition ?Stop_Command.
      ?Completing ISA88:State_is_connected_with_Transition ?Abort_Command;
          ISA88:State_is_connected_with_Transition ?Completing_State_Complete;
          ISA88:State_is_connected_with_Transition ?Stop_Command.
      ?Execute ISA88:State_is_connected_with_Transition ?Abort_Command;
          ISA88:State_is_connected_with_Transition ?Stop_Command;
          ISA88:State_is_connected_with_Transition ?Execute_State_Complete;
          ISA88:State_is_connected_with_Transition ?Hold_Command;
          ISA88:State_is_connected_with_Transition ?Suspend_Command.
```

```
    ?Holding ISA88:State_is_connected_with_Transition ?Abort_Command;
        ISA88:State_is_connected_with_Transition ?Stop_Command;
        ISA88:State_is_connected_with_Transition ?Holding_State_Complete.
    ?Resetting ISA88:State_is_connected_with_Transition ?Abort_Command;
        ISA88:State_is_connected_with_Transition ?Stop_Command;
        ISA88:State_is_connected_with_Transition ?Resetting_State_Complete.
    ?Starting ISA88:State_is_connected_with_Transition ?Abort_Command;
        ISA88:State_is_connected_with_Transition ?Stop_Command;
        ISA88:State_is_connected_with_Transition ?Starting_State_Complete.
    ?Stopping ISA88:State_is_connected_with_Transition ?Abort_Command;
        ISA88:State_is_connected_with_Transition ?Stopping_State_Complete.
    ?Suspending ISA88:State_is_connected_with_Transition ?Abort_Command;
        ISA88:State_is_connected_with_Transition ?Stop_Command;
        ISA88:State_is_connected_with_Transition ?Suspending_State_Complete.
    ?Unholding ISA88:State_is_connected_with_Transition ?Abort_Command;
        ISA88:State_is_connected_with_Transition ?Stop_Command;
        ISA88:State_is_connected_with_Transition ?Unholding_State_Complete.
    ?Unsuspending ISA88:State_is_connected_with_Transition ?Abort_Command;
        ISA88:State_is_connected_with_Transition ?Stop_Command;
        ISA88:State_is_connected_with_Transition ?Unsuspending_State_Complete.
    ?Aborted ISA88:State_is_connected_with_Transition ?Clear_Command.
    ?Held ISA88:State_is_connected_with_Transition ?Abort_Command;
        ISA88:State_is_connected_with_Transition ?Stop_Command;
        ISA88:State_is_connected_with_Transition ?Un_Hold_Command.
    ?Idle ISA88:State_is_connected_with_Transition ?Abort_Command;
        ISA88:State_is_connected_with_Transition ?Stop_Command;
        ISA88:State_is_connected_with_Transition ?Start_Command.
    ?Stopped ISA88:State_is_connected_with_Transition ?Abort_Command;
        ISA88:State_is_connected_with_Transition ?Reset_Command.
    ?Suspended ISA88:State_is_connected_with_Transition ?Abort_Command;
        ISA88:State_is_connected_with_Transition ?Stop_Command;
        ISA88:State_is_connected_with_Transition ?Unsuspend_Command.

    # transition to state
    ?Abort_Command ISA88:Transition_is_connected_with_State ?Aborting.
    ?Clear_Command ISA88:Transition_is_connected_with_State ?Clearing.
    ?Hold_Command ISA88:Transition_is_connected_with_State ?Holding.
    ?Reset_Command ISA88:Transition_is_connected_with_State ?Resetting.
    ?Start_Command ISA88:Transition_is_connected_with_State ?Starting.
    ?Stop_Command ISA88:Transition_is_connected_with_State ?Stopping.
    ?Suspend_Command ISA88:Transition_is_connected_with_State ?Suspending.
    ?Un_Hold_Command ISA88:Transition_is_connected_with_State ?Unholding.
    ?Unsuspend_Command ISA88:Transition_is_connected_with_State ?Unsuspending.
    ?Aborting_State_Complete ISA88:Transition_is_connected_with_State ?Aborted
        .

    ?Aborting_State_Complete ISA88:Transition_is_connected_with_State ?Aborted
        .
    ?Clearing_State_Complete ISA88:Transition_is_connected_with_State ?Stopped
```

```
    .
?Completing_State_Complete ISA88:Transition_is_connected_with_State ?
    Complete.
?Execute_State_Complete ISA88:Transition_is_connected_with_State ?
    Completing.
?Holding_State_Complete ISA88:Transition_is_connected_with_State ?Held.
?Resetting_State_Complete ISA88:Transition_is_connected_with_State ?Idle.
?Starting_State_Complete ISA88:Transition_is_connected_with_State ?Execute
    .
?Stopping_State_Complete ISA88:Transition_is_connected_with_State ?Stopped
    .
?Suspending_State_Complete ISA88:Transition_is_connected_with_State ?
    Suspended.
?Unholding_State_Complete ISA88:Transition_is_connected_with_State ?
    Execute.
?Unsuspending_State_Complete ISA88:Transition_is_connected_with_State ?
    Execute.

# defines the element to bind the states and transitions to
?DescribedElement
#connected to states
ISA88:described_by_ISA88_Element ?Aborting;
ISA88:described_by_ISA88_Element ?Clearing;
ISA88:described_by_ISA88_Element ?Complete;
ISA88:described_by_ISA88_Element ?Completing;
ISA88:described_by_ISA88_Element ?Execute;
ISA88:described_by_ISA88_Element ?Holding;
ISA88:described_by_ISA88_Element ?Resetting;
ISA88:described_by_ISA88_Element ?Starting;
ISA88:described_by_ISA88_Element ?Stopping;
ISA88:described_by_ISA88_Element ?Suspending;
ISA88:described_by_ISA88_Element ?Unholding;
ISA88:described_by_ISA88_Element ?Unsuspending;
ISA88:described_by_ISA88_Element ?Aborted;
ISA88:described_by_ISA88_Element ?Held;
ISA88:described_by_ISA88_Element ?Idle;
ISA88:described_by_ISA88_Element ?Stopped;
ISA88:described_by_ISA88_Element ?Suspended;
# connected to transitions
ISA88:described_by_ISA88_Element ?Abort_Command;
ISA88:described_by_ISA88_Element ?Clear_Command;
ISA88:described_by_ISA88_Element ?Hold_Command;
ISA88:described_by_ISA88_Element ?Reset_Command;
ISA88:described_by_ISA88_Element ?Start_Command;
ISA88:described_by_ISA88_Element ?Stop_Command;
ISA88:described_by_ISA88_Element ?Suspend_Command;
ISA88:described_by_ISA88_Element ?Un_Hold_Command;
ISA88:described_by_ISA88_Element ?Aborting_State_Complete;
ISA88:described_by_ISA88_Element ?Clearing_State_Complete;
```

```
      ISA88:described_by_ISA88_Element ?Completing_State_Complete;
      ISA88:described_by_ISA88_Element ?Execute_State_Complete;
      ISA88:described_by_ISA88_Element ?Holding_State_Complete;
      ISA88:described_by_ISA88_Element ?Resetting_State_Complete;
      ISA88:described_by_ISA88_Element ?Starting_State_Complete;
      ISA88:described_by_ISA88_Element ?Stopping_State_Complete;
      ISA88:described_by_ISA88_Element ?Suspending_State_Complete;
      ISA88:described_by_ISA88_Element ?Unholding_State_Complete;
      ISA88:described_by_ISA88_Element ?Unsuspending_State_Complete;
      ISA88:described_by_ISA88_Element ?Unsuspend_Command.
} WHERE {
      # ------------------------------------------------------------------ #
      BIND(STR("ExampleSystemIRI") AS ?SystemType).
      BIND(STR("ExampleDescribedElement") AS ?DescribedElement).
      # ------------------------------------------------------------------ #
      # Defines the general namespace for all individuals
      BIND(STR("ExampleNamespace") AS ?NameSpace).
      # ------------------------------------------------------------------ #
      # states
      BIND(IRI(CONCAT(?NameSpace,?SystemType,"_Aborting")) AS ?Aborting).
      BIND(IRI(CONCAT(?NameSpace,?SystemType,"_Clearing")) AS ?Clearing).
      BIND(IRI(CONCAT(?NameSpace,?SystemType,"_Complete")) AS ?Complete).
      BIND(IRI(CONCAT(?NameSpace,?SystemType,"_Completing")) AS ?Completing).
      BIND(IRI(CONCAT(?NameSpace,?SystemType,"_Execute")) AS ?Execute).
      BIND(IRI(CONCAT(?NameSpace,?SystemType,"_Holding")) AS ?Holding).
      BIND(IRI(CONCAT(?NameSpace,?SystemType,"_Resetting")) AS ?Resetting).
      BIND(IRI(CONCAT(?NameSpace,?SystemType,"_Starting")) AS ?Starting).
      BIND(IRI(CONCAT(?NameSpace,?SystemType,"_Stopping")) AS ?Stopping).
      BIND(IRI(CONCAT(?NameSpace,?SystemType,"_Suspending")) AS ?Suspending).
      BIND(IRI(CONCAT(?NameSpace,?SystemType,"_Unholding")) AS ?Unholding).
      BIND(IRI(CONCAT(?NameSpace,?SystemType,"_Unsuspending")) AS ?Unsuspending)
          .
      BIND(IRI(CONCAT(?NameSpace,?SystemType,"_Aborted")) AS ?Aborted).
      BIND(IRI(CONCAT(?NameSpace,?SystemType,"_Held")) AS ?Held).
      BIND(IRI(CONCAT(?NameSpace,?SystemType,"_Idle")) AS ?Idle).
      BIND(IRI(CONCAT(?NameSpace,?SystemType,"_Stopped")) AS ?Stopped).
      BIND(IRI(CONCAT(?NameSpace,?SystemType,"_Suspended")) AS ?Suspended).
      # transitions
      BIND(IRI(CONCAT(?NameSpace,?SystemType,"_Abort_Command")) AS ?
          Abort_Command).
      BIND(IRI(CONCAT(?NameSpace,?SystemType,"_Clear_Command")) AS ?
          Clear_Command).
      BIND(IRI(CONCAT(?NameSpace,?SystemType,"_Hold_Command")) AS ?Hold_Command)
          .
      BIND(IRI(CONCAT(?NameSpace,?SystemType,"_Reset_Command")) AS ?
          Reset_Command).
      BIND(IRI(CONCAT(?NameSpace,?SystemType,"_Start_Command")) AS ?
          Start_Command).
      BIND(IRI(CONCAT(?NameSpace,?SystemType,"_Stop_Command")) AS ?Stop_Command)
```

```
        .
    BIND(IRI(CONCAT(?NameSpace,?SystemType,"_Suspend_Command")) AS ?
        Suspend_Command).
    BIND(IRI(CONCAT(?NameSpace,?SystemType,"_Un_Hold_Command")) AS ?
        Un_Hold_Command).
    BIND(IRI(CONCAT(?NameSpace,?SystemType,"_Unsuspend_Command")) AS ?
        Unsuspend_Command).
    BIND(IRI(CONCAT(?NameSpace,?SystemType,"_Aborting_State_Complete")) AS ?
        Aborting_State_Complete).
    BIND(IRI(CONCAT(?NameSpace,?SystemType,"_Clearing_State_Complete")) AS ?
        Clearing_State_Complete).
    BIND(IRI(CONCAT(?NameSpace,?SystemType,"_Completing_State_Complete")) AS ?
        Completing_State_Complete).
    BIND(IRI(CONCAT(?NameSpace,?SystemType,"_Execute_State_Complete")) AS ?
        Execute_State_Complete).
    BIND(IRI(CONCAT(?NameSpace,?SystemType,"_Holding_State_Complete")) AS ?
        Holding_State_Complete).
    BIND(IRI(CONCAT(?NameSpace,?SystemType,"_Resetting_State_Complete")) AS ?
        Resetting_State_Complete).
    BIND(IRI(CONCAT(?NameSpace,?SystemType,"_Starting_State_Complete")) AS ?
        Starting_State_Complete).
    BIND(IRI(CONCAT(?NameSpace,?SystemType,"_Stopping_State_Complete")) AS ?
        Stopping_State_Complete).
    BIND(IRI(CONCAT(?NameSpace,?SystemType,"_Suspending_State_Complete")) AS ?
        Suspending_State_Complete).
    BIND(IRI(CONCAT(?NameSpace,?SystemType,"_Unholding_State_Complete")) AS ?
        Unholding_State_Complete).
    BIND(IRI(CONCAT(?NameSpace,?SystemType,"_Unsuspending_State_Complete")) AS
         ?Unsuspending_State_Complete).
}
```

Listing C.4: Assertionales Muster für eine Systemstruktur entsprechend des VDI 2206 ODM

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX VDI2206: <http://www.hsu-ifa.de/ontologies/VDI2206#>
INSERT {
    ?System a VDI2206:System.
    ?System VDI2206:SystemConsistsOfSystem ?SubSystem.
    ?SubSystem a VDI2206:System.
    ?SubSystem VDI2206:SystemConsistsOfModule ?Module.
    ?Module a VDI2206:Module.
    ?Module VDI2206:ModuleConsistsOfComponent ?Component.
    ?Component a VDI2206:Component.
} WHERE {
    BIND(IRI(STR("http://ExampleSystem")) AS ?System).
    BIND(IRI(STR("http://ExampleSubSystem")) AS ?SubSystem).
    BIND(IRI(STR("http://ExampleModule")) AS ?Module).
```

```
    BIND(IRI(STR("http://ExampleComponent")) AS ?Component).
}
```

Listing C.5: Assertionales Muster für ein mechatronisches System (Modul) entsprechend des VDI2206 ODM

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX VDI2206: <http://www.hsu-ifa.de/ontologies/VDI2206#>
INSERT {

    ?Sensor a VDI2206:Sensor.
    ?Actor a VDI2206:Actor.
    ?InformationProcessing a VDI2206:InformationProcessing.
    ?ActorSignal a VDI2206:Information.
    ?SensorSignal a VDI2206:Information.

    ?Module VDI2206:ModuleConsistsOfComponent ?Sensor.
    ?Module VDI2206:ModuleConsistsOfComponent ?Actor.
    ?Module VDI2206:ModuleConsistsOfComponent ?InformationProcessing.
    ?InformationProcessing VDI2206:hasInput ?SensorSignal.
    ?InformationProcessing VDI2206:hasOutput ?ActorSignal.
    ?Actor VDI2206:hasInput ?ActorSignal.
    ?Actor VDI2206:hasOutput ?ActorOutput.
    ?Sensor VDI2206:hasInput ?SensorInput.
    ?Sensor VDI2206:hasOutput ?SensorSignal.

    ?BasicSystem a VDI2206:BasicSystem.
    ?BasicSystem VDI2206:hasInput ?InputProduct.
    ?BasicSystem VDI2206:hasInput ?ActorOutput.
    ?BasicSystem VDI2206:hasOutput ?SensorInput.
    ?BasicSystem VDI2206:hasOutput ?OutputProduct.
    ?InputProduct a VDI2206:Product.
    ?OutputProduct a VDI2206:Product.
    ?SensorInput a VDI2206:Energy.
    ?ActorOutput a VDI2206:Energy.

} WHERE {
    BIND(IRI(STR("http://ExampleModule")) AS ?Module).
    BIND(IRI(STR("http://ExampleInformationProcessing")) AS ?
        InformationProcessing).
    BIND(IRI(STR("http://ExampleSensor")) AS ?Sensor).
    BIND(IRI(STR("http://ExampleSensorInput")) AS ?SensorInput).
    BIND(IRI(STR("http://ExampleSensorSignal")) AS ?SensorSignal).
    BIND(IRI(STR("http://ExampleActor")) AS ?Actor).
    BIND(IRI(STR("http://ExampleActorSignal")) AS ?ActorSignal).
    BIND(IRI(STR("http://ExampleActorOutput")) AS ?ActorOutput).
    BIND(IRI(STR("http://ExampleBasicSystem")) AS ?BasicSystem).
```

```
    BIND(IRI(STR("http://ExampleInputProduct")) AS ?InputProduct).
    BIND(IRI(STR("http://ExampleOutputProduct")) AS ?OutputProduct).
}
```

Listing C.6: Assertionales Muster für eine Typbeschreibung des DIN EN 61360 ODM

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX DE6: <http://www.hsu-ifa.de/ontologies/DINEN61360#>

INSERT {
    # Declaration of individuals
      # Definition of DE Type Description
      ?DE_TypeIRI rdf:type DE6:Type_Description;
      a owl:NamedIndividual;
      rdf:type ?DataTypeIRI;
      rdfs:label ?DET_Label.

    # Declaration of DataProperties for the DE Type Description
      #non optional
      ?DE_TypeIRI DE6:Code ?Code;
      DE6:Version_Number ?Version_Number;
      DE6:Revision_Number ?Revision_Number;
      DE6:Preferred_Name ?Preferred_Name;
      DE6:Short_Name ?Short_Name;
      DE6:Definition ?Definition;

        #optional
      DE6:Synonymous_Name ?Synonymous_Name;
      DE6:backwards_compatible_Revision ?backwards_compatible_Revision;
      DE6:backwards_compatible_Version ?backwards_compatible_Version;
      DE6:Value_Format_Field_length ?Value_Format_Field_length;
      DE6:Value_Format_Field_length_Variable ?Value_Format_Field_length_Variable
          ;
      DE6:Value_Format_non-quantitativ ?Value_Format_non_quantitativ;
      DE6:Value_Format_quantitativ ?Value_Format_quantitativ;
      DE6:Value_List ?Value_List;
      DE6:Value_List_Member ?Value_List_Member;
      DE6:Source_Document_of_Definition ?Source_Document_of_Definition;
      DE6:Synonymous_Letter_Symbol ?Synonymous_Letter_Symbol;
      DE6:Note ?Note;
      DE6:Remark ?Remark;
      DE6:Preferred_Letter_Symbol ?Preferred_Letter_Symbol;
      DE6:Formula ?Formula;
      DE6:Drawing_Reference ?Drawing_Reference;
```

```
        #case-dependent
      DE6:Unit_of_Measure ?Unit_of_Measure.

} WHERE {
      #non optional
      BIND(STR("ExampleCode") AS ?Code).
      BIND(STR("ExampleVersion") AS ?Version_Number).
      BIND(STR("ExampleRevision") AS ?Revision_Number).
      BIND(STR("ExamplePreferredName") AS ?Preferred_Name).
      BIND(STR("ExampleShortName") AS ?Short_Name).
      BIND(STR("ExampleDefinition") AS ?Definition).
      #optional
      BIND(STR("ExampleSynonymous_Name") AS ?Synonymous_Name).
      BIND(STR("ExampleBackwards_compatible_Revision") AS ?
         backwards_compatible_Revision).
      BIND(STR("ExampleBackwards_compatible_Version") AS ?
         backwards_compatible_Version).
      BIND(STR("ExampleValue_Format_Field_length") AS ?Value_Format_Field_length
         ).
      BIND(STR("ExampleValue_Format_Field_length_Variable") AS ?
         Value_Format_Field_length_Variable).
      BIND(STR("ExampleValue_Format_non_quantitativ") AS ?
         Value_Format_non_quantitativ).
      BIND(STR("ExampleValue_Format_quantitativ") AS ?Value_Format_quantitativ).
      BIND(STR("ExampleValue_List") AS ?Value_List).
      BIND(STR("ExampleValue_List_Member") AS ?Value_List_Member).
      BIND(STR("ExampleSource_Document_of_Definition") AS ?
         Source_Document_of_Definition).
      BIND(STR("ExampleSynonymous_Letter_Symbol") AS ?Synonymous_Letter_Symbol).
      BIND(STR("ExampleNote") AS ?Note).
      BIND(STR("ExampleRemark") AS ?Remark).
      BIND(STR("ExamplePreferred_Letter_Symbol") AS ?Preferred_Letter_Symbol).
      BIND(STR("ExampleFormula") AS ?Formula).
      BIND(STR("ExampleDrawing_Reference") AS ?Drawing_Reference).
      BIND(STR("ExampleUnitOfMeasure") AS ?Unit_of_Measure).

      # Defines the general namespace for all individuals
      BIND(?Preferred_Name AS ?DET_Label).
      BIND(<ExampleTypeIRI> AS ?DE_TypeIRI).
      BIND(<ExampleDataTypeIRI> AS ?DataTypeIRI).
}
```

Listing C.7: Assertionales Muster für eine Instazbeschreibung des DIN EN 61360 ODM

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX DE6: <http://www.hsu-ifa.de/ontologies/DINEN61360#>
```

```
INSERT {
    # Declaration of individuals
    ?DE_uuid rdf:type DE6:Data_Element;
    a owl:NamedIndividual;
    rdfs:label ?DE_Label.
    ?DE_Instance_uuid rdf:type DE6:Instance_Description;
    a owl:NamedIndividual;
    rdfs:label ?DEI_Label.

    # Declaration of ObjectProperties
    ?DE_uuid DE6:has_Instance_Description ?DE_Instance_uuid;
    DE6:has_Type_Description ?DE_TypeIRI.
    ?DE_TypeIRI DE6:Type_Description_has_Instance ?DE_Instance_uuid.

    # Defines the individual to connect the data element to
    ?DescribedIndividualIRI DE6:has_Data_Element ?DE_uuid.

    # Declaration of DataProperties for the DEs Instance description
    ?DE_Instance_uuid DE6:Expression_Goal ?Expression_Goal;
    DE6:Logic_Interpretation ?Logic_Interpretation;
    DE6:Value ?Value;
    DE6:Created_at_Date ?Created_at_Date.

} WHERE {

      BIND(<ExampleDescribedIndividualIRI> AS ?DescribedIndividualIRI).

      BIND(STR("ExpressionGoal") AS ?Expression_Goal).
      BIND(STR("LogicInterpretation") AS ?Logic_Interpretation).

      BIND(STR("ExampleValue") AS ?Value).

      BIND(STR("ExampleCode") AS ?Code).
      ?DE_TypeIRI DE6:Code ?Code;
      rdfs:label ?DET_Label.

      BIND(CONCAT(?DET_Label,"_Data_Element") AS ?DE_Label).
      BIND(CONCAT(?DET_Label,"_Instance") AS ?DEI_Label).
      BIND(NOW() AS ?Created_at_Date).
      BIND(UUID() AS ?DE_uuid).
      BIND(UUID() AS ?DE_Instance_uuid).
}
```

Listing C.8: SPARQL-Abfrage zur Erfüllung der Anforderung 1 aus Fallstudie WA

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX DE6: <http://www.hsu-ifa.de/ontologies/DINEN61360#>
```

```
PREFIX a: <http://a_Knowledge_Base#>

# ID = 1: Was ist der aktuelle Status eines LRS "Y"?

SELECT ?LRS_Instance ?CurrentStatus ?DataElementDefinition WHERE {
    # Definiere bestimmtes LRS
    BIND(IRI("http://a_Knowledge_Base#LRS-1") AS ?LRS_Instance).

    # Suche den Status des LRS
    ?LRS_Instance DE6:has_Data_Element ?DataElement.
    ?DataElement DE6:has_Type_Description ?TypeDescription;
        DE6:has_Instance_Description ?InstanceDescription.

    # Suche die Instanz mit dem Typ "Status", identifiziert durch den Code
    ?TypeDescription DE6:Code "DE-Code-DE-XUD145";
            DE6:Version_Number "1.0";
            DE6:Definition ?DataElementDefinition.
    ?InstanceDescription DE6:has_Data_Type ?DataType;
                        DE6:Expression_Goal 'Actual_Value'.

    ?DataType DE6:Value ?CurrentStatus.

}
```

Tabelle C.5.: Exemplarische Rückgabe zu Listing C.8

?LRS_Instance	?DataElementDefinition	?Value
http://a_Knowledge_Base#LRS-1	E1	OK

Listing C.9: SPARQL-Abfrage zur Erfüllung der Anforderung 3 aus Fallstudie WA

```
PREFIX VDI2206: <http://www.hsu-ifa.de/ontologies/VDI2206#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX a: <http://a_Knowledge_Base#>

# ID = 3: Welche LRS existieren in Maschine "M"

SELECT ?Machine ?LRS_Instance ?LRS_Label
{
    # Definition der Maschine "M"
    BIND(IRI("http://a_Knowledge_Base#DMG-40-1234") AS ?Machine).

    # Finde alle LRS aus denen die Maschine besteht und suche auch das Label
    ?Machine a VDI2206:System.
    ?Machine VDI2206:consistsOf ?LRS_Instance.
    ?LRS_Instance rdfs:label ?LRS_Label.

}
```

Tabelle C.6.: Exemplarische Rückgabe zu Listing C.9

?Machine	?LRS_Instance	?LRS_Label
http://a_Knowledge_Base#DMG-40-1234	http://a_Knowledge_Base#LRS-1	LRS-Typ A X-Achse
http://a_Knowledge_Base#DMG-40-1234	http://a_Knowledge_Base#LRS-2	LRS-Typ A Y-Achse
http://a_Knowledge_Base#DMG-40-1234	http://a_Knowledge_Base#LRS-3	LRS-Typ A Z-Achse

Listing C.10: SPARQL-Abfrage zur Erfüllung der Anforderung 1 aus Fallstudie AF

```
PREFIX VDI3682: <http://www.hsu-ifa.de/ontologies/VDI3682#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX VDI2206: <http://www.hsu-ifa.de/ontologies/VDI2206#>
PREFIX Cap: <http://www.hsu-ifa.de/ontologies/capability-model#>

# ID = 3: Welche Capabilities besitzt Modul "X"?

SELECT ?capability ?input ?output WHERE {
    # Finde Faehigkeit von Modul "x"
    BIND(IRI("http://www.hsu-ifa.de/ontologies/capability-model#X") AS ?module).
    ?module a VDI2206:Module.
    ?module Cap:hasCapability ?capability.
    ?capability a Cap:Capability.

    # Finde optional noch die Inputs
    OPTIONAL{
        ?capability VDI3682:hasInput ?input.
        ?input a ?fpbElement.
        VALUES ?fpbElement {VDI3682:Energy VDI3682:Product VDI3682:Information}
    }

    # Finde optional noch die Outputs
    OPTIONAL{
        ?capability VDI3682:hasOutput ?output.
        ?output a ?fpbElement.
        VALUES ?fpbElement {VDI3682:Energy VDI3682:Product VDI3682:Information}
    }
}
```

Listing C.11: SPARQL-Abfrage zur Erfüllung der Anforderung 1 aus Fallstudie PA-1&2

```
PREFIX VDI2206: <http://www.hsu-ifa.de/ontologies/VDI2206#>
PREFIX C: <http://C_Knowledge_Base#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
```

Tabelle C.7.: Exemplarische Rückgabe zu Listing C.10

?capability	?input	?output
b:Bohren-1	b:ZylinderD12	b:Lochzylinderk100
b:Bohren-2	b:ZylinderD4	b:Lochzylinderk200
b:Bohren-3	b:ZylinderD8	b:Lochzylinderk900

PREFIX b: http://b_Knowledge_Base#

```
PREFIX DINEN61360: <http://www.hsu-ifa.de/ontologies/DINEN61360#>

# ID = 1: Welche Module besitzt die Anlage "T-2-000"?

SELECT ?System ?SystemLabel ?Module ?ModuleLabel ?VRLink WHERE {
    # Definiere das betrachtete System
    BIND(IRI("http://C_Knowledge_Base#T-2-000") AS ?System)

    # Definiere, das es sich um ein System handelt, welches aus Modulen besteht
    ?System a VDI2206:System;
        rdfs:label ?SystemLabel;
        VDI2206:consistsOf ?Module.

    # Finde die zu den Modulen gehoerigen Virtual-Reality-Links
    ?Module DINEN61360:has_Data_Element ?VRLink-DE;
        rdfs:label ?ModuleLabel.

    ?VRLink-DE DINEN61360:has_Type_Description C:VRLink_TD;
        DINEN61360:has_Instance_Description ?VRLink-ID.

    ?VRLink-ID DINEN61360:Value ?VRLink.
}
```

Tabelle C.8.: Exemplarische Rückgabe zu Listing C.11

?System	?SystemLabel	?Module	?ModuleLabel	?VRLink
c:T-2-000	Preforminganlage	c:T-2-001	Uebergabetisch	https://matterport.com/show...
c:T-2-000	Preforminganlage	c:T-2-002	Bilderkennung	https://matterport.com/show...
c:T-2-000	Preforminganlage	c:T-2-004	Rollenfoerderer	https://matterport.com/show...
c:T-2-000	Preforminganlage	c:T-2-100	Rob_01	https://matterport.com/show...

PREFIX c: http://c_Knowledge_Base#

Listing C.12: SPARQL-Abfrage zur Erfüllung der Anforderung 2 aus Fallstudie PA-1&2

```
PREFIX VDI: <http://www.hsu-ifa.de/ontologies/VDI3682#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX DIN: <http://www.hsu-ifa.de/ontologies/DINEN61360#>

# ID = 2: Welche Prozessschritte und Ressourcen sind zur Fertigung von
    Produkttyp "X" notwendig?
```

```
SELECT DISTINCT ?ProductID ?ProcessID ?ProcessLabel ?ResourceLabel ?VRLink WHERE
     {
    # Definiere das betrachtete Produkt
    BIND(IRI("http://C_Knowledge_Base#P_005800a") AS ?ProductID)

    # Finde alle Individuen die Vorgaenger sind und nehme jene, welche Prozesse
        sind
    ?ProductID a VDI:Product;
        VDI:upFlow* ?ProcessID.

    ?ProcessID a VDI:Process;
        rdfs:label ?ProcessLabel.

    # Finde die dazugehoerige Technische Ressource
    ?ProcessID VDI:ProcessOperatorIsAssignedToTechnicalResource ?ResourceID.
    ?ResourceID rdfs:label ?ResourceLabel.

    # Ergaenze optional einen Virtual-Reality-Link
    OPTIONAL {
        ?ResourceID DIN:has_Data_Element ?VRLink_DE.
        ?VRLink_DE DIN:has_Instance_Description ?VRLink_DEI;
            DIN:has_Type_Description ?VRLink_DET.
        ?VRLink_DET DIN:Code "ASDQWE154".
        ?VRLink_DEI DIN:Value ?VRLink.
    }
}
```

Tabelle C.9.: Exemplarische Rückgabe zu Listing C.12

?ProductID	?ProcessID	?ProcessLabel	?ResourceLabel	?VRLink
c:P_005800a	c:O_331	Entformen durch Roboter	Rob_02	https://matterport.com/show...
c:P_005800a	c:O_331	Entformen durch Roboter	Aufgabeplatz	https://matterport.com/show...
c:P_005800a	c:O_321	Tempern im Ofen	Temperofen	https://matterport.com/show...
c:P_005800a	c:O_311	1K Injektion	RTM-Presse	https://matterport.com/show...
c:P_005800a	c:O_301	Konturscan	LLS-Sensor	https://matterport.com/show...
c:P_005800a	c:O_201	Wareneingang	Konsolidierkasette	https://matterport.com/show...

PREFIX c: http://c_Knowledge_Base#

Listing C.13: SPARQL-Abfrage zur Erfüllung der Anforderung 3 aus Fallstudie PA-1&2

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX VDI: <http://www.hsu-ifa.de/ontologies/VDI3682#>
PREFIX DE6: <http://www.hsu-ifa.de/ontologies/DINEN61360#>
```

```
# ID = 3: Welche Prozesswerte muessen fuer die Ausfuehrung von Prozess "X"
    definiert werden?

SELECT DISTINCT ?DataElementType ?Code ?Name ?Unit WHERE {
    # Definiere den betrachteten Prozess
    BIND(IRI("http://C_Knowledge_Base#ON456789_O_2-21") AS ?Process)

    # Finde alle am Prozess haengenden Typbeschreibungen von Merkmalen
    ?Process DE6:has_Data_Element ?DataElement.
    ?DataElement DE6:complex_Data_Type_has_Member ?DataElementInstance.
    ?DataElementInstance DE6:Instance_Description_has_Type ?DataElementType.

    ?DataElementType DE6:Code ?Code.
    ?DataElementType DE6:Preferred_Name ?Name.
    ?DataElementType DE6:Unit_of_Measure ?Unit.

}
```

Tabelle C.10.: Exemplarische Rückgabe zu Listing C.13

?DataElementType	?Code	?Name	?Unit
c:E_224_TD	E_224_TD	Vacuum	mbar
c:E_226_TD	E_226_TD	TemperatureHC1	∘C
c:E_225_TD	E_225_TD	ProcessTime	s
c:E_223_TD	E_223_TD	CurrentProgram	n/a
c:E_229_TD	E_229_TD	Pressure	bar

PREFIX c: http://c_Knowledge_Base#

Literaturverzeichnis

Dieses Verzeichnis enthält die im Rahmen der Arbeit referenzierte Literatur, wobei die Literatur in fünf Abschnitte unterteilt wird:

1. Wissenschaftliche Publikationen: Wissenschaftliche Publikationen, die nicht vom Autor der vorliegenden Arbeit stammen.

2. Normen und Richtlinien: Normen und Richtlinien, die im Allgemeinen nicht als wissenschaftliche Literatur gelten, bspw. DIN oder ISO Standards.

3. Publikationen des Autors: Wissenschaftliche Publikationen, welche vom Autor der vorliegenden Arbeit als Erst- oder Co-Autor verfasst wurden. Publikationen des Autors sind durch ein hochgestelltes Asteriskus-Zeichen (*) gekennzeichnet.

4. Studentische Arbeiten: Studentische Arbeiten, die vom Autor der vorliegenden Arbeit betreut wurden und deren Erkenntnisse in das Ergebnis der Arbeit einflossen. Kurzverweise studentischer Arbeiten sind durch ein hochgestelltes Prozentzeichen (%) gekennzeichnet.

5. Internetquellen und Software: Quellen aus dem World-Wide-Web bzw. Software die durch eine URL referenziert werden. Kurzverweise von Internetquellen sind durch ein hochgestelltes At-Zeichen (@) gekennzeichnet.

Wissenschaftliche Publikationen

[ADH19] Abdelghany Abdelghany, Nagy Darwish und Hesham Hefni: „An Agile Methodology for Ontology Development“. In: "International Journal of Intelligent Engineering and Systems" 12.2, 2019, S. 170–181.

[Alm14] Ricardo de Almeida Falbo: „SABiO: Systematic approach for building ontologies“. In: "CEUR Workshop Proceedings". Bd. 1301. 2014.

[Ang+10] Ronald Angerbauer, Raphael Buck, Ulrich Doll, Monika Hackel, Karl-Heinz Kayser, Michael Klebl, Heiner Mack, Reinhold Siegler, Florian Wascher und Rainer Würslin: "aquimo: Ein Leitfaden für Maschinen- und Anlagenbauer - Adaptierbares Modellierungswerkzeug und Qualifizierungsprogramm für den Aufbau firmenspezifischer mechatronischer Engineeringprozesse". Frankfurt, M.: VDMA-Verlag, 2010.

[Aue+13] Sören Auer, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Claus Stadler und Jörg Unbehauen: „Extraktion, Mapping und Verlinkung von Daten im Web“. In: "Datenbank-Spektrum" 13.2, 2013, S. 77–87.

[AuHe07] Sören Auer und Heinrich Herre: „RapidOWL — An Agile Knowledge Engineering Methodology“. In: "Perspectives of Systems Informatics". Hrsg. von Irina Virbitskaite und Andrei Voronkov. Bd. 4378. Lecture notes in computer science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, S. 424–430.

[AVH16] Pekka Aarnio, Valeriy Vyatkin und David Hästbacka: „Context Modeling with Situation Rules for Industrial Maintenance“. In: "21st IEEE International Conference on Emerging Technologies and Factory Automation". 2016.

[AvZd05] Paris Avgeriou und Uwe Zdun: „Architectural patterns revisited – a pattern language“. In: "European Conference on Pattern Languages of Programs (EuroPlop)". 2005.

[Baa+05] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi und Peter F. Patel-Schneider: "The description logic handbook: Theory, implementation, and applications". Cambridge: Cambridge Univ. Press, 2005.

[BaKa15] Srividya K. Bansal und Sebastian Kagemann: „Integrating Big Data: A Semantic Extract-Transform-Load Framework“. In: "Computer" 48.3, 2015, S. 42–50.

[Bar+12] Mike Barth, Rainer Drath, Alexander Fay, Florian Zimmer und Karin Eckert: „Evaluation of the openness of automation tools for interoperability in engineering tool chains“. In: "17th IEEE Conference on Emerging Technologies & Factory Automation". 2012, S. 1–8.

[BHP16] Eva Blomqvist, Karl Hammar und Valentina Presutti: „Engineering Ontologies with Patterns - The eXtreme Design Methodology“. In: "Ontology engineering with ontology design patterns". Hrsg. von Aldo Gangemi, Adila Krisnadhi, Krzysztof Janowicz, Pascal Hitzler und Valentina Presutti. Studies on the semantic web. Amsterdam, Netherlands: IOS Press, 2016.

[BiSa16] Stefan Biffl und Marta Sabou, Hrsg.: "Semantic web technologies in intelligent engineering applications". Springer, 2016.

[Blf+17] Aisha Blfgeh, Jennifer Warrender, Catharien M. U. Hilkens und Phillip Lord: „A document-centric approach for developing the tolAPC ontology“. In: "Journal of biomedical semantics" 8.1, 2017, S. 54.

[Blo+18] Henry Bloch, Stephan Hensel, Mario Hoernicke, Katharina Stark, Anna Menschner, Alexander Fay, Leon Urbas, Torsten Knohl und Jens Bernshausen: „State-based control of process services within modular process plants“. In: "Procedia CIRP" 72, 2018, S. 1088–1093.

[Bra+07] Simone Braun, Andreas Schmidt, Andreas Walter, Gabor Nagypal und Valentin Zacharias: „Ontology Maturing: a Collaborative Web 2.0 Approach to Ontology Engineering“. In: "16th International World Wide Web Conference (WWW2007)". 2007.

[Bre+11] Brecher Christian, Christian Diedrich, Juejing Feng, Mehdi Hamadou, Werner Herfs, Andreas W. Müller, Denis Özdemir, Nico Suchold und Wolfgang Wohlgemuth: „Informationen im PLM-Prozess“. In: "Virtuelle Techniken im industriellen Umfeld". Hrsg. von Werner Schreiber und Peter Zimmermann. Springer-Verlag, 2011.

[Bro+06] Saartje Brockmans, Peter Haase, Pascal Hitzler und Rudi Studer: „A Metamodel and UML Profile for Rule-Extended OWL DL Ontologies“. In: "The Semantic Web: Research and Applications". Hrsg. von David Hutchison u. a. Bd. 4011. Lecture notes in computer science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, S. 303–316.

[Bur+05] Andrew Burton-Jones, Veda C. Storey, Vijayan Sugumaran und Punit Ahluwalia: „A semiotic metrics suite for assessing the quality of ontologies“. In: "Data & Knowledge Engineering" 55.1, 2005, S. 84–102.

[Cal+16] Diego Calvanese, Benjamin Cogrel, Sarah Komla-Ebri, Roman Kontchakov, Davide Lanti, Martin Rezk, Mariano Rodriguez-Muro und Guohui Xiao: „Ontop: Answering SPARQL queries over relational databases“. In: "Semantic Web" 8.3, 2016, S. 471–487.

[Cho96] C. W. Choo: „The knowing organization: How organizations use information to construct meaning, create knowledge and make decisions“. In: "International Journal of Information Management" 16.5, 1996, S. 329–340.

[Civ+13] Cristina Civili u. a.: „Mastro Studio: Managing Ontology-Based Data Access applications“. In: "Proceedings of the VLDB Endowment" 6.12, 2013, S. 1314–1317.

[Cop98] James O. Coplien: „Software design patterns: Common questions and answers“. In: "The patterns handbook: Techniques, strategies, and applications" 13, 1998,

[CuBl15] Olivier Curé und Guillaume Blin: "RDF Database Systems". Elsevier, 2015.

[Dau+16a] Marian Daun, Philipp Bohn, Jennifer Brings und Thorsten Weyer: „Structured Model-based Engineering of Long-living Embedded Systems: The SPES Methodological Building Blocks Framework“. In: "Softwaretechnik-Trends" 36.1, 2016,

[Dau+16b] Marian Daun, Jennifer Brings, Thorsten Weyer und Bastian Tenbergen: „Fostering concurrent engineering of cyber-physical systems a proposal for an ontological context framework“. In: "2016 3rd International Workshop on Emerging Ideas and Trends in Engineering of Cyber-Physical Systems (EITEC)". 2016, S. 5–10.

[Dau+16c] Marian Daun, Bastian Tenbergen, Jennifer Brings und Thorsten Weyer: „SPES XT Context Modeling Framework“. In: "Advanced Model-Based Engineering of Embedded Systems". Hrsg. von Klaus Pohl, Manfred Broy, Heinrich Daembkes und Harald Hönninger. Cham: Springer International Publishing, 2016.

[DiLi15] Christian Diedrich und Zheng Liu: „Ontologie in der Automation?“ In: "AUTOMATION". Hrsg. von VDI - Verein Deutscher Ingenieure e.V. 2015.

[Dim+14] Anastasia Dimou, Miel Vander Sande, Pieter Colpaert, Ruben Verbough, Erik Mannens und Rik van de Walle: „RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data“. In: "Proceedings of the 7th Workshop on Linked Data on the Web". 2014.

[DLA15] Aline Dresch, Daniel Pacheco Lacerda und José Antônio Valle Antunes Jr: "Design Science Research". Cham: Springer International Publishing, 2015.

[Dot+18] Mariagrazia Dotoli, Alexander Fay, Marek Miśkowicz und Carla Seatzu: „An overview of current technologies and emerging trends in factory automation“. In: "International Journal of Production Research", 2018, S. 1–21.

[DPW18] Henrik Dibowski, Joern Ploennigs und Martin Wollschlaeger: „Semantic Device and System Modeling for Automation Systems and Sensor Networks“. In: "IEEE Transactions on Industrial Informatics" 14.4, 2018, S. 1298–1311.

[Dra10] Rainer Drath: "Datenaustausch in der Anlagenplanung mit AutomationML". Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[DTW12] Marian Daun, Bastian Tenbergen und Thorsten Weyer: „Requirements Viewpoint“. In: "Model-Based Engineering of Embedded Systems". Hrsg. von Klaus Pohl, Harald Hönninger, Reinhold Achatz und Manfred Broy. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, S. 51–68.

[Dum+15] Roman Dumitrescu, Christian Bremer, Arno Kühn, Ansgar Trächtler und Tanja Frieben: „Model-based development of products, processes and production resources“. In: "at - Automatisierungstechnik" 63.10, 2015,

[Eis89] Kathleen M. Eisenhardt: „Building Theories from Case Study Research“. In: "Academy of Management Review" 14.4, 1989, S. 532–550.

[ElM06] Hoda A. ElMaraghy: „Flexible and reconfigurable manufacturing systems paradigms“. In: "International Journal of Flexible Manufacturing Systems" 17.4, 2006, S. 261–276.

[Epp11] Ulrich Epple: „Merkmale als Grundlage der Interoperabilität technischer Systeme“. In: "at - Automatisierungstechnik" 59.7, 2011, S. 440–450.

[EuSh13] Jérôme Euzenat und Pavel Shvaiko: "Ontology Matching". Springer Berlin Heidelberg, 2013.

[Fay+15] Alexander Fay, Christian Diedrich, Mario Thron, André Scholz und Philipp Puntel Schmidt: „Wie bekommt Industrie 4.0 Bedeutung? Normen und Standards als semantische Basis“. In: "atp - edition" 57.7-8, 2015, S. 30–41.

[FFV13] David Friedrich, Stefan Feldmann und Birgit Vogel-Heuser: „Modularisierung und Wiederverwendung in der Elektroplanung“. In: "Engineering von der Anforderung bis zum Betrieb". Hrsg. von Birgit Vogel-Heuser. Kassel, Deutschland: Kassel University Press, 2013, S. 28–40.

[FGJ97] Mariano Fernández-López, Asunción Gómez-Pérez und Natalia Juristo: „METHONTOLOGY: From Ontological Art Towards Ontological Engineering“. In: "Proceedings of the Ontological Engineering AAAI-97 Spring Symposium Series". 1997.

[Fio+15] Sandro Rama Fiorini u. a.: „Extensions to the core ontology for robotics and automation“. In: "Robotics and Computer-Integrated Manufacturing" 33, 2015, S. 3–11.

[FNB07] Jennifer Ferreira, James Noble und Robert Biddle: „Agile Development Iterations and UI Design“. In: "AGILE 2007". 2007, S. 50–58.

[FSM08] David Fox, Jonathan Sillito und Frank Maurer: „Agile Methods and User-Centered Design: How These Two Methodologies are Being Successfully Integrated in Industry“. In: "Agile 2008 Conference". 2008, S. 63–72.

[FuVa04] Lidia Fuentes-Fernández und Antonio Vallecillo-Moreno: „An Introduction to UML Profiles“. In: "UML and Model Engineering", 2004,

[GaBe12] Jürgen Gausemeier und Benjamin Behmann, Hrsg.: "Produkte und Produktionssysteme integrativ konzipieren: Modellbildung und Analyse in der frühen Phase der Produktentstehung". 1. Auflage. Carl Hanser Fachbuchverlag, 2012.

[Gan05] Aldo Gangemi: „Ontology Design Patterns for Semantic Web Content“. In: "The Semantic Web – ISWC 2005". Hrsg. von David Hutchison u. a. Bd. 3729. Lecture notes in computer science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, S. 262–276.

[GaPr09] Aldo Gangemi und Valentina Presutti: „Ontology Design Patterns“. In: "Handbook on Ontologies". Hrsg. von Steffen Staab und Rudi Studer. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[Gau05] D. C. Gause: „Why Context Matters–And What Can We Do about It?“ In: "IEEE Software" 22.5, 2005, S. 13–15.

[GFC04] Asunción Gómez-Pérez, Mariano Fernández-López und Oscar Corcho: "Ontological Engineering". London: Springer-Verlag, 2004.

[Gla+15] Matthias Glawe, Christopher Tebbe, Alexander Fay und Karl-Heinz Niemann: „Knowledge-based engineering of automation systems using ontologies and engineering data“. In: "7th International Conference on Knowledge Engineering and Ontology Development". 2015.

[GlFa16] Matthias Glawe und Alexander Fay: „Wissensbasiertes Engineering automatisierter Anlagen unter Verwendung von AutomationML und OWL“. In: "at - Automatisierungstechnik" 64.3, 2016,

[Gra+18] Irlán Grangel-González, Lavdim Halilaj, Maria-Esther Vidal, Omar Rana, Steffen Lohmann, Sören Auer und Andreas W. Müller: „Knowledge Graphs for Semantically Integrating Cyber-Physical Systems“. In: "Database and Expert Systems Applications". Hrsg. von Sven Hartmann, Hui Ma, Abdelkader Hameurlain, Günther Pernul und Roland R. Wagner. Bd. 11029. Lecture notes in computer science. Cham: Springer International Publishing, 2018, S. 184–199.

[Gra18] Markus Graube: „Linked Enterprise Data als semantischer, integrierter Informationsraum für die industrielle Datenhaltung“. Dissertation. Dresden, 2018.

[GrFo95] Michael Grüninger und Mark S. Fox: „Methodology for the design and evaluation of ontologies“. In: "IJCAI95 Workshop on Basic Ontological Issues in Knowledge Sharing". 1995.

[Hal+16] Lavdim Halilaj, Niklas Petersen, Irlán Grangel-González, Christoph Lange, Sören Auer, Gökhan Coskun und Steffen Lohmann: „VoCol: An Integrated Environment to Support Version-Controlled Vocabulary Development“. In: "Knowledge Engineering and Knowledge Management". Hrsg. von Eva Blomqvist, Paolo Ciancarini, Francesco Poggi und Fabio Vitali. Bd. 10024. Lecture notes in computer science. Cham: Springer International Publishing, 2016, S. 303–319.

[Hal05] Alon Halevy: „Why your data won't mix“. In: "Queue" 3.8, 2005, S. 50.

[Ham17] Karl Hammar: „Content Ontology Design Patterns: Qualities, Methods, and Tools“. Dissertation. Jönköping University, 2017.

[HaVr15] Ondrej Harcuba und Pavel Vrba: „Ontologies for Flexible Production Systems“. In: "20th IEEE Conference on Emerging Technologies & Factory Automation". 2015.

[Hel02] Markus Helfert: „Planung und Messung der Datenqualitat in Data-Warehouse-Systemen“. Dissertation. St. Gallen, 2002.

[Hen11] B. Henderson-Sellers: „Bridging metamodels and ontologies in software engineering“. In: "Journal of Systems and Software" 84.2, 2011, S. 301–313.

[Hev+04] Alan R. Hevner, Salvatore T. March, Park Jinsoo und Sudha Ram: „Design Science in Information Systems Research“. In: "MIS Quarterly" 28.1, 2004,

[Hit+08] Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph und York Sure: "Semantic Web". Springer Berlin Heidelberg, 2008.

[HKR10] Pascal Hitzler, Markus Krötzsch und Sebastian Rudolph: "Foundations of semantic web technologies". Chapman & Hall/CRC textbooks in computing. Boca Raton, Fla., London und New York: CRC Press, 2010.

[HuHe19] Yingbing Hua und Bjorn Hein: „Interpreting OWL Complex Classes in AutomationML based on Bidirectional Translation“. In: "24th IEEE International Conference on Emerging Technologies and Factory Automation". 2019, S. 79–86.

[HWH14] Tobias Helbig, Engelbert Westkämper und Johannes Hoos: „Identifying automation components in modular manufacturing systems“. In: "2014 IEEE Emerging Technology and Factory Automation". 2014.

[Iar+16] Sergii Iarovyi, Wael M. Mohammed, Andrei Lobov, Borja Ramis Ferrer und Jose L. Martinez Lastra: „Cyber–Physical Systems for Open-Knowledge-Driven Manufacturing Execution Systems“. In: "Proceedings of the IEEE" 104.5, 2016, S. 1142–1154.

[Jan+16] Krzysztof Janowicz, Aldo Gangemi, Pascal Hitzler, Adila Krisnadhi und Valentina Presutti: „Introduction: Ontology Design Patterns in a Nutshell“. In: "Ontology engineering with ontology design patterns". Hrsg. von Aldo Gangemi, Adila Krisnadhi, Krzysztof Janowicz, Pascal Hitzler und Valentina Presutti. Studies on the semantic web. Amsterdam, Netherlands: IOS Press, 2016.

[Jär+19] Eeva Järvenpää, Niko Siltala, Otto Hylli und Minna Lanz: „The development of an ontology for describing the capabilities of manufacturing resources“. In: "Journal of Intelligent Manufacturing" 30.2, 2019, S. 959–978.

[Jär12] Eeva Järvenpää: „Capability-based Adaptation of Production Systems in a Changing Environment“. Dissertation. Tampere: Tampere University of Technology, 2012.

[JaSa05] D. Janzen und H. Saiedian: „Test-driven development concepts, taxonomy, and future direction“. In: "Computer" 38.9, 2005, S. 43–50.

[JaUs99] Robert Jasper und Mike Uschold: „A framework for understanding and classifying ontology applications.“ In: "Proceedings 12th Int. Workshop on Knowledge Acquisition, Modelling, and Management KAW". 1999.

[Jir+18] Vaclav Jirkovsky, Marek Obitko, Petr Kadera und Vladimir Marik: „Toward Plug&Play Cyber-Physical System Components“. In: "IEEE Transactions on Industrial Informatics" 14.6, 2018, S. 2803–2811.

[JOM17] Vaclav Jirkovsky, Marek Obitko und Vladimir Marik: „Understanding Data Heterogeneity in the Context of Cyber-Physical Systems Integration“. In: "IEEE Transactions on Industrial Informatics" 13.2, 2017, S. 660–667.

[JoPe14] Paul Johannesson und Erik Perjons: "An Introduction to Design Science". Cham: Springer International Publishing, 2014.

[JSL13] Giouvanni Désiré Jules, Mozafar Saadat und Nan Li: „On Designing a Unified Ontology for Holonic Manufacturing Networks“. In: "Integration of Practice-Oriented Knowledge Technology: Trends and Prospectives". Hrsg. von Madjid Fathi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, S. 207–220.

[Kaz+13] Daria Kazanskaia, Sergey Kozhevnikov, Vladimir Larukhin, Petr Skobelev, Alexander Tsarev und Yaroslav Shepilov: „Knowledge-Driven Adaptive Production Management Based on Real-Time User Feedback and Ontology Updates“. In: "IEEE International Conference on Systems, Man, and Cybernetics". IEEE, 2013, S. 4755–4759.

[KeŁa16] C. Maria Keet und Agnieszka Ławrynowicz: „Test-Driven Development of Ontologies“. In: "The Semantic Web. Latest Advances and New Domains". Hrsg. von Harald Sack, Eva Blomqvist, Mathieu d'Aquin, Chiara Ghidini, Simone Paolo Ponzetto und Christoph Lange. Bd. 9678. Lecture notes in computer science. Cham: Springer International Publishing, 2016, S. 642–657.

[KHH17] Nazifa Karima, Karl Hammar und Pascal Hitzler: „How to Document Ontology Design Patterns“. In: "Advances in ontology design and patterns". Hrsg. von Adila Krisnadhi, Agnieszka Ławrynowicz, Andrea Giovanni Nuzzolese, Karl Hammar, Monika Solanki und Pascal Hitzler. Studies on the semantic web. Amsterdam, Netherlands: IOS Press, 2017.

[KJW17] Matthias Klein, Nasser Jazdi und Michael Weyrich: „A Concept of Semantic Description for e-Production Systems in Manufacturing“. In: "Procedia CIRP" 62, 2017, S. 589–593.

[KoSp15] Nikolaos Konstantinou und Dimitrios-Emmanuel Spanos: "Materializing the Web of Linked Data". Cham: Springer International Publishing, 2015.

[KoVo06] Konstantinos Kotis und George A. Vouros: „Human-centered ontology engineering: The HCOME methodology“. In: "Knowledge and Information Systems" 10.1, 2006, S. 109–131.

[LaMc01] Ora Lassila und Deborah McGuinness: „The role of frame-based representation on the semantic web“. In: "Linköping Electronic Articles in Computer and Information Science" 6.5, 2001,

[LaWe10] Jesko G. Lamm und Tim Weilkiens: „Functional Architectures in SysML“. In: "Tag des Systems Engineering". Hrsg. von M. Maurer und S.-O. Schulze. 2010.

[Lee08] Edward A. Lee: „Cyber Physical Systems: Design Challenges“. In: "International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)". 2008, S. 363–369.

[Leg12] Christoph Legat: „Knowledge-based Technologies for Future Factory Engineering and Control“. In: "IFAC Proceedings Volumes" 45.6, 2012, S. 44–48.

[Lem+06] Severin Lemaignan, Ali Siadat, Jean-Yves Dantan und Anatoli Semenenko: „MASON: A Proposal For An Ontology Of Manufacturing Domain“. In: "IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and its Applications". Piscataway, NJ: IEEE, 2006.

[Lie+19] Thorsten Liebig, Andreas Maisenbacher, Michael Opitz, Jan R. Seyler, Gunther Sudra und Jens Wissmann: „Building a Knowledge Graph for Products and Solutions in the Automation Industry“. In: "The Semantic Web". Hrsg. von Pascal Hitzler, Miriam Fernández, Krzysztof Janowicz, Amrapali Zaveri, Alasdair J.G. Gray, Vanessa Lopez, Armin Haller und Karl Hammar. Cham: Springer International Publishing, 2019.

[Los+12] Matthias Loskyll, Ines Heck, Jochen Schlick und Michael Schwarz: „Context-Based Orchestration for Control of Resource-Efficient Manufacturing Processes“. In: "Future Internet" 4.4, 2012, S. 737–761.

[Los13] Matthias Loskyll: „Entwicklung einer Methodik zur dynamischen kontextbasierten Orchestrierung semantischer Feldgerätefunktionalitäten“. Dissertation. Technische Universität Kaiserslautern, 2013.

[LSV14] Christoph Legat, Daniel Schütz und Birgit Vogel-Heuser: „Automatic generation of field control strategies for supporting (re-)engineering of manufacturing systems“. In: "Journal of Intelligent Manufacturing" 25.5, 2014, S. 1101–1111.

[Lud+18] Alexander Ludewig, Marian Daun, Ana Petrovska, Wolfgang Böhm und Alexander Fay: „Requirements for modeling dynamic function networks for collaborative embedded systems“. In: "Fachtagung Modellierung". Hrsg. von Ina Schäfer und Dimitris Karagiannis. 2018.

[LZB17] Maxime Lefrançois, Antoine Zimmermann und Noorani Bakerally: „A SPARQL Extension for Generating RDF from Heterogeneous Formats“. In: "The Semantic Web". Hrsg. von Eva Blomqvist, Diana Maynard, Aldo Gangemi, Rinke Hoekstra, Pascal Hitzler und Olaf Hartig. Bd. 10249. Lecture notes in computer science. Cham: Springer International Publishing, 2017, S. 35–50.

[Mar+07] David Martin, Mark Burstein, Drew McDermott, Sheila McIlraith, Massimo Paolucci, Katia Sycara, Deborah L. McGuinness, Evren Sirin und Naveen Srinivasan: „Bringing Semantics to Web Services with OWL-S“. In: "World Wide Web" 10.3, 2007, S. 243–277.

[Mat+15] Nicolas Matentzoglu, Jared Leo, Valentino Hudhra, Bijan Parsia und Uli Sattler: „A survey of current, stand-alone OWL Reasoners“. In: "Informal Proceedings of the 4th International Workshop on OWL Reasoner Evaluation". 2015.

[MLD09] Wolfgang Mahnke, Stefan-Helmut Leitner und Matthias Damm: "OPC Unified Architecture". Springer-Verlag Berlin Heidelberg, 2009.

[Mon15] László Monostori: „Cyber-physical production systems: Roots from manufacturing science and technology“. In: "at - Automatisierungstechnik" 63.10, 2015,

[Mus15] Mark A. Musen: „The Protégé Project: A Look Back and a Look Forward“. In: "AI matters" 1.4, 2015, S. 4–12.

[NiMi16] Antonio de Nicola und Michele Missikoff: „A lightweight methodology for rapid ontology engineering“. In: "Communications of the ACM" 59.3, 2016, S. 79–86.

[NMN09] Antonio de Nicola, Michele Missikoff und Roberto Navigli: „A software engineering approach to ontology building“. In: "Information Systems" 34.2, 2009, S. 258–275.

[OHM10] Martin J. O'Connor, Christian Halaschek-Wiener und Mark A. Musen: „Mapping Master: A Flexible Approach for Mapping Spreadsheets to OWL“. In: "The Semantic Web – ISWC 2010". Hrsg. von Peter F. Patel-Schneider, Yue Pan, Pascal Hitzler, Peter Mika, Lei Zhang, Jeff Z. Pan, Ian Horrocks und Birte Glimm. Bd. 6497. Lecture notes in computer science. Springer Berlin Heidelberg, 2010, S. 194–208.

[OPV19] Felix Ocker, Christiaan J.J. Paredis und Birgit Vogel-Heuser: „Applying Knowledge Bases to Make Factories Smarter“. In: "at - Automatisierungstechnik", 2019,

[Pan+17] Jeff Z. Pan, Guido Vetere, Jose Manuel Gomez-Perez und Honghan Wu: "Exploiting Linked Data and Knowledge Graphs in Large Organisations". Cham: Springer International Publishing, 2017.

[Pau16] Heiko Paulheim: „Knowledge graph refinement: A survey of approaches and evaluation methods“. In: "Semantic Web" 8.3, 2016, S. 489–508.

[Pau18] Heiko Paulheim: „How much is a Triple? Estimating the Cost of Knowledge Graph Creation“. In: "The Semantic Web – ISWC 2018". Hrsg. von Denny Vrandečić, Kalina Bontcheva, Mari Carmen Suárez-Figueroa, Valentina Presutti, Irene Celino, Marta Sabou, Lucie-Aimée Kaffee und Elena Simperl. Cham: Springer International Publishing, 2018.

[Per+19] Alexander Perzylo, Stefan Profanter, Markus Rickert und Alois Knoll: „OPC UA NodeSet Ontologies as a Pillar of Representing Semantic Digital Twins of Manufacturing Resources“. In: "24th IEEE International Conference on Emerging Technologies and Factory Automation". 2019.

[Per17] Silvio Peroni: „A Simplified Agile Methodology for Ontology Development“. In: "OWL: Experiences and Directions – Reasoner Evaluation". Hrsg. von Mauro Dragoni, María Poveda-Villalón und Ernesto Jimenez-Ruiz. Bd. 10161. Lecture notes in computer science. Cham: Springer International Publishing, 2017, S. 55–69.

[Pet+17] Niklas Petersen, Lavadim Halilaj, Irlán Grangel-González, Steffen Lohmann, Christoph Lange und Sören Auer: „Realizing an RDF-based Information Model for a Manufacturing Company – A Case Study“. In: "The semantic web – ISWC 2017". Hrsg. von Claudia D'Amato. Lecture notes in computer science. Cham: Springer, 2017.

[Pfr+16] Julius Pfrommer, Sten Grüner, Thomas Goldschmidt und Dirk Schulz: „A common core for information modeling in the Industrial Internet of Things“. In: "at - Automatisierungstechnik" 64.9, 2016, S. 1.

[PLM13] Juha Puttonen, Andrei Lobov und Martinez Lastra: „Semantics-Based Composition of Factory Automation Processes Encapsulated by Web Services“. In: "IEEE Transactions on Industrial Informatics" 9.4, 2013, S. 2349–2359.

[Pre+09] Valentina Presutti, Enrico Daga, Aldo Gangemi und Eva Blomqvist: „eXtreme Design with Content Ontology Design Patterns“. In: "Proceedings of the 2009 International Conference on Ontology Patterns - Volume 516". WOP'09. Aachen, Germany, Germany: CEUR-WS.org, 2009.

[Pre+12] Valentina Presutti, Eva Blomqvist, Enrico Daga und Aldo Gangemi: „Pattern-Based Ontology Design“. In: "Ontology Engineering in a Networked World". Hrsg. von Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez, Enrico Motta und Aldo Gangemi. Springer Berlin Heidelberg, 2012.

[PST04] Sofia Pinto, Steffen Staab und Christoph Tempich: „DILIGENT: Towards a fine-grained methodology for DIstributed, Loosely-controlled and evolvInG Engineering of oNTologies“. In: "Proceedings". Hrsg. von Ramón López de Mántaras. Frontiers in artificial intelligence and applications. Amsterdam u.a.: IOS Press, 2004.

[Pun+16] Philipp Puntel Schmidt, Michel Goß, Alexander Fay, Henning Bertsch, Berthold Welling und Frank Marek: „Ein Reifegradmodell für Engineering-Daten“. In: "wt Werkstatttechnik" 106.9, 2016,

[Pun17] Philipp Puntel Schmidt: „Methoden zur simulationsbasierten Absicherung von Steuerungscode fertigungstechnischer Anlagen“. Dissertation. Hamburg: Helmut-Schmidt-Universität, 2017.

[Reg+18] Nikolaus Regnat, Andreas Vogelsang, Philipp Diebold, Sieglinde Kranz, Thorsten Weyer, Andreas Wortmann, Arvid Butting und Diego Marmsoler: "Consolidated Requirements for a Seamless Modeling Methodology for Collaborative Embedded Systems". Hrsg. von Teilprojekt Methdischer Querschnitt 1 - Modellierung des Projektes CrESt. Projektinterner Bericht des Projektes CrESt (Collaborative Embedded Systems), 2018.

[Rei00] Jacqueline René Reich: „Ontological Design Patterns: Metadata of Molecular Biological Ontologies, Information and Knowledge“. In: "Database and Expert Systems Applications". Hrsg. von Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, Mohamed Ibrahim, Josef Küng und Norman Revell. Bd. 1873. Lecture notes in computer science. Springer Berlin Heidelberg, 2000, S. 698–709.

[Ren+14] Yuan Ren, Artemis Parvizi, Chris Mellish, Jeff Z. Pan, Kees van Deemter und Robert Stevens: „Towards Competency Question-Driven Ontology Authoring“. In: "The Semantic Web: Trends and Challenges". Hrsg. von David Hutchison u. a. Bd. 8465. Lecture notes in computer science. Cham: Springer International Publishing, 2014, S. 752–767.

[Row07] Jennifer Rowley: „The wisdom hierarchy: representations of the DIKW hierarchy“. In: "Journal of Information Science" 33.2, 2007, S. 163–180.

[Run+11] Stefan Runde, Alexander Fay, Stefan Schmitz und Ulrich Epple: „Wissensbasierte Systeme im Engineering der Automatisierungstechnik“. In: "at - Automatisierungstechnik" 59.1, 2011,

[Run10] Stefan Runde: „Wissensbasierte Engineeringunterstützung in der Automatisierungstechnik am Beispiel der Gebäudeautomatisierung“. Dissertation. Hamburg: Helmut-Schmidt-Universität, 2010.

[Sab+19] Marta Sabou, Stefan Biffl, Alfred Einfalt, Lukas Krammer, Wolfgang Kastner und Fajar J. Ekaputra: „Semantics for Cyber-Physical Systems: A Cross-Domain Perspective“. In: "Semantic Web – Interoperability, Usability, Applicability", 2019,

[Sak+18] Sherif Sakr, Marcin Wylot, Raghava Mutharaju, Danh Le Phuoc und Irini Fundulaki: "Linked data: Storing, querying, and reasoning". Cham: Springer, 2018.

[Sau98] Günther Sauter: „Interoperabilität von Datenbanksystemen bei struktureller Heterogenität“. Dissertation. 1998.

[Sch+12] Craig Schlenoff, Edson Prestes, Raj Madhavan, Paulo Goncalves, Howard Li, Stephen Balakirsky, Thomas Kramer und Emilio Miguelanez: „An IEEE standard Ontology for Robotics and Automation“. In: "2012 IEEE/RSJ International Conference on Intelligent Robots and Systems". 2012, S. 1337–1342.

[Sch+14] Nicole Schmidt, Arndt Lüder, Stefan Biffl und Heinrich Steininger: „Analyzing Requirements on Software Tools According to the Functional Engineering Phase in the Technical Systems Engineering Process“. In: "2014 IEEE Emerging Technology and Factory Automation". 2014.

[Sch13] Miriam Schleipen: „Adaptivität und semantische Interoperabilität von Manufacturing Execution Systemen“. Dissertation. Technische Universität Kaiserslautern, 2013.

[Sch18] André Scholz: „Unterstützung des Engineerings von fertigungstechnischen Produktionssystemen mit Hilfe von Maschinenfunktionen“. Dissertation. Hamburg: Helmut-Schmidt-Universität, 2018.

[Sch99] Eckehard Schnieder: "Methoden der Automatisierung: Beschreibungsmittel, Modellkonzepte und Werkzeuge für Automatisierungssysteme". Studium Technik. Wiesbaden: Vieweg+Teubner Verlag, 1999.

[ScSc08] Eckehard Schnieder und Lars Schnieder: „Axiomatik der Begriffe für die Automatisierungstechnik“. In: "atp - edition" 10, 2008, S. 62–73.

[SGV09] Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez und Boris Villazón-Terrazas: „How to Write and Use the Ontology Requirements Specification Document“. In: "On the Move to Meaningful Internet Systems: OTM 2009". Hrsg. von David Hutchison u. a. Bd. 5871. Lecture notes in computer science. Springer Berlin Heidelberg, 2009, S. 966–982.

[Sil+11] Tiago Silva da Silva, Angela Martin, Frank Maurer und Milene Silveira: „User-Centered Design and Agile Methods: A Systematic Review“. In: "2011 AGILE Conference". 2011, S. 77–86.

[SiLu14] Elena Simperl und Markus Luczak-Rösch: „Collaborative ontology engineering: A survey“. In: "The Knowledge Engineering Review" 29.01, 2014, S. 101–131.

[Sir+07] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur und Yarden Katz: „Pellet: A practical OWL-DL reasoner“. In: "Web Semantics: Science, Services and Agents on the World Wide Web" 5.2, 2007, S. 51–53.

[SKF19] Umutcan Simsek, Elias Kärle und Dieter Fensel: „RocketRML - A NodeJS implementation of a use-case specific RML mapper“. In: "Knowledge Graph Building Workshop at European Semantic Web Conference". Hrsg. von Anastasia Dimou, David Chaves Fraga, Freddy Priyatna, Juan F. Sequeda und Pieter Heyvaert. Cham: Springer International Publishing, 2019.

[Skj+18] Martin G. Skjaeveland, Daniel P. Lupp, Leif Harald Karlsen und Henrik Forssell: „Practical Ontology Pattern Instantiation, Discovery, and Maintenance with Reasonable Ontology Templates“. In: "The Semantic Web – ISWC 2018". Hrsg. von Denny Vrandečić, Kalina Bontcheva, Mari Carmen Suárez-Figueroa, Valentina Presutti, Irene Celino, Marta Sabou, Lucie-Aimée Kaffee und Elena Simperl. Cham: Springer International Publishing, 2018.

[SPH12] Mari Carmen Suárez-Figueroa, Camille Pradel und Nathalie Hernandez: „Verifying ontology requirements with SWIP“. In: "18th International Conference on Knowledge Engineering and Knowledge Management". 2012.

[SSS09] York Sure, Steffen Staab und Rudi Studer: „Ontology Engineering Methodology“. In: "Handbook on Ontologies". Hrsg. von Steffen Staab und Rudi Studer. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[Sta+01] S. Staab, R. Studer, H.-P. Schnurr und Y. Sure: „Knowledge processes and ontologies“. In: "IEEE Intelligent Systems" 16.1, 2001, S. 26–34.

[Str+11] Martin Strube, Stefan Runde, Helmut Figalist und Alexander Fay: „Risk minimization in modernization projects of plant automation — A knowledge-based approach by means of semantic web technologies“. In: "Factory Automation (ETFA 2011)". 2011, S. 1–8.

[StSt09] Steffen Staab und Rudi Studer, Hrsg.: "Handbook on Ontologies". Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[Stu11] Heiner Stuckenschmidt: "Ontologien: Konzepte, Technologien und Anwendungen". 2. Aufl. Informatik im Fokus. Berlin: Springer, 2011.

[Sy07] Desirée Sy: „Adapting usability investigations for agile user-centered design“. In: "Journal of usability Studies" 2.3, 2007, S. 112–132.

[Tar03] Peter Tarasewich: „Towards a Comprehensive Model of Context for Mobile and Wireless Computing“. In: "9th American Conference on Information Systems". 2003.

[Thr10] Kleanthis Thramboulidis: „The 3+1 SysML View-Model in Model Integrated Mechatronics“. In: "Journal of Software Engineering and Applications" 03.02, 2010, S. 109–118.

[Tol06] Andreas. Tolk: „What Comes After the Semantic Web - PADS Implications for the Dynamic Web“. In: "20th Workshop on Principles of Advanced and Distributed Simulation (PADS'06)". 2006, S. 55.

[UsGr04] Michael Uschold und Michael Gruninger: „Ontologies and semantics for seamless connectivity“. In: "ACM SIGMOD Record" 33.4, 2004, S. 58.

[UsGr96] Mike Uschold und Michael Gruninger: „Ontologies: Principles, methods and applications“. In: "The Knowledge Engineering Review" 11.02, 1996, S. 93.

[UTY16] Inayat Ullah, Dunbing Tang und Leilei Yin: „Engineering Product and Process Design Changes: A Literature Overview“. In: "Procedia CIRP" 56, 2016, S. 25–33.

[VDB13] Birgit Vogel-Heuser, Christian Diedrich und Manfred Broy: „Anforderungen an CPS aus Sicht der Automatisierungstechnik“. In: "at – Automatisierungstechnik" 61.10, 2013,

[ViGó12] Boris Villázon-Terrazas und Asunción Gómez-Pérez: „Reusing and Re-engineering Non-ontological Resources for Building Ontologies“. In: "Ontology Engineering in a Networked World". Hrsg. von Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez, Enrico Motta und Aldo Gangemi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[Vil+17] Boris Villazon-Terrazas, Nuria Garcia-Santa, Yuan Ren, Kavitha Srinivas, Mariano Rodriguez-Muro, Panos Alexopoulos und Jeff Z. Pan: „Construction of Enterprise Knowledge Graphs (I)“. In: "Exploiting Linked Data and Knowledge Graphs in Large Organisations". Hrsg. von Jeff Z. Pan, Guido Vetere, Jose Manuel Gomez-Perez und Honghan Wu. Cham: Springer International Publishing, 2017.

[Vog+09] Birgit Vogel-Heuser, Gunther Kegel, Klaus Bender und Klaus Wucherer: „Global Information Architecture for Industrial Automation“. In: "atp - edition" 1, 2009, S. 108–115.

[Vog+14a] Birgit Vogel-Heuser, Christian Diedrich, Alexander Fay, Sabine Jeschke, Stefan Kowalewski, Martin Wollschlaeger und Peter Göhner: „Challenges for Software Engineering in Automation“. In: "Journal of Software Engineering and Applications" 07.05, 2014, S. 440–451.

[Vog+14b] Birgit Vogel-Heuser, Daniel Schütz, Timo Frank und Christoph Legat: „Model-driven engineering of Manufacturing Automation Software Projects – A SysML-based approach“. In: "Mechatronics" 24.7, 2014, S. 883–897.

[VrGa06] Denny Vrandečić und Aldo Gangemi: „Unit Tests for Ontologies“. In: "On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops". Hrsg. von David Hutchison u. a. Bd. 4278. Lecture notes in computer science. Springer Berlin Heidelberg, 2006, S. 1012–1020.

[Wau+14] Yves Wautelet, Samedi Heng, Manuel Kolp und Isabelle Mirbel: „Unifying and Extending User Story Models“. In: "Advanced Information Systems Engineering". Hrsg. von David Hutchison u. a. Bd. 8484. Lecture notes in computer science. Cham: Springer International Publishing, 2014, S. 211–225.

[WeSo14] Tim Weilkiens und Richard Mark Soley: "Systems Engineering mit SysML/UML: Anforderungen, Analyse, Architektur". 3., überarb. und aktualisierte Aufl. Heidelberg: dpunkt.verl., 2014.

[Wey10] Thorsten Weyer: „Kohärenzprüfung von Verhaltensspezifikationen gegen spezifische Eigenschaften des operationellen Kontexts“. Dissertation. Duisburg-Essen: Universität Duisburg-Essen, 2010.

[Wie+07] H.-P. Wiendahl, H. A. ElMaraghy, P. Nyhuis, M. F. Zäh, H.-H. Wiendahl, N. Duffie und M. Brieke: „Changeable Manufacturing - Classification, Design and Operation“. In: "CIRP Annals - Manufacturing Technology" 56.2, 2007, S. 783–809.

[Wie14] Roel J. Wieringa: "Design Science Methodology for Information Systems and Software Engineering". Springer Berlin Heidelberg, 2014.

[YCY19a] Lan Yang, Kathryn Cormican und Ming Yu: „Learning Systems Engineering Domain Ontologies from Text Documents“. In: "2019 International Symposium on Systems Engineering (ISSE)". IEEE, 2019.

[YCY19b] Lan Yang, Kathryn Cormican und Ming Yu: „Ontology-based systems engineering: A state-of-the-art review“. In: "Computers in Industry" 111, 2019, S. 148–171.

Normen und Richtlinien

[ANSI/ISA 88] ANSI/ISA: "Machine and Unit States: An implementation example of ANSI/ISA-88.00.01". ANSI/ISA 88. 2015.

[DIN 62541-1] DIN Deutsches Institut für Normung e. V.: "OPC Unified Architecture: Teil 1: Übersicht und Konzepte". DIN 62541-1. 2011.

[DIN 8580] DIN Deutsches Institut für Normung e. V.: "Fertigungsverfahren". DIN 8580. 2003.

[DIN EN 61360-1] DIN Deutsches Institut für Normung e. V.: "Genormte Datenelementtypen mit Klassifikationsschema für elektrische Bauteile Teil 1: Definitionen - Regeln und Methoden". DIN EN 61360-1. 2018.

[DIN EN 62264 -1] DIN Deutsches Institut für Normung e. V.: "Integration von Unternehmensführungs- und Leitsystemen". DIN EN 62264 -1. 2014.

[DIN EN 62541-5] DIN Deutsches Institut für Normung e. V.: "OPC Unified Architecture: Teil 5: Informationsmodell". DIN EN 62541-5. 2019.

[DIN EN 62714-1] DIN Deutsches Institut für Normung e. V.: "Datenaustauschformat für Planungsdaten industrieller Automatisierungssysteme". DIN EN 62714-1. 2015.

[IEEE 1872-2015] IEEE Institute of Electrical and Electronics Engineers: "IEEE Standard Ontologies for Robotics and Automation". IEEE 1872-2015. Piscataway, NJ, USA, 2015.

[ISO 10303-44] International Organization for Standardization: "Industrial automation systems and integration - Product data representation and exchange - Part 44: Integrated generic resource: Product structure configuration". ISO 10303-44. 2000.

[ISO 15288] ISO IEC IEEE: "Systems and software engineering - System life cycle processes". ISO 15288. 2015.

[VDI 2206] VDI - Verein Deutscher Ingenieure e.V.: "Entwicklungsmethodik für mechatronische Systeme". VDI 2206. 2004.

[VDI 3695-1] VDI - Verein Deutscher Ingenieure: "Engineering von Anlagen". VDI 3695-1. 2010.

[VDI 4499-1] VDI - Verein Deutscher Ingenieure e.V.: "Digitale Fabrik". VDI 4499-1. 2008.

[VDI 4499-3] VDI - Verein Deutscher Ingenieure e.V.: "Digitale Fabrik". VDI 4499-3. 2016.

[VDI 5600-3] VDI - Verein Deutscher Ingenieure e.V.: "Fertigungsmanagementsysteme (Manufacturing Execution Systems - MES): Logische Schnittstelle zur Maschinen- und Anlagensteuerung". VDI 5600-3. 2013.

[VDI 5610-1] VDI - Verein Deutscher Ingenieure e.V.: "Wissensmanagement im Ingenieurwesen". VDI 5610-1. 2009.

[VDI/VDE 3682-1] VDI/VDE - Verein Deutscher Ingenieure, Verband der Elektrotechnik, Elektronik, Informationstechnik: "Formalisierte Prozessbeschreibungen". VDI/VDE 3682-1. 2015.

[VDI/VDE 3682-2] VDI/VDE - Verein Deutscher Ingenieure, Verband der Elektrotechnik, Elektronik, Informationstechnik: "Formalisierte Prozessbeschreibungen". VDI/VDE 3682-2. 2015.

Publikationen des Autors

[Gla+18*] Matthias Glawe, Constantin Hildebrandt, Jörn Peschke und Alexander Fay: „Semantische Ermittlung kinematischer Fähigkeiten aus Anlagenplanungsdaten“. In: "at - Automatisierungstechnik" 66.5, 2018, S. 385–396.

[Hil+17*a] Constantin Hildebrandt, André Scholz, Alexander Fay, Tizian Schröder, Thomas Hadlich, Christian Diedrich, Martin Dubovy, Christian Eck und Ralf Wiegand: „Semantic Modeling for Collaboration and Cooperation of Systems in the production domain“. In: "22nd IEEE Emerging Technology and Factory Automation". 2017.

[Hil+17*b] Constantin Hildebrandt, André Scholz, Alexander Fay, Tizian Schröder, Thomas Hadlich, Christian Diedrich, Martin Dubovy, Christian Eck und Ralf Wiegand: „Semantische Allianz 4.0: Semantische Inhalte für Industrie 4.0“. In: "AUTOMATION". Hrsg. von VDI - Verein Deutscher Ingenieure e.V. 2017.

[Hil+18*a] Constantin Hildebrandt, Sebastian Törsleff, Torsten Bandyszak, Birte Caesar, Alexander Ludewig und Alexander Fay: „Ontology Engineering for Collaborative Embedded Systems – Requirements and Initial Approach“. In: "Fachtagung Modellierung". Hrsg. von Ina Schäfer und Dimitris Karagiannis. 2018.

[Hil+18*b] Constantin Hildebrandt, Sebastian Törsleff, Birte Caesar und Alexander Fay: „Ontology Building for Cyber-Physical Systems: A domain expertcentric approach“. In: "2018 14th IEEE Conference on Automation Science and Engineering (CASE 2018)". 2018.

[Hil+19*] Constantin Hildebrandt, Torsten Bandyszak, Ana Petrovska, Nishanth Laxman, Emilia Cioroaica und Sebastian Törsleff: „EURECA: Epistemic Uncertainty Classification Scheme for Runtime Information Exchange in Collaborative System Groups“. In: "Computer Science - Research and Development", 2019,

[Hil+20*] Constantin Hildebrandt, Aljosha Köcher, Christof Küstner, Carlos-Manuel Lopez-Enriquez, Andreas W. Müller, Birte Caesar, Claas Steffen Gundlach und Alexander Fay: „Ontology Building for Cyber Physical Systems: Application in the Manufacturing Domain“. In: "IEEE Transactions on Automation Science and Engineering" 17.3, 2020,

[Köc+20*] Aljosha Köcher, Constantin Hildebrandt, Luis Miguel Vieira da Silva und Alexander Fay: „A Formal Capability and Skill Model for Use in Plug and Produce Scenarios“. In: "25th IEEE International Conference on Emerging Technologies and Factory Automation". 2020.

[NHF20*] Hamied Nabizada, Constantin Hildebrandt und Alexander Fay: „Offenes, webbasiertes Werkzeug zur Informationsmodellierung mit Formalisierter Prozessbeschreibung“. In: "AUTOMATION". Hrsg. von VDI - Verein Deutscher Ingenieure e.V. 2020.

[SHF17*] A. Scholz, C. Hildebrandt und A. Fay: „Functional modelling in production engineering workflows“. In: "2017 13th IEEE Conference on Automation Science and Engineering (CASE 2017)". 2017, S. 695–700.

[Tör+18*] Sebastian Törsleff, Constantin Hildebrandt, Marian Daun, Jennifer Brings und Alexander Fay: „Developing Ontologies for the Collaboration of Cyber-Physical Systems: Requirements and Solution Approach“. In: "Cyber-Physical System Week". 2018.

Studentische Arbeiten

[Gre19%] Tobias Grebner: „Automatisierte Reduktion semantischer Heterogenität in zentralen Datenbanken“. Masterarbeit. Hamburg: Helmut-Schmidt-Universität, 2019.

Internetquellen und Software

[Ber+20@] Thomas Bergwinki, Michael Luggen, Elf Pavlik, Blake Regalia, Piero Savastano und Ruben Verborgh: "RDF/JS: Data model specification". 2020. URL: http://rdf.js.org/data-model-spec/, besucht am 29. 12. 2020,

[BrGu14@] Dan Brickley und R. V. Guha: "RDF Schema 1.1". 2014. URL: https://www.w3.org/TR/2014/REC-rdf-schema-20140225/, besucht am 16. 01. 2020,

[CWL14@] Richard Cyganiak, David Wood und Markus Lanthaler: "RDF 1.1 Concepts and Abstract Syntax". 2014. URL: https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/, besucht am 16. 01. 2020,

[Ecl20@] Eclipse Foundation: "The rdf4j server REST API". 2020. URL: https://rdf4j.org/documentation/reference/rest-api/, besucht am 29. 12. 2020,

[Goo20@] Google LLC: "One framework, Mobile and Desktop". 2020. URL: https://angular.io/, besucht am 12. 03. 2020,

[GPP13@] Paula Gearon, Alexandre Passant und Axel Polleres: "SPARQL 1.1 Update". 2013. URL: https://www.w3.org/TR/sparql11-update/, besucht am 22. 01. 2020,

[Had09@] Marc Hadley: "Web Application Description Language". 2009. URL: https://www.w3.org/Submission/wadl/, besucht am 27. 12. 2019,

[Hal+17@] Armin Haller, Krzysztof Janowicz, Simon Cox, Danh Le Phuoc, Kerry Taylor und Maxime Lefrancois: "Semantic Sensor Network Ontology". 2017. URL: https://www.w3.org/TR/vocab-ssn/, besucht am 02. 01. 2020,

[HiKö19@a] Constantin Hildebrandt und Aljosha Köcher: "Industrial Standard Ontology Design Patterns". 2019. URL: https://github.com/hsu-aut/Industrial-Standard-Ontology-Design-Patterns, besucht am 29. 12. 2020,

[HiKö19@b] Constantin Hildebrandt und Aljosha Köcher: "Lightweight Industrial Ontology Support". 2019. URL: https://github.com/hsu-aut/lion, besucht am 29. 12. 2020,

[Hir+19@] Hartmut Hirsch-Kreinsen, Uwe Kubach, Rainer Stark, Georg von Wichert, Simone Hornung, Lisa Hubrecht, Joachim Sedlmeir und Steffen Steglich: "Themenfelder Industrie 4.0". Hrsg. von Forschungsbeirat der Plattform Industrie 4.0. 2019. URL: https://www.acatech.de/publikation/themenfelder-industrie-4-0/download-pdf?lang=de, besucht am 02. 12. 2019,

[Hit+12@] Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider und Sebastian Rudolph: "OWL 2 Web Ontology Language Primer (Second Edition)". Hrsg. von W3C - World Wide Web Consortium. 2012. URL: https://www.w3.org/TR/owl2-primer/.

[Hor+04@] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof und Mike Dean: "SWRL: A Semantic Web Rule Language Combining OWL and RuleML". 2004. URL: `https://www.w3.org/Submission/SWRL`, besucht am 19. 02. 2020,

[Köc20@] Aljosha Köcher: "Semantic Machine Capability and Skill Model". 2020. URL: `https://github.com/aljoshakoecher/machine-skill-model`, besucht am 20. 12. 2020,

[Leu06@] Sandro Leuchter: "Entwicklungsprozess von Ontologien". 2006. URL: `https://www.iosb.fraunhofer.de/servlet/is/2369/visIT_02_06.pdf`, besucht am 14. 08. 2019,

[Man19@] Manchester University, Hrsg.: "OWL API". 2019. URL: `https://github.com/owlcs/owlapi`, besucht am 05. 11. 2019,

[Mcva04@] Deborah McGuinness und Frank van Harmelen: "OWL Web Ontology Language: Overview". 2004. URL: `https://www.w3.org/TR/owl-features/`, besucht am 05. 11. 2019,

[Nab19@] Hamied Nabizada: "FPB.js: Offenes, webbasiertes Werkzeug für die Formalisierte Prozessbeschreibung nach VDI 3682". 2019. URL: `https://github.com/HamiedNabizada/FPB.JS`.

[NoMc11@] Natalya Noy und Deborah McGuinness: "Ontology Development 101: A Guide to Creating Your First Ontology". 2011. URL: `http://protege.stanford.edu/publications/ontology_development/ontology101.pdf`, besucht am 12. 12. 2019,

[Obj14@] Object Management Group: "Object Constraint Language". 2.2014. URL: `https://www.omg.org/spec/OCL/2.4/PDF`, besucht am 05. 11. 2019,

[Obj16@] Object Management Group: "OMG Meta Object Facility (MOF) Core Specification". 11.2016. URL: `https://www.omg.org/spec/MOF/2.5.1/PDF`, besucht am 29. 12. 2020,

[Obj17@] Object Management Group: "Unified Modeling Language". 12.2017. URL: `https://www.omg.org/spec/UML/2.5.1/PDF`, besucht am 05. 11. 2019,

[Obj18@] Object Management Group: "Systems Modeling Language". 10.2018. URL: `https://www.omg.org/spec/SysML/1.6/Beta1/PDF`, besucht am 05. 11. 2019,

[Ont19@] Ontotext: "GraphDB-Free". 2019. URL: `http://graphdb.ontotext.com/documentation/free/`, besucht am 05. 12. 2019,

[Ope20@] OpenJS Foundation: "Node.js". 2020. URL: `https://nodejs.org/en/`, besucht am 29. 12. 2020,

[Pet+12@] David Peterson, Shudi Gao, Ashok Malhotra, C. M. Sperberg-McQueen und Henry S. Thompson: "XML Schema Definition Language (XSD)". 2012. URL: `https://www.w3.org/TR/xmlschema11-2/`, besucht am 12. 03. 2020,

[Pla16@] Plattform Industrie 4.0: "Fortschreibung der Anwendungsszenarien der Plattform Industrie 4.0". Hrsg. von Bundesministerium für Wirtschaft und Energie. 2016. URL: http://www.plattform-i40.de/I40/Redaktion/DE/Downloads/Publikation/fortschreibung-anwendungsszenarien.pdf?__blob=publicationFile&v=4, besucht am 20. 11. 2016,

[Rüß+15@] Michael Rüßmann, Markus Lorenz, Philipp Gerbert, Manuela Waldner, Jan Justus, Pascal Engel und Michael Harnisch: "Industry 4.0". Hrsg. von Boston Consulting Group. 2015. URL: https://www.bcg.com/de-de/publications/2015/engineered_products_project_business_industry_4_future_productivity_growth_manufacturing_industries.aspx, besucht am 04. 11. 2019,

[Sch+12@] Michael Schneider, Jeremy Carroll, Ivan Herman und Peter F. Patel-Schneider: "OWL 2 Web Ontology Language RDF-Based Semantics (Second Edition)". 2012. URL: https://www.w3.org/TR/2012/REC-owl2-rdf-based-semantics-20121211, besucht am 22. 01. 2020,

[Spa18@] Sparx Systems, Hrsg.: "Enterpriese Architect". 2018. URL: https://www.sparxsystems.de/, besucht am 05. 11. 2019,

[Sta19@] Stanford University, Hrsg.: "Protégé". 2019. URL: https://protege.stanford.edu/, besucht am 05. 11. 2019,

[Sta20@] Stardog: "Virtual Graph Mappings". 2020. URL: https://www.stardog.com/blog/virtual-graphs-relational-data-in-stardog/, besucht am 17. 02. 2020,

[Top19@] TopQuadrant, Hrsg.: "TopBraidComposer". 2019. URL: https://www.topquadrant.com/products/topbraid-composer/, besucht am 05. 11. 2019,

[W3C08@] W3C SPARQL Working Group: "SPARQL Query Language for RDF". Hrsg. von W3C - World Wide Web Consortium. 2008. URL: https://www.w3.org/TR/sparql11-overview/.

[W3C11@] W3C - World Wide Web Consortium: "Turtle - Terse RDF Triple Language". 2011. URL: https://www.w3.org/TeamSubmission/turtle/, besucht am 18. 10. 2019,

[W3C12@a] W3C - World Wide Web Consortium: "OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax (Second Edition)". 2012. URL: https://www.w3.org/TR/owl2-syntax/, besucht am 18. 10. 2019,

[W3C12@b] W3C - World Wide Web Consortium: "R2RML: RDB to RDF Mapping Language". 2012. URL: http://www.w3.org/TR/2012/REC-r2rml-20120927/, besucht am 12. 12. 2019,

[Wal+19@] Thomas Walloschke, Bernd Waser, Jörg Wende, Mathias Wiegand und Constantin Ziesche: "Details of the Asset Administration Shell". Hrsg. von Bundesministerium für Wirtschaft und Energie. 2019. URL: https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/2018-verwaltungsschale-im-detail.html, besucht am 02. 12. 2019,