



**FRIEDRICH-SCHILLER-
UNIVERSITÄT
JENA**

**Die Erstellung einer Prozessontologie
zur Modellierung von Verwaltungsprozessen**

Bachelorarbeit

zur Erlangung des akademischen Grades

Bachelor of Science (B. Sc.)

Im Studiengang Informatik

Friedrich-Schiller-Universität Jena

Fakultät für Mathematik und Informatik

eingereicht von Jonas Hoyer

geboren am 03.06.2001 in Friedrichroda

Themenverantwortliche: Prof. Dr. Birgitta König-Ries,

Betreuer: Leila Feddoul,

Marianne Jana Mauch

Jena, 11. April 2023

Kurzfassung

Die Digitalisierung ist ein sehr umfangreiches, weit verbreitetes und aktuelles Thema. In dieser Arbeit beschäftigen wir uns deswegen konkret mit der Digitalisierung von Verwaltungsleistungen. Im Rahmen dieser Arbeit soll dafür eine Ontologie als Wissensbasis für Verwaltungsprozesse erstellt werden. Um dieses Ziel zu erreichen, wird eine Methodologie, die den Erstellungsprozess der Ontologie untergliedert, ausgewählt. Dabei wird ein konzeptionelles Modell der Ontologie erstellt, das anschließend digital umgesetzt wird. Um die Qualität der Ontologie nachzuweisen, wird diese auf häufig auftretende Fehler bei der Modellierung untersucht und ihre Fähigkeit, benötigte Informationen auszugeben, nachgewiesen.

Inhaltsverzeichnis

1. Einleitung	8
1.1. Motivation	8
1.2. Zielsetzung	8
2. Grundlagen	10
2.1. Semantic Web	10
2.1.1. Extensible Markup Language	10
2.1.2. Uniform- und Internationalized Resource Identifier	12
2.1.3. Resource Description Framework	13
2.1.4. Resource Description Framework Scheme	14
2.1.5. Web Ontology Language	14
2.1.6. SPARQL Protocol And RDF Query Language	15
2.2. Ontologie	16
2.2.1. Bestandteile	16
2.2.2. Methoden zur Erstellung von Ontologien	17
2.2.3. Methoden zur Evaluation von Ontologien	24
3. Verwandte Arbeiten	25
3.1. Ontologien basierend auf BPMN	25
3.2. Ontologien im Anwendungsbereich öffentlicher Verwaltungen	27
4. Konzept	28
4.1. Zielsetzung	28
4.2. Analyse wiederverwendbarer Ontologien	29
4.2.1. Recherche	29
4.2.2. Kriterien	30
4.2.3. Bewertung	30
4.3. Entwurf	33
5. Implementierung	39
5.1. Klassenhierarchie	39
5.2. Object Properties	39
5.3. Data Properties	42
5.4. Individuen	44
6. Evaluation	45
6.1. SPARQL-Anfragen	45
6.2. Ontology Pitfall Scanner!	51

7. Fazit	54
8. Ausblick	55
Literaturverzeichnis	56
Abbildungsverzeichnis	60
Tabellenverzeichnis	61
Selbstständigkeitserklärung	62

Abkürzungsverzeichnis

ASCII American Standard Code for Information Interchange

BBO BPMN 2.0 Based Ontology for Business Process Representation

BPD Business Process Diagram

BPMN Business Process Modeling Notation

BPMNO Business Process Modeling Notation Ontology

BPMN 2.0 Business Process Modeling Notation 2.0

BPMO Business Process Modelling Ontology

CLeAR Conducting Literature Search for Artifact Reuse

CDPs Content Ontology Design Patterns

DAML DARPA Agent Markup Language

DTD Dokumenttyp-Definition

DDPO DOLCE + DnS Plan Ontology

FIM Förderales Informationsmanagement

GFO General Formal Ontology

HTTP Hypertext Transfer Protocol

ISBN Internationale Standardbuchnummer

IRI Internationalized Resource Identifier

LeiKa Leistungskatalog

m3po Multi Metamodel Process Ontology

NORs nicht-ontologische Ressourcen

OZG Onlinezugangsgesetz

OdV Ontologie des Verwaltungshandelns

ODPs Ontology Design Patterns

OIL Ontology Inference Layer Language

OOPS! OntOlogy Pitfall Scanner!

ORSD Ontology Requirements Specification Document

OWL Web Ontology Language

PSL Process Specification Language

RAG Referenzaktivitätengruppe

RDF Resource Description Framework

RDFS Resource Description Framework Scheme

SUPER Semantics Utilised for Process Management within/between Enterprises

SPARQL SPARQL Protocol And RDF Query Language

URI Uniform Resource Identifier

URL Uniform Resource Locator

URN Uniform Resource Name

WSML Web Service Modeling Language

W3C World Wide Web Konsortium

XML Extensible Markup Language

1. Einleitung

Das Gesetz zur Verbesserung des Onlinezugangsgesetz (OZG)[1] von Verwaltungsleistungen sieht vor, dass Bund, Länder und Kommunen in naher Zukunft ihre Verwaltungsleistungen in digitaler Form anbieten.

1.1. Motivation

In diesem Zusammenhang untersuchen die Projekte Canarèno und simpLEX [2] Methoden zur Unterstützung und Beschleunigung der Digitalisierung dieser Leistungen. Dabei liegt bei Canarèno der Fokus auf der computerunterstützten Analyse elektronisch verfügbarer Rechtsnormen und simpLEX beschäftigt sich vorwiegend mit der Vereinfachung der Erstellung und Verarbeitung elektronischer Dokumente durch Zuhilfenahme maschinenlesbarer Normentexte und Dokumentbausteine. Jede Verwaltungsleistung basiert auf einem Prozess, bestehend aus einer Reihe von Schritten, die befolgt werden müssen. Bei der Erstellung solcher Prozesse müssen verschiedene Rechtsnormen in Betracht gezogen werden. In diesem Zusammenhang werden also relevante Rechtsnormen identifiziert und analysiert, um spezifische Hinweise über die erforderlichen Prozessschritte sowie die Beteiligten zu ermitteln. Allerdings sind die wichtigen Begriffe im Zusammenhang mit Verwaltungsprozessen weder spezifiziert noch formalisiert. Dieser Mangel an eindeutiger semantischer Beschreibung führt zur Konfusion der verschiedenen Begriffe, was die Interoperabilität und Datenintegrität zwischen den Systemen einschränkt. Daraus resultiert die Notwendigkeit eines Modells, das die Domänenkonzepte und ihre Beziehungen eindeutig und formal beschreibt. Eine Ontologie stellt ein solches Modell dar. Sie beinhaltet eine Reihe von Konzepten und Beziehungen einer spezifischen Problemdomäne und steigert die Interoperabilität und Kommunikation zwischen den Systemen. Außerdem trägt eine Ontologie zur Verbesserung der semantischen Suche und zur Bereitstellung eines kontrollierten Vokabulars zur Durchführung semantischer Annotationen, zum Beispiel von Texten, zur Unterstützung der automatischen Analyse, bei [3][4].

1.2. Zielsetzung

Das Ziel dieser Arbeit ist die Implementation einer prozessorientierten Ontologie im Bereich der öffentlichen Verwaltung als Wissensbasis für Verwaltungsprozesse. Zur Erreichung dieses Ziels werden für Verwaltungsprozesse relevante Begriffe modelliert, bereits bestehende Ontologien bewertet und gegebenenfalls wiederverwendet. Die

Ontologie wird anschließend anhand von Beispielen mit konkreten Entitäten und Eigenschaften populiert. Als Qualitätsnachweis werden die Anforderungen an die Ontologie in Form aufgestellter Kompetenzfragen nach Fertigstellung mit Hilfe von SPARQL-Anfragen evaluiert. Die Ontologie wird während der Implementierung mit Hilfe des Ontology Pitfall Scanner! (OOPS!)[5] validiert. Um diese Ziele zu erreichen, werden wir folgende Forschungsfragen untersuchen.

1. Was ist der Umfang und Fokus der Ontologie? Welche Kompetenzfragen soll die Ontologie beantworten können?
2. Welche Methodologie liegt der Erstellung und Implementierung der Ontologie zugrunde?
3. Welche Ontologien können wiederverwendet werden?
4. Welche Konzepte und Beziehungen müssen modelliert werden?
5. Wie wird die Qualität der Ontologie bewertet?

Für die Beantwortung dieser Forschungsfragen, haben wir die Arbeit wie folgt gegliedert. In Kapitel 2 werden wichtige Begriffe und Sachverhalte definiert und erläutert. Kapitel 3 leitet zu unserem Anwendungsgebiet verwandte Arbeiten ein. In Kapitel 4 wird das konzeptionelle Modell der Ontologie erstellt. Kapitel 5 beschäftigt sich mit der digitalen Umsetzung des konzeptionellen Modells und in Kapitel 6 wird die Ontologie evaluiert. Abschließend ziehen wir ein Fazit und geben einen Ausblick für zukünftige Ergänzungen der Ontologie.

2. Grundlagen

Dieses Kapitel dient der Definition und Erläuterung grundlegender Begriffe. Dabei werden Semantic Web Technologien thematisiert, Ontologien allgemein definiert und ein Überblick über mögliche Erstellungsmethoden gegeben.

2.1. Semantic Web

Durch das Web wird eine enorme Datenmenge für jeden Anwendungsbereich zur Verfügung gestellt. Ein Mensch kann von dieser Informationsquelle profitieren, da er sein Grund- und Kontextwissen mit Inhalten eines bestimmten Themengebiets verknüpfen und in Beziehung setzen kann. Maschinen hingegen sind in der Lage, ausschließlich die Informationen zu verarbeiten, die ihnen zur Verfügung gestellt werden und für die durch den Mensch aufgestellte Regeln gelten. Mit Hilfe des Semantic Webs können jedoch auch Computerprogramme, durch grundlegende Algorithmen Daten in Beziehung setzen und deren Bedeutung bestimmen, wodurch das Semantic Web eins der größten Anwendungsgebiete für Ontologien darstellt. Dass es sich bei den Informationen um strukturierte Daten handelt, ist dabei von großer Bedeutung, um die Maschinenlesbarkeit zu gewährleisten. Dadurch sind Programme in der Lage, eine eindeutige Unterscheidung zwischen dem Objekt "Ball" als Spielobjekt für Ballsportarten und dem Objekt "Ball" als Tanzveranstaltung zu treffen. Die Architektur des Semantic Web ist anschaulich in Abbildung 2.1 dargestellt [6].

Jede einzelne dieser Schichten beschreibt dabei eine wichtige Technologie des Semantic Web und ist in der Lage, die Fähigkeiten darunterliegender Schichten für sich zu nutzen. Während die Technologien des Stacks von unten nach oben bis einschließlich Web Ontology Language (OWL) standardisiert und allgemein akzeptiert sind, muss der obere Teil noch implementiert und umgesetzt werden. In den folgenden Unterabschnitten werden die wichtigsten Technologien des Semantic Web spezifischer thematisiert [6].

2.1.1. Extensible Markup Language

Extensible Markup Language (XML) ist eine vom World Wide Web Konsortium (W3C) entwickelte Meta- und Markup-Sprache. Wie der Name schon suggeriert, ist XML erweiterbar. Man ist also in der Lage, mit Hilfe von XML eigene Markups für Dokumente und damit neue Markup-Sprachen zu definieren. Das bedeutet,

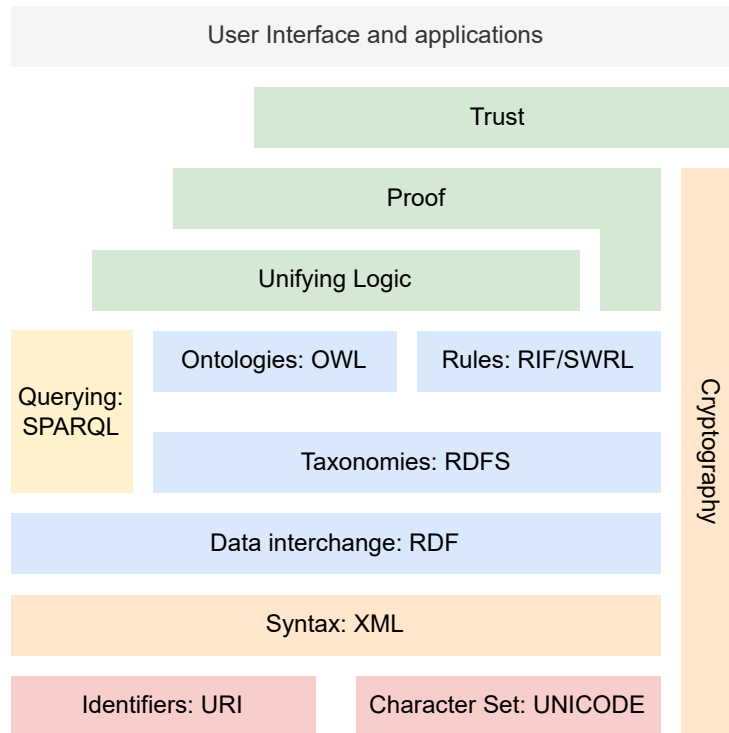


Abbildung 2.1.: Semantic Web Stack [7]

dass XML nicht mit vorgegebenen Befehlen arbeitet, sondern vielmehr der Dokumentersteller selbst die Regeln und Befehle für sein XML-Dokument vorgibt. Durch XML ist es beispielsweise möglich, bestimmten Bereichen in Texten zusätzliche Informationen hinzuzufügen. So lässt sich das Dokument beispielsweise hierarchisch strukturieren, wodurch die Lesbarkeit für Mensch und Maschine verbessert wird [8].

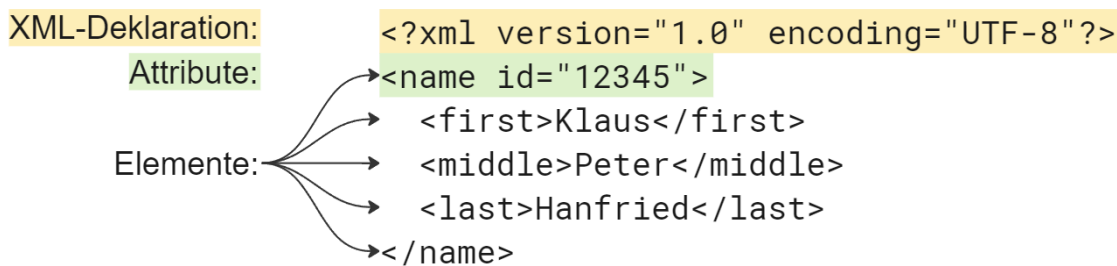


Abbildung 2.2.: XML-Dokument

XML-Dokumente bestehen aus Textzeichen auf Basis des Unicode-Zeichensatzes und starten mit einer XML-Deklaration, damit der Prozessor das Dokument korrekt als XML-Dokument identifizieren kann [8]. Ein XML-Dokument ist beispielhaft in Abbildung 2.2 aufgeführt. Die XML-Deklaration besteht dabei aus drei verschiedenen Attributen, bei denen die Reihenfolge essentiell ist. Das erste Attribut ist *version*, wodurch die Versionsnummer der zugrundeliegenden XML-Spezifikation definiert wird. Dieser Parameter ist obligatorisch und muss Bestandteil jeder XML-

Deklaration sein. Das zweite Attribut ist *encoding*, bei dem es sich um ein optionales Attribut handelt, das die Kodierung der Datei bestimmt. Wird dieser Parameter ausgelassen, so wird die Kodierung in Unicode angenommen. Das letzte Attribut ist *standalone*, ebenfalls optional und empfängt als gültige Werte “yes” oder “no”. Wenn dieser Parameter “no” als Attributwert annimmt, so wird eine externe Dokumenttyp-Definition (DTD) referenziert. Eine DTD beinhaltet dabei eine formale Syntax, die für den Aufbau von Dokumenten benötigt wird. Wenn jedoch “yes” als Attributwert angenommen wird, so wird entweder keine DTD, oder eine interne DTD verwendet. Als optionales Attribut kann *standalone* ebenfalls ausgelassen werden, wodurch es standardmäßig “no” als Attributwert annimmt [9].

Die Grundbausteine eines XML-Dokumentes sind die sogenannten Elemente, um eine vernünftige Struktur des Textes zu gewährleisten. Ein Element startet dabei mit einem Anfangs-Tag und endet mit einem Ende-Tag. Dabei ist wichtig zu beachten, dass jeder Anfangs-Tag ein dazugehöriges Ende-Tag haben muss, um Syntaxfehler zu vermeiden. Den Inhalt bilden dann die Textzeichen zwischen diesen beiden Tags eines Elementes, was man auch sehr gut in Abbildung 2.2 erkennen kann [9].

2.1.2. Uniform- und Internationalized Resource Identifier

Der Uniform Resource Identifier (URI) beschreibt, wie der Name schon suggeriert, einen Indikator und sorgt für die eindeutige Identifikation einer abstrakten oder physischen Ressource. Stellt man sich beispielsweise das Internet als Raum vor, der mit Informationen an vielen verschiedenen Punkten populierte ist, so ist es mit Hilfe von URIs möglich, jeden dieser Punkte und die entsprechende Information eindeutig zu identifizieren. Als konkrete Zeichenfolge des American Standard Code for Information Interchange (ASCII)-Zeichensatzes sind URIs dabei meist in digitale Dokumente integriert. Die beiden bekanntesten Teilmengen von URIs sind der Uniform Resource Locator (URL) und der Uniform Resource Name (URN). Während URIs Ressourcen identifizieren, ohne den Zugriff garantieren zu können, identifiziert ein URL eine Ressource unter Angabe der Zugriffsmöglichkeit und Lokation. Dadurch ist sie ortsabhängig und muss sich bei eventueller Änderung der Lokation der entsprechenden Ressource anpassen. Ein URN hingegen identifiziert eine Ressource dauerhaft und unabhängig von ihrer Lokation. URNs werden also beispielsweise für die Identifikation von Büchern verwendet, aufgrund der eindeutigen Identifizierung durch die Internationale Standardbuchnummer (ISBN) und des dauerhaften Ortswechsels. Der Aufbau einer URI ist anschaulich in Abbildung 2.3 dargestellt [10].

"scheme" : // "authority" "path" ? "query" # "fragment "

Abbildung 2.3.: Allgemeine Syntax einer URI

Das erste Element der Syntax ist *scheme* und wird vom Rest der URI durch einen Doppelpunkt separiert. *Scheme* beinhaltet die essentielle Information über das verwendete Protokoll. Das bekannteste Protokoll ist dabei das Hypertext Transfer Protocol (HTTP). Die Domäne wird durch die optionale *authority* beschrieben und

besteht entweder aus einem registrierten Name oder einer IP-Adresse. Mit Hilfe des *paths* wird der entsprechende Pfad der Ressource spezifiziert. Ausschließlich *scheme* und *path* müssen in jeder URI auftreten. Sollte die genaue Ortsangabe einer Ressource durch den *path* nicht möglich sein, so werden weitere Informationen zur Identifizierung der entsprechenden Ressource benötigt. Diese Informationen befinden sich in der *query* und müssen durch eine Abfrage auf die durch den *path* angegebene Quelle abgerufen werden. Die *query* wird dabei vom *path* durch ein Fragezeichen getrennt. Das Ende einer URI bildet das optionale *fragment*, das durch ein Doppelkreuz eingeleitet wird und die Referenzierung einer bestimmten Stelle innerhalb einer Ressource ermöglicht [10].

Da die Zeichenmenge vom URI durch den ASCII-Zeichensatz begrenzt ist, gilt der Internationalized Resource Identifier (IRI) als Ergänzung durch seinen vergrößerten Zeichensatz, bei dem es sich um den Universal Coded Character Set handelt. IRIs sollen also URIs bei der Identifizierung von Ressourcen ersetzen, die diesen Zeichensatz unterstützen. Diese Ergänzung ist von großer Bedeutung, da URIs Informationen aller Art, die nicht durch den ASCII-Zeichensatz darstellbar sind, enthalten können [11].

2.1.3. Resource Description Framework

Das Resource Description Framework (RDF) ist ein vom W3C standardisiertes Framework für die Beschreibung und den Austausch strukturierter Informationen, was bei Ontologien sehr häufig Anwendung findet. Für die Beschreibung strukturierter Informationen enthält das RDF die entsprechende Syntax und Semantik, die sich, genau wie ein simpler deutscher Satz, aus Subjekt, Prädikat und Objekt zusammensetzen [12]. In Abbildung 2.4 ist ein vereinfachtes graphbasiertes Datenmodell dargestellt. Zum einen enthalten solche Datenmodelle Knoten, die für die Darstellung von Subjekten und Objekten zuständig sind und zum anderen Kanten, die eine Verbindung zwischen Subjekt und Objekt schaffen und somit die Prädikate bilden. Im Beispiel ist also das Subjekt *Peter* über das Prädikat *studiert* mit dem Objekt *Informatik* verbunden. Dadurch wurde der simple Satz *Peter studiert Informatik* in ein graphbasiertes Datenmodell übersetzt. Beide Knoten bilden dabei Individuen, die mit Hilfe des URIs eindeutig identifizierbar sind. Eine Ausnahme bilden dabei Literale, bei denen es sich um konkrete Werte handelt und die damit nicht eindeutig identifizierbar sind. Im Beispiel ist dafür das Subjekt *Peter* über das Prädikat *heißt* mit dem Literal *Peter* verbunden. Das Literal nimmt also im Satz die Stelle des Objekts ein, ohne einen bestimmte URI zu haben, was es uns möglich macht, durch konkrete Werte die Eigenschaften von Knoten zu spezifizieren. Es existieren jedoch auch Knoten, für die weder ein URI, noch ein Literal angegeben ist. Bei dieser Art von Knoten handelt es sich um sogenannte *blank nodes*, die anonyme Ressourcen repräsentieren.

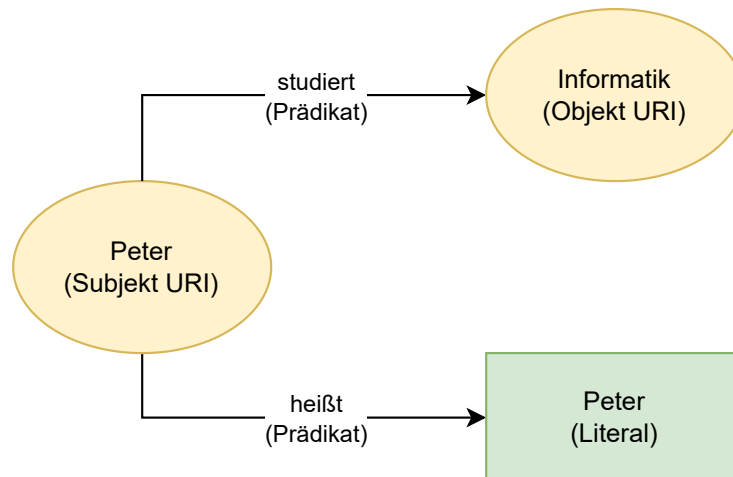


Abbildung 2.4.: Beispiel für ein grafisches Datenmodell in RDF

2.1.4. Resource Description Framework Scheme

Damit im RDF formulierte Aussagen interpretiert werden können, wird ein Vokabular für die Typisierung dieser RDF-Ressourcen benötigt, wodurch die Entwicklung maschineninterpretierbarer Ontologien ermöglicht wird. Ein solches Vokabular wird durch das Resource Description Framework Scheme (RDFS) zur Verfügung gestellt. Dadurch liefert das RDFS die Repräsentation der RDF-Ressourcen, deren Eigenschaften und Relationen untereinander und stellt die Anordnung existierender Klassen mittels Klassenhierarchie zur Verfügung. Eine Klasse fasst also eine Menge von RDF-Instanzen zusammen, die jeweils ein konkretes Individuum repräsentieren. Damit gilt das RDFS als Erweiterung von RDF, die ebenfalls durch das W3C standardisiert wurde [13].

2.1.5. Web Ontology Language

Die OWL ist eine durch das W3C standardisierte, logikbasierte Sprache für die Repräsentation komplexer Informationen über Klassen, Eigenschaften und deren Beziehungen. OWL wurde aus der Kombination von DARPA Agent Markup Language (DAML) und Ontology Inference Layer Language (OIL) entwickelt und 2006 dann erfolgreich durch das W3C standardisiert. Als gängiges Dateiformat für Ontologien wird OWL häufig zur Beschreibung dieser verwendet. OWL beinhaltet dabei die drei verschiedenen Subsprachen *OWL Full*, *OWL DL* und *OWL Lite*. Jede dieser Subsprachen schließt jeweils einen anderen Kompromiss zwischen Ausdruckskraft und Berechenbarkeit, wodurch jeder Entwickler die Entscheidung treffen muss, welche Subsprache den eigenen Anforderungen und Bedürfnissen am besten gerecht wird. Die Eigenschaften der Subsprachen sind anschaulich in Tabelle 2.1 zusammengefasst [14].

OWL Full	OWL DL	OWL Lite
beinhaltet alle OWL Sprachkonstrukte, ohne Einschränkungen	beinhaltet alle OWL Sprachkonstrukte, mit gewissen Einschränkungen	beinhaltet weniger Sprachkonstrukte als OWL DL
enthält OWL DL, OWL Lite	Teilsprache von OWL Full, enthält OWL Lite	Teilsprache von OWL Full, OWL DL
nicht entscheidbar	entscheidbar	entscheidbar
hohe Ausdruckskraft	hohe Ausdruckskraft, unter Beibehaltung rechnerischer Vollständigkeit	niedrige Ausdruckskraft
Berechenbarkeit der Schlussfolgerung nicht garantiert	Berechenbarkeit aller Schlussfolgerungen garantiert	einfache Klassenhierarchien

Tabelle 2.1.: OWL-Subklassen [14]

2.1.6. SPARQL Protocol And RDF Query Language

SPARQL Protocol And RDF Query Language (SPARQL) ist eine durch das W3C standardisierte graphbasierte Abfragesprache für RDF-Inhalte und Datenbanken. Sie ist also in der Lage, Abfragen auf große Datenmengen, die Informationen verschiedener Formate und Quellen beinhalten, auszuführen. Das Resultat ist meist die entsprechende Ergebnismenge, oder ein konkretes RDF-Diagramm. Es existieren vier unterschiedliche Arten von Abfragen [15].

1. Mit Hilfe der Abfrage *ASK* wird überprüft, ob mindestens eine Übereinstimmung in den RDF-Daten vorliegt.
2. Durch die Abfrage *SELECT* werden entweder alle oder einige Übereinstimmungen mit den RDF-Daten in Form einer Tabelle angegeben.
3. Die Abfrage *CONSTRUCT* erstellt einen RDF-Graph, indem Tripel für die Variablen der Übereinstimmungen erstellt werden.
4. Mit Hilfe der Abfrage *DESCRIBE* werden die entsprechenden Übereinstimmungen mit den RDF-Daten durch die Erstellung eines RDF-Diagramms beschrieben.

Wenn man beispielsweise von einer Schulklasse ausgeht, dann werden die Schüler durch den Lehrer, nennen wir ihn Max, unterrichtet. Dadurch lässt sich ein einfacher deutscher Satz bilden. Dabei ist das Subjekt *Max* über das Prädikat *unterrichtet* mit dem Objekt *Schüler* verbunden. Mit Hilfe einer SPARQL-Abfrage ist es nun möglich, alle Individuen der Klasse *Schüler* anzugeben, die über die Beziehung *unterrichtet* mit dem Subjekt *Max* verbunden sind. Das Ergebnis ist dann eine Liste der Namen dieser Schüler. SPARQL erlaubt also den Zugriff auf die einzelnen Informationen der RDF-Daten und deren Darstellung.

2.2. Ontologie

In der Informatik ist es in vielen Anwendungsbereichen essentiell, Informationen zu repräsentieren und vorhandenes Wissen über Fakten und Regeln auszutauschen, um eine gemeinsame Grundlage zu schaffen. Menschen sind in der Lage ihr vorhandenes Grund- und Kontextwissen zu nutzen, was sie sich mit Hilfe von verfügbaren Ressourcen angeeignet haben und dieses mit Inhalten eines bestimmten Themengebiets zu verknüpfen. Wenn jedoch beispielsweise ein Automat, in Bezug auf gespeichertes Wissen, Informationen austauschen, oder bestimmte Daten selektieren und interpretieren soll, so benötigt er eine geeignete Repräsentation der Begriffe und ihrer Zusammenhänge - eine Ontologie. Eine Ontologie ist also eine Wissensbasis, die eine konkrete Terminologie und die Zusammenhänge der dort definierten Begriffe eines bestimmten Gegenstandsbereiches beinhaltet. Anwendung finden Ontologien also in der Informatik fast überall, vor allem bei der Kommunikation zwischen Systemen, oder bei der Repräsentation und Wiederverwendung von Wissen [16].

2.2.1. Bestandteile

Eine Ontologie hat dabei die Aufgabe, bestimmte Prozesse spezieller Anwendungsbereiche der realen Welt in eine für den Computer geeignete Darstellungsform umzuwandeln, damit sie den entsprechenden Vorgang regelkonform abbilden kann. Dabei wird mit Hilfe der folgenden grundlegenden Konzepte ein umfassendes Begriffssystem gebildet [17].

Klassen: Eine Klasse ist als abstrakte Beschreibung gemeinsamer Eigenschaften definiert und fasst somit Objekte zusammen, die dieselbe Eigenschaft aufweisen. Mehrere Klassen bilden dabei eine Klassenhierarchie, in der eine Vielzahl von Unterklassen zu einer Oberklasse zusammengefasst werden kann, was die Lesbarkeit verbessert. Dabei wird durch Vererbung sichergestellt, dass für die Unterklassen die definierten Eigenschaften der Oberklasse ebenfalls gelten [18].

Instanzen: Nachdem die Ontologie mit Instanzen populiert wurde, bilden diese die eigentliche Wissensrepräsentation der Ontologie. Dabei werden die Instanzen einer konkreten Klasse zugeordnet und durch entsprechende Eigenschaften und Relationen in Verbindung gesetzt [19].

Relationen: Relationen stellen Verbindungen zwischen zwei oder mehreren Klassen dar. Dabei wird zwischen hierarchischen und nicht-hierarchischen Relationen unterschieden. Bei hierarchischen Relationen werden die einzelnen Klassen durch eine konkrete Taxonomie in Verbindung gesetzt. Nicht-hierarchische Relationen hingegen setzen die Klassen ohne entsprechende Taxonomie in Beziehung. Zudem lassen sich Aussagen über die universelle Notwendigkeit, oder die Optionalität einer Relation treffen. Wenn eine Relation universell notwendig ist, dann muss diese Relation für jede Instanz der entsprechenden Klasse gelten. Optionale Relationen hingegen haben die Möglichkeit zu gelten, wobei sie nicht essentiell für jede Instanz der entsprechenden Klasse sind [20].

Axiome: Bei Axiomen handelt es sich um immer wahre Aussagen innerhalb der Ontologie. Dabei definieren Axiome bestimmte Bedingungen für Klassen und Relationen, mit dem Zweck, diese Konzepte näher zu spezifizieren. Wie bereits erwähnt, lässt sich für Relationen eine Aussage über deren universelle Notwendigkeit, oder Optionalität treffen. Das ist nur möglich, wenn die Relation durch das entsprechende Axiom näher spezifiziert wird. Durch Axiome werden also Widersprüche innerhalb der Ontologie vermieden und die Konsistenz der Ontologie wird sichergestellt [18].

2.2.2. Methoden zur Erstellung von Ontologien

Bei der Entwicklung von Ontologien ist es von Vorteil, den Ablauf durch eine konkrete Schrittfolge zu gliedern, die dabei hilft, die eigenen Vorstellungen und Ziele bestmöglich umzusetzen. Eine Methodologie liefert dabei den Grundbaustein und sorgt dafür, den Überblick über die eigenen Ziele und den aktuellen Stand nicht zu verlieren. Aufgrund der Vielzahl vorhandener Methodologien ist es enorm wichtig, die gegebenen Möglichkeiten zu analysieren und dementsprechend die optimale Auswahl zu treffen. In den folgenden Unterkapiteln werden bestimmte Methodologien ausführlicher thematisiert [21].

Ontology Development 101

Ontology Development 101 [22] beschreibt eine konkrete Anleitung für die Erstellung von Ontologien. Der Ablauf ist dabei in sieben Einzelschritte unterteilt. Im ersten Schritt werden, unter Beachtung des vorliegenden Anwendungsbereiches, Anforderungen an die Ontologie gestellt. Dafür werden vor der digitalen Umsetzung Kompetenzfragen formuliert, die durch die finale Ontologie evaluiert werden müssen. Der zweite Schritt thematisiert die Betrachtung wiederverwendbarer Ontologien. Dabei werden die Konzepte und Ziele bereits bestehender Ontologien analysiert. Anschließend wird eine Entscheidung getroffen, ob die Wiederverwendung der entsprechenden Ontologie vorteilhaft ist, oder ob der entstehende Aufwand den Nutzen übersteigt. Eine erfolgreiche Analyse sorgt letztendlich für eine Steigerung der Interoperabilität. Nach dieser ausführlichen Vorarbeit folgt im dritten Schritt die Definition wichtiger Begriffe der Ontologie, um das Verständnis des Nutzers zu verbessern. Dafür eignet sich ein Entwurf, der die Konzepte und deren Beziehungen untereinander veranschaulicht und konkret beschreibt. Anschließend wird der Entwurf digital umgesetzt. Im vierten Schritt werden dabei die Klassen und die entsprechende Klassenhierarchie implementiert. Der fünfte Schritt beschreibt nachfolgend die Implementierung der Beziehungen, der sogenannten “Object Properties”, zwischen den Klassen. Da die einzelnen Konzepte und Beziehungen konkrete Werte mit unterschiedlichen Datentypen haben können, werden im sechsten Schritt die sogenannten “Data Properties” implementiert. Dadurch ist es möglich, Aussagen über die individuellen Eigenschaften von Klassen und Beziehungen und deren Kardinalität zu treffen. Um die Implementierung abzuschließen, beschreibt der siebte und letzte Schritt die Population der Ontologie mit konkreten Individuen. Dabei

wird jedes Individuum einer bestimmten Klasse zugeordnet und anschließend werden Aussagen über die entsprechenden Object- und Data Properties getroffen. Erst dadurch wird die Ontologie mit Wissen versorgt. Für die Evaluation der Ontologie wird ihre Fähigkeit getestet, die aufgestellten Kompetenzfragen durch entsprechende SPARQL-Abfragen zu beantworten.

NeOn

Die Methodologie NeOn beschreibt eine weitere mögliche Anleitung, die Erstellung einer Ontologie zu strukturieren. NeOn basiert dabei auf neun verschiedenen Szenarien, die beliebig untereinander kombiniert werden können. Diese Szenarien werden in den folgenden Unterabschnitten genauer thematisiert [23].

- Szenario 1 beschreibt die allgemeine Entwicklung von Ontologien. Dabei muss der Entwickler die benötigten Anforderungen an die Ontologie definieren und auf dieser Basis den Umfang, den Anwendungsbereich, die Sprache und die Zielgruppe festlegen. Das Ziel ist dabei, die eigenen Interessen im Ontology Requirements Specification Document (ORSD) zusammenzufassen. Anschließend folgen die Konzeptionalisierung und die technische Umsetzung der definierten Ziele. Die Wiederverwendung bereits implementierter Ontologien ist dabei optional und wird in den folgenden Szenarien näher erläutert.
- Szenario 2 thematisiert die Wiederverwendung und das Reengineering nicht-ontologischer Ressourcen (NORs). Es werden also geeignete NORs auf Basis des ORSDs ausgewählt und anschließend in die Ontologie integriert. Bei NORs handelt es sich beispielsweise um Taxonomien oder bestimmte Begriffsdefinitionen, die für die Erstellung der eigenen Ontologie von Bedeutung sein können.
- Szenario 3 beschreibt ausschließlich die Wiederverwendung ontologischer Ressourcen, auf Basis des ORSDs. Dabei gibt es drei unterschiedliche Möglichkeiten, was den Umfang der Wiederverwendung betrifft. Man kann entweder die ganze Ontologie, einen Teil der Ontologie, oder lediglich bestimmte Aussagen der Ontologie wiederverwenden. Für die Integration ausgewählter Ontologien in das eigene Projekt, liefern dann die Szenarien 4 bis 6 je nach Art der benötigten Integration konkrete Anleitungen.
- Szenario 4 thematisiert eine Möglichkeit, für die Wiederverwendung ausgewählte ontologische Ressourcen in das eigene Projekt durch Reengineering zu integrieren. Wird also beispielsweise ein anderes Dateiformat, oder eine andere Sprache für eine relevante Ressource benötigt, so muss diese Ressource vor der Integration durch Reengineering angepasst werden.
- Szenario 5 beschreibt eine weitere Möglichkeit, für die Wiederverwendung ausgewählte ontologische Ressourcen in das eigene Projekt zu integrieren. Im Gegensatz zu Szenario 4 werden dabei die relevanten Ressourcen mit dem eigenen Projekt verschmolzen. Wenn also beispielsweise mehrere Ressourcen einer Domain wiederverwendet werden sollen und sich deren Inhalte überschneiden, so

ist die Erstellung einer neuen Ressource essentiell, die entsprechende Inhalte ohne Überschneidungen kombiniert.

- Szenario 6 liefert eine Ergänzung zu Szenario 5, in dem die neue Ressource mit den kombinierten Inhalten zusätzlich durch Reengineering angepasst und anschließend integriert werden kann.
- Szenario 7 beschreibt die Wiederverwendung von Ontology Design Patterns (ODPs), bei denen es sich um kleine, übersichtliche Ontologien handelt. Das Ziel ist dabei, mit Hilfe dieser ODPs den Prozess der Modellierung zu beschleunigen und auftretende Schwierigkeiten während der Modellierung zu reduzieren. Mögliche Arten dieser ODPs werden im Folgekapitel konkret definiert.
- Szenario 8 thematisiert die Umstrukturierung ontologischer Ressourcen, die in das eigene Projekt integriert werden sollen. Dabei ist die Modularisierung der Ontologie in bestimmte Inhaltsbereiche und die Beseitigung redundanter Inhalte der Ontologie sinnvoll. Die beseitigten Inhalte können durch relevante Konzepte und Beziehungen ersetzt werden. Außerdem sollten Bereiche der Ontologie, die mehr Granularität benötigen, spezifiziert werden.
- Szenario 9 beschreibt die Verfügbarkeit der eigenen Ontologie in verschiedenen Sprachen. Um eine multilinguale Ontologie zu erhalten, müssen dabei die Konzepte in eine oder mehrere Sprachen übersetzt werden. Ein mögliches Anwendungsbeispiel ist Spanien, wo verschiedenen Regionen unterschiedliche Sprachen haben.

Entwurfsmuster

Für die Entwicklung von Ontologien erweist sich eine Analyse bereits implementierter Ontologien als hilfreich, um relevante Konzepte in das eigene Projekt zu integrieren. Häufig findet man dabei viel zu große und mangelhaft dokumentierte Ontologien, bei denen die Wiederverwendung den Nutzen übersteigt. Eine mögliche Lösung für dieses Problem liefern ODPs. Bei diesen Entwurfsmustern handelt es sich um kleine und übersichtliche Ontologien, die sich auf bestimmte Anwendungsgebiete beziehen und Lösungen für unterschiedliche Probleme bei der Modellierung liefern. Einige Beispiele für solche Entwurfsmuster werden in den folgenden Unterabschnitten erläutert [24].

Strukturelle ODPs setzen sich zusammen aus logischen ODPs und architektonischen ODPs. Logische ODPs unterstützen dabei die Problemlösung in Bezug auf die Ausdruckskraft der Ontologie. Sie dienen also der Designoptimierung, wo die entsprechende Repräsentation bestimmte logische Konstrukte nicht unterstützt. Architektonische ODPs hingegen beeinflussen die interne- und externe Form der Ontologie. Sie beschreiben also, wie die finale Ontologie aufgebaut sein soll, um zu hoher Komplexität entgegenzuwirken.

Korrespondenz ODPs beinhalten Reengineering ODPs und Mapping ODPs. Reengineering ODPs unterstützen dabei die Lösung auftretender Probleme bei der technischen Umsetzung eines konzeptionellen Modells. Sie beinhalten also eine Schrittfolge

für die Erstellung einer neuen Ontologie und dabei geltende Regeln. Mapping ODPs hingegen bieten eine Lösung für die Erstellung von Assoziationen zwischen zwei bereits implementierten Ontologien. So ist es möglich, Verbindungen zwischen beiden Ontologien zu erstellen, ohne die logischen Typen der entsprechenden Elemente zu ändern.

Presentations ODPs verbessern die Homogenität der Benennungsverfahren und erhöhen damit die Benutzerfreundlichkeit und die Lesbarkeit von Ontologien. Zudem unterstützen sie die Evaluation, Auswahl und damit auch die Wiederverwendung entsprechender Ontologien.

Content Ontology Design Patterns (CDPs) bieten als Instanzen von logischen ODPs Lösungsvorschläge für auftretende Probleme in Klassen und Beziehungen einer Ontologie. Das betrifft dabei nur den Bereich der Ontologie, der sich speziell mit Problemen bei der Modellierung befasst.

CLeAR

Conducting Literature Search for Artifact Reuse (CLeAR) beschreibt eine Methodologie für die Wiederverwendung ontologischer Ressourcen bei der Entwicklung von Ontologien. Der iterative Ablauf von CLeAR ist in drei Kreisläufe unterteilt und in Abbildung 2.5 anschaulich dargestellt [25].

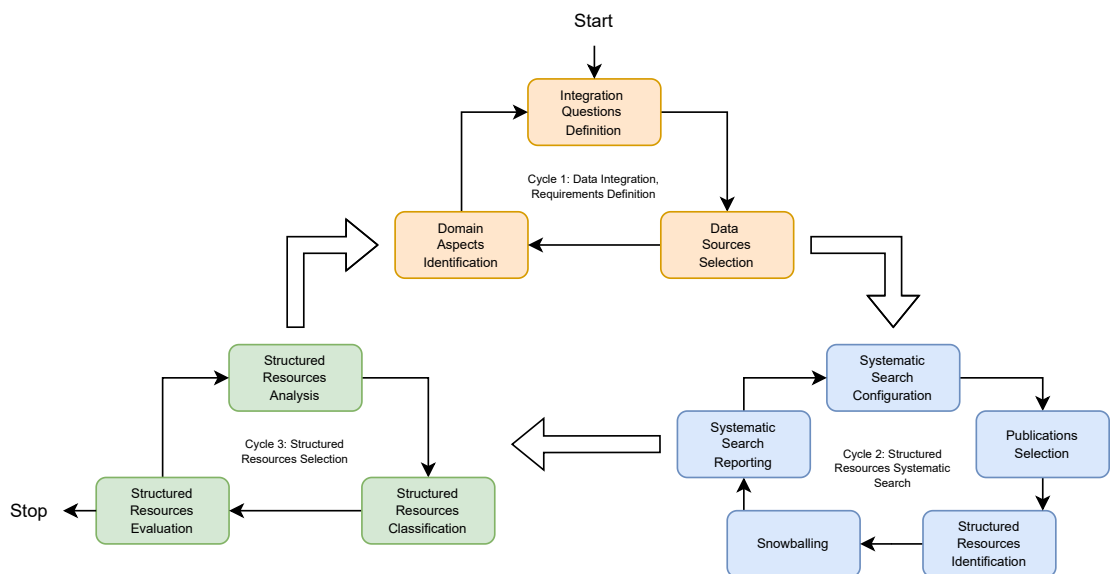


Abbildung 2.5.: Iterativer Ablauf von CLeAR

Im ersten Kreislauf werden die Anforderungen für die Integration von ontologischen Ressourcen und der Umfang der Ontologie definiert. Dabei werden bestimmte Fragen formuliert, die bei der Integration zu beachten sind. Zusätzlich werden von den entsprechenden Ontologie-Ingenieuren relevante Datenquellen zur Beantwortung dieser Fragen festgelegt und ausführlich untersucht.

Der zweite Kreislauf beschreibt die Identifikation für die Wiederverwendung relevanter ontologischer Ressourcen, auf Basis der im ersten Schritt festgelegten Anforderungen. Dabei wird eine Suchmaschine ausgewählt und Schlüsselbegriffe formuliert, die bei der Suche beliebig kombiniert werden. Zudem werden Bewertungskriterien für die Ergebnisse der Suche aufgestellt. Anschließend wird die Suche nach wissenschaftlichen Arbeiten durchgeführt und eine Auswahl getroffen. Durch die Anwendung der Bewertungskriterien werden dann die ontologischen Ressourcen der ausgewählten Veröffentlichungen analysiert. Abschließend wird durch die Evaluation der Resultate die Einhaltung der Ziele verifiziert.

Im dritten Kreislauf wird die Auswahl der analysierten ontologischen Ressourcen für die Wiederverwendung beschrieben. Dabei werden essentielle Eigenschaften der Resultate, wie eine ausführliche Dokumentation oder die Verfügbarkeit der Daten überprüft und bewertet. Anschließend werden die Resultate klassifiziert und erneut evaluiert. Dabei wird die Fähigkeit der Resultate verfügbare Daten zu repräsentieren überprüft. Das Ergebnis sind dann die für die Wiederverwendung ausgewählten ontologischen Ressourcen.

Ontology Evaluation Method

Die Ontology Evaluation Method beschreibt eine Anleitung, um für den Prozess der Wiederverwendung bereits implementierter Ontologien die am besten geeigneten auszuwählen. Dabei verfolgt diese Methodologie einen komponentenorientierten Ansatz, der die Vervollständigung verschiedener Evaluationsaufgaben vorsieht [26].

- Die *inhaltliche Bewertung* besteht dabei aus einer Analyse der durch die Ontologie modellierten Domain, um eine Entscheidung bezüglich der Relevanz für das eigene Projekt zu treffen.
- Die *Bewertung der Wissensrepräsentation* überprüft domänenunabhängig die Qualität des konzeptionellen Modells der implementierten Ontologie.
- Mit Hilfe der *technischen Bewertung* wird die Ontologie nach dem aktuellen Stand der Technik und der entsprechenden Werkzeuge eingeschätzt, um zu entscheiden, wie gut sich die implementierte Ontologie eignet, um die eigenen Anforderungen umzusetzen.
- Durch die *Bewertung der Anwendung* wird die Verwendbarkeit der implementierten Ontologie für einen konkreten Anwendungsbereich, Zweck, oder für eine konkrete Aufgabe überprüft.
- Die *Bewertung der Verfügbarkeit* liefert die Kosten und Bedingungen, die für die Nutzung der implementierten Ontologie zu beachten sind.

Für die Einhaltung der genannten Evaluationsaufgaben wird folgende Schrittfolge vorgeschlagen. Die Zusammenhänge der einzelnen Teilschritte sind anschaulich in Abbildung 2.6 dargestellt.

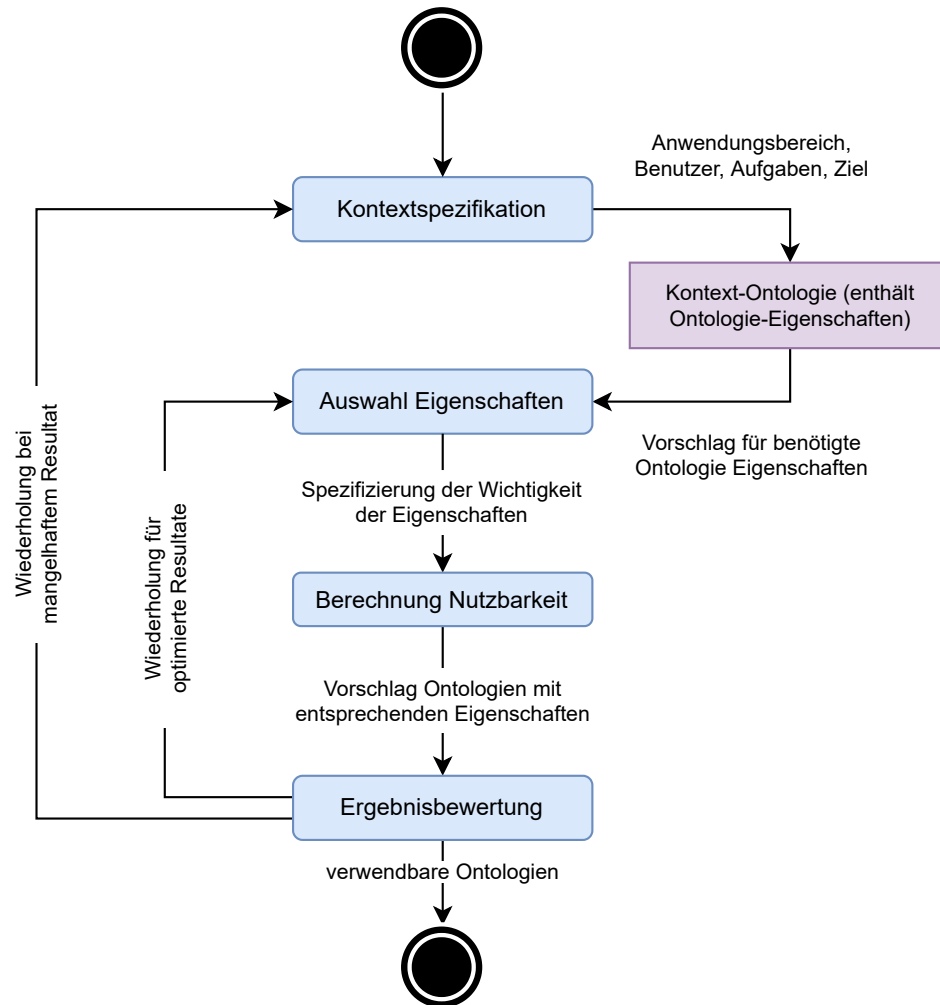


Abbildung 2.6.: Ablauf Ontology Evaluation Method

Der erste Schritt bezeichnet die *Kontextspezifikation* und dient in erster Linie dazu geeignete Bewertungskriterien für die eigene Ontologie aufzustellen. Dabei wird vor allem der entsprechende Anwendungsbereich benötigt, um für die Wiederverwendung geeignete Ontologien zu finden. Jedoch sind auch Informationen über die Teilnehmer des Evaluationsprozesses von Bedeutung, um die Kriterien der Evaluation weiter zu individualisieren. Dabei handelt es sich um *benutzerbezogene Informationen*. Man unterscheidet zusätzlich *aufgabenbezogene Informationen*, bei denen es sich um die Wichtigkeit der Wiederverwendung und Dimension der Bewertung handelt, und *umgebungsbezogene Informationen*, die das Anwendungsgebiet der finalen Ontologie und ihren Zweck beinhalten. Zusätzlich existieren *zielbezogene Informationen*, die entsprechende Ontologien mit den verfügbaren Metadaten beschreiben. Das Ziel ist es also geeignete Ontologien für die Wiederverwendung auszuwählen, unter Beachtung der beschriebenen Kontextinformationen. Für jeden Anwendungsfall gelten dabei bestimmte Kriterien, die bei der Wiederverwendung bereits bestehender Ontologien zu beachten sind, um die optimale Auswahl für die eigenen Anforderungen zu treffen. Die benötigten Eigenschaften für die einzelnen Szenarien sind anschaulich in Tabelle 2.2 dargestellt.

Anwendung	Ontologie-Eigenschaft
Integration	Validität und Konsistenz für maschinelle Verarbeitbarkeit Eindeutige Bezeichnungen für Matching Überschaubare Größe für Verständnis und Leistung Formale Repräsentation für Reasoning Einfache Struktur für Mappings
Semantische Suche/ Abruf	(Semi)formale Repräsentation für Reasoning Validität und Konsistenz für Reasoning Taxonomische Struktur für Reasoning Domainbeziehungen für Navigation und Präsentation Einheitliche Sprache für Abfrageformulierungen Manuelles Engineering für Domain-Repräsentation
Semantische Indexierung/ Annotation	Eindeutige Bezeichnungen für manuelle Indizierung Einheitliche Sprache für Dokumente Namenskonventionen für automatische Indizierung Lernfähigkeit für Ontologie
Software-Engineering	Validität und Konsistenz für maschinelle Verarbeitbarkeit Formale Repräsentation für Reasoning
Wissens-Repräsentation	Dokumentationen, Definitionen, Kommentare Eindeutige Bezeichnungen für Verständnis

Tabelle 2.2.: Ontologie-Eigenschaften für spezielle Anwendungsszenarien [26]

Durch die Kontextspezifikation werden also Ontologie Eigenschaften vorgeschlagen, die mit dem vorliegenden Kontext übereinstimmen. Dieser Prozess wird durch die Kontext-Ontologie unterstützt, die Informationen über wiederverwendbare Ontologie-Eigenschaften beinhaltet. Der nächste Schritt beschreibt anschließend die *Auswahl der Eigenschaften*, in dem der Gutachter die Wichtigkeit der vorgeschlagenen Eigenschaften festlegt.

Die ausgewählten Eigenschaften und ihre Wichtigkeit sind im nächsten Schritt die Eingaben für die *Berechnung der Nutzbarkeit*. Dabei gibt es zwei unterschiedliche Möglichkeiten. Die erste Möglichkeit beschreibt die Erweiterung der durch den Gutachter ausgewählten Eigenschaften. Dabei handelt es sich um Merkmale, die für eine sinnvolle Bewertung der Nutzbarkeit essentiell sind, jedoch in der Benutzerspezifikation als unwichtig eingestuft wurden. Die zweite Möglichkeit beschreibt die Anpassung bestimmter Eigenschaften, bei der die Gewichtungen einzelner Merkmale überarbeitet werden. Beide Möglichkeiten resultieren nach Durchführung in einer geänderten Rangfolge der vorgeschlagenen Ontologien, die das Endergebnis optimiert.

Das resultierende Endergebnis mit den vorgeschlagenen Ontologien und der entsprechenden Rangliste werden anschließend in der *Ergebnisbewertung* an den Gutachter weitergegeben. Sollte der Vorschlag von den Vorstellungen des Gutachters abweichen, kann dieser den Schritt der *Auswahl der Eigenschaften* wiederholen, um die Gewichtung der Endergebnisse zu optimieren.

2.2.3. Methoden zur Evaluation von Ontologien

Die Evaluation von Ontologien ist bei der Entwicklung ein wichtiger Bestandteil und gilt als Qualitätsnachweis des Resultats. Dadurch wird sichergestellt, dass der Inhalt der Ontologie definierte Anforderungen korrekt erfüllt und die Funktion fehlerfrei nachgewiesen wird [27]. Die meisten Ansätze für die Evaluation von Ontologien sind Teil einer der folgenden Kategorien, wobei die Ontologie als Ganzes evaluiert wird [28].

- Vergleich der Ontologie mit einem allgemein anerkannten optimalen Standard
- Anwendung der Ontologie und Überprüfung der Resultate
- Vergleich der Ontologie mit verschiedenen Quellen über die implementierte Domäne
- Vergleich der Ontologie mit vordefinierten Kriterien

Im Gegensatz zu den genannten Kategorien ist es ebenfalls möglich, die Bestandteile der Ontologie getrennt voneinander zu evaluieren. Dabei sind die folgenden Ebenen von Bedeutung.

- Bei der Datenebene liegt der Fokus auf den implementierten Konzepten und dem entsprechenden Vokabular für die Identifizierung dieser. Dadurch beinhaltet die Evaluation der Datenebene häufig den Vergleich der implementierten Informationen mit Informationen aus verschiedenen Quellen über die implementierte Domäne.
- Mit Hilfe der Evaluation der Taxonomie einer Ontologie ist es möglich, Aussagen über die Präzision definierter Beziehungen zu treffen.
- Die Evaluation der Kontextebene wird anhand der Anwendung der Ontologie in einem bestimmten Szenario durchgeführt.
- Durch die Evaluation der syntaktischen Ebene der Ontologie wird sichergestellt, dass Wiederholungen und somit redundante Informationen vermieden werden.
- Die Evaluation der Struktur der Ontologie kann durch die Anwendung vordefinierter Kriterien durchgeführt werden.

3. Verwandte Arbeiten

In diesem Kapitel werden verschiedene wissenschaftliche Arbeiten thematisiert, die sich ebenfalls mit der Ontologieentwicklung beschäftigen. Da das Ziel dieser Arbeit die Entwicklung einer Prozessontologie im Anwendungsbereich öffentlicher Verwaltungen ist, wird im ersten Teil die Business Process Modeling Notation (BPMN) beschrieben und anschließend auf BPMN-basierende Prozessontologien vorgestellt. Im zweiten Teil werden dann konkrete Ontologien im Anwendungsbereich öffentlicher Verwaltungen erläutert. Wichtig zu beachten ist, dass dieses Kapitel lediglich der kurzen Einleitung bestimmter Arbeiten dient. Eine ausführliche Analyse verschiedener Ontologien folgt im nächsten Kapitel.

3.1. Ontologien basierend auf BPMN

Das Ziel von BPMN ist es, eine allgemein anerkannte Notation zur Verfügung zu stellen, die für den Nutzer einfach zu verstehen und zu verwenden ist. Dabei stellt sie die Verbindung der Prozessmodellierung und -implementierung dar. BPMN formuliert also ein Flussdiagramm, ein sogenanntes Business Process Diagram (BPD), bestimmter Prozessabläufe, in dem die Reihenfolge der Ausführung definiert ist [29]. Als grafische Spezifikationssprache ist BPMN damit in der Lage, Prozesse und deren Abläufe zu modellieren und zu dokumentieren [30]. In den nachfolgenden Abschnitten werden zwei unterschiedliche, auf BPMN basierende Prozessontologien vorgestellt.

BPMN 2.0 Based Ontology for Business Process Representation

Bereits in der Vergangenheit wurden Ontologien entwickelt, die sich mit einer geeigneten Repräsentation von Business-Prozessen beschäftigt haben. Eins dieser Projekte ist die BPMN 2.0 Based Ontology for Business Process Representation (BBO), mit dem Ziel einer detaillierten Prozess- und Wissensrepräsentation [31]. Den Kern dieser Ontologie bilden Fragmente des Business Process Modeling Notation 2.0 (BPMN 2.0)-Metamodells, die durch notwendige Konzepte, Beziehungen und Attribute ergänzt wurden. Die Entwicklung bestand aus fünf konkreten Einzelschritten.

- In der *Spezifikation* werden Anforderungen in Form von Kompetenzfragen an die Ontologie gestellt und der Umfang definiert.

- Auf Basis dieser Informationen wird dann in der *Konzeptionalisierung* ein konzeptionelles Modell erstellt.
- Dieses Modell wird anschließend während der *Formalisierung* in ein formales Modell mit Hilfe einer geeigneten Sprache für die Wissensrepräsentation übersetzt.
- Die *Implementation* beinhaltet anschließend die technische Umsetzung des formalen Modells.
- Den Abschluss bildet die *Wartung*, in der nach erfolgreichem Einsatz auftretende Fehler und Probleme durch entsprechende Updates beseitigt werden.

Für die Evaluation wurden die Schema-Metriken *Schema Deepness*, für den Nachweis einer detaillierten Klassenhierarchie und *Relationship Diversity*, für den Nachweis vielzähliger Beziehungen, die die umfangreiche Klassenhierarchie in Verbindung setzen, berechnet. Zudem wurden die in der Spezifikation aufgestellten Kompetenzfragen mittels SPARQL-Anfragen an die Ontologie beantwortet. Anwendung findet BBO beispielsweise bei der Überwachung der Montage digitaler und physischer Komponenten eines Fahrzeugs.

An Ontology for the Business Process Modeling Notation

Eine weitere mögliche Repräsentation für Business-Prozesse liefert die Business Process Modeling Notation Ontology (BPMNO) [30]. Die Basis dieser Ontologie bildet erneut BPMN, durch die Wiederverwendung spezifischer Bestandteile, die im Verlauf der Entwicklung durch relevante Konzepte, Beziehungen und Attribute ergänzt werden. Das Ziel von BPMNO ist dabei, die BPMN-Spezifikation zu formalisieren und jedes BPMN-Prozessmodell als eine Menge von Individuen und deren Beziehungen darzustellen. Die Entwicklung wurde in drei Einzelschritte aufgeteilt.

- Als Erstes wird jedes Element der BPMN-Spezifikation eindeutig identifiziert und anschließend klassifiziert.
- Im zweiten Schritt werden die entsprechenden Klassen durch Beziehungen und Attribute in Verbindung gesetzt.
- Zum Schluss werden dann für jedes Element Bedingungen aufgestellt, die bei der Nutzung für ein BPD zu beachten sind.

Für die Evaluation werden an die Ontologie gestellte Kompetenzfragen mit Hilfe von SPARQL-Anfragen beantwortet. Verwendung findet BPMNO beispielsweise bei der Zuordnung von Prozessabläufen zwischen verschiedenen Notationen.

3.2. Ontologien im Anwendungsbereich öffentlicher Verwaltungen

Öffentliche Verwaltungen sind in ihrem Handeln an Gesetze gebunden. Dementsprechend ist bei der Entwicklung einer Ontologie in diesem Anwendungsbereich die Arbeit mit Rechtstexten essentiell, um beispielsweise relevante Konzepte eindeutig identifizieren zu können. In den folgenden Abschnitten werden also zwei Ontologien für den Anwendungsbereich öffentlicher Verwaltungen vorgestellt und die verwendeten Methoden erläutert.

Ontologie des Verwaltungshandelns und ihre Anwendung

Die erste Ontologie für den Anwendungsbereich öffentlicher Verwaltungen wird in [32] beschrieben. Das Ziel dieser Ontologie ist die Identifikation und Beschreibung relevanter Elemente in Rechtstexten. Bei der Entwicklung wurden zwei konkrete Schritte befolgt. Der erste Schritt beschreibt den Aufbau eines konzeptionellen Modells, um den Umfang der Ontologie zu definieren und wichtige Konzepte zu beschreiben. Der zweite Schritt beschreibt anschließend die technische Umsetzung des konzeptionellen Modells durch die Auswahl einer geeigneten Beschreibungssprache. Anwendung findet die Ontologie beispielsweise bei der Softwareentwicklung, um wichtige Anforderungen für einen bestimmten Zeitraum zu priorisieren. Außerdem können mit Hilfe der Ontologie bei Gesetzesänderung betroffene Anforderungen eindeutig bestimmt werden.

eGovernment Process Knowledge Ontology

Ein weiteres Beispiel für eine Ontologie im Anwendungsbereich öffentlicher Verwaltungen wird in [33] erläutert. Das Ziel ist dabei die Entwicklung einer Ontologie, die die Überarbeitung und damit die Automatisierung von Prozessen ermöglicht, um die jeweilige öffentliche Verwaltung bestmöglich zu unterstützen. Dafür wird der Aufbau von öffentlichen Diensten und die Abhängigkeiten zwischen den internen Elementen definiert und darauf basierend die entsprechende Ontologie entwickelt. Evaluiert wird die Ontologie mit Hilfe des Prozesses der Fahrzeugregistrierung in Deutschland. Dabei wird dieser öffentliche Dienst für den Nutzer digital zur Verfügung gestellt und dafür notwendige Änderungen erläutert.

4. Konzept

In diesem Kapitel wird der Entwurf unserer Ontologie auf Basis von Kompetenzfragen beschrieben. Dabei wird anhand einer konkreten Methodologie der Verlauf und speziell die Wiederverwendung bereits implementierter Ontologien thematisiert. In den Grundlagen wurden bereits diverse Methodologien erläutert. Für die Erstellung unserer Ontologie haben wir uns für *Ontology Development 101* [22] entschieden, da diese den Ablauf der Entwicklung optimal gliedert und dabei den Aspekt der Wiederverwendung von Ontologien mit einbezieht. Die folgenden Unterabschnitte entsprechen also den ersten drei Entwicklungsschritten dieser Methodologie.

4.1. Zielsetzung

Um die Zielsetzung zu konkretisieren, wird im ersten Schritt der Umfang der Ontologie definiert. Um Anforderungen an unsere Ontologie zu stellen, haben wir die folgenden Kompetenzfragen festgelegt [34].

- CQ1: Welche Leistungen werden angeboten?
- CQ2: Welche Prozesse sind zur Bearbeitung einer Leistung notwendig?
- CQ3: Welche Prozessschritte sind zur Bearbeitung einer Leistung notwendig?
- CQ4: Was ist der erste/letzte Prozessschritt eines Prozesses?
- CQ5: Welche LeikaID hat eine Leistung?
- CQ6: Welche Bearbeitungsfrist hat ein Prozess?
- CQ7: Welche Ressourcen/Datenfelder/Dokumente sind zur Bearbeitung einer Leistung notwendig?
- CQ8: Welche Abgabefrist hat ein Dokument?
- CQ9: Welche DatenfeldID hat ein Datenfeld?
- CQ10: Welche(r) Akteure/Hauptakteur/Ergebnisempfänger/Mitwirkende führen/führt welche Prozesse aus?
- CQ11: Welche(r) Akteure/Hauptakteur/Ergebnisempfänger/Mitwirkende beteiligen/beteiligt sich an welchen Prozessschritten?
- CQ12: Auf welchen Handlungsgrundlagen beruht ein Prozess?
- CQ13: Auf welchen Handlungsgrundlagen beruht welcher Prozessschritt?

- CQ14: Welche Referenzaktivitätengruppen beinhaltet ein Prozess?
- CQ15: Welcher Referenzaktivitätengruppe entspricht welcher Prozessschritt?

4.2. Analyse wiederverwendbarer Ontologien

Der zweite Schritt der Methodologie beinhaltet die Betrachtung wiederverwendbarer Ontologien. Die Suche nach relevanten Ontologien für das eigene Projekt ist dabei immer eine große Hürde, da man durch den speziellen Anwendungsbereich enorm eingeschränkt ist. Um dennoch erfolgreich zu sein, bedarf es einer bestimmten Herangehensweise. Da die Methodologie *Ontology Development 101* [22] nur sehr generische Vorgaben bezüglich der Wiederverwendung von Ontologien beinhaltet, haben wir verschiedene Aspekte von *CLeAR* [25] wie folgt kombiniert.

1. Als erstes wird eine Recherche durchgeführt, um geeignete Ontologien zu finden. Dafür werden konkrete Schlüsselbegriffe definiert und eine Suchmaschine ausgewählt.
2. Danach werden Auswahlkriterien für die gefundenen Ontologien aufgestellt.
3. Abschließend wird eine Auswahl der Ontologien getroffen, durch die Anwendung der entsprechenden Kriterien.

4.2.1. Recherche

Für die Recherche werden im ersten Schritt eine Reihe von Schlüsselbegriffen für unseren Anwendungsbereich aufgestellt. Diese Schlüsselbegriffe werden dann bei der Suche beliebig kombiniert, um die Wahrscheinlichkeit, potentielle Projekte für die Wiederverwendung zu finden, zu maximieren. Da wir uns mit der Entwicklung einer Prozessontologie im Bereich des öffentlichen Dienstes beschäftigen, haben wir uns für die folgenden Schlüsselbegriffe bei der Suche entschieden: *BPMN*, *Knowledge Graph*, *Public Administration*, *Public Service*, *Process*, *Ontology*. Als Suchmaschine haben wir Google Scholar ausgewählt, da diese eine Vielzahl wissenschaftlicher Arbeiten zu diversen Themengebieten zur Verfügung stellt. Durch die Suchbegriffe *Public Administration Ontology* und *Public Service Ontology* haben wir bereits bestehende Projekte im Bereich öffentlicher Verwaltungen selektiert. Mit Hilfe der Suchbegriffe *BPMN Process Ontology* und *Process Ontology* haben wir versucht, andere Prozessontologien zu finden, von deren Aufbau zu lernen und davon zu profitieren. BPMN mit in die Suche zu integrieren war sinnvoll, da viele Ontologien BPMN als Grundbaustein nutzen und BPMN selbst das Ziel einer allgemein anerkannten Notation verfolgt, was zusätzlich die Interoperabilität steigert. Die Recherche wurde im Jahr 2022 durchgeführt und für unsere Analyse haben wir die folgenden Ontologien gefunden: BBO [31], Cyc [35], BPMNO [30], OdV [32], BPMO [36], PSL [37], SUPER, m3po [38], DDPO [39] und GFO [40].

4.2.2. Kriterien

Nach dieser ausführlichen Recherche ist es wichtig, sich intensiv mit den entsprechenden Ergebnissen zu beschäftigen und diese zu bewerten. Dafür erweist es sich als sinnvoll, Bewertungskriterien aufzustellen. Um die gefundenen Ontologien optimal zu analysieren, haben wir also die folgenden Fragen formuliert.

- Ist ein Download der Ontologie im Dateiformat OWL vorhanden?
- Gibt es eine Nutzungslizenz?
- Beinhaltet die Ontologie für uns relevante wiederverwendbare Konzepte?
- Basiert die Ontologie auf BPMN?
- Gibt es eine übersichtliche Dokumentation der Ontologie?

Zu Beginn der Analyse muss für jede Ontologie geprüft werden, ob ein Download im Dateiformat OWL vorhanden ist. Anschließend ist die Art der Nutzungslizenz jeder Ontologie zu prüfen, damit die Wiederverwendung nicht eingeschränkt wird. Sobald diese beiden Kriterien erfüllt sind, wird der Aufbau und die Zielsetzung der Ontologien analysiert und dabei speziell auf wiederverwendbare Konzepte geachtet. Da wir durch das förderale Informationsmanagement (FIM) Prozesse verwenden, die auf BPMN 2.0 basieren, sollten die Ontologien im Optimalfall ebenfalls auf BPMN basieren. FIM dient dabei der Bereitstellung leicht verständlicher Bürgerinformationen, einheitlicher Datenfelder für Formulare Systeme und standardisierter Prozessvorgaben für den Verwaltungsdienst, mit dem Ziel den Übersetzungs- und Implementierungsaufwand rechtlicher Vorgaben zu reduzieren [41]. Abschließend ist es wichtig, die Dokumentation der Ontologie zu kontrollieren, damit die Wiederverwendung der Ontologie den letztendlichen Nutzen nicht übersteigt.

4.2.3. Bewertung

Durch die Anwendung der aufgestellten Bewertungskriterien ist es nun möglich, eine Auswahl für die Wiederverwendung relevanter Ontologien zu treffen. Als Übersicht sind in Tabelle 4.1 grundlegende Informationen zu den entsprechenden Ontologien zusammengefasst.

BBO ist aus dem Jahr 2019 und damit die aktuellste der gefundenen Ontologien. Sie verfolgt zu unserem Projekt ähnliche Ziele und konkretisiert diese durch die Formulierung von Kompetenzfragen. Die wiederverwendbaren Konzepte werden durch einen lizenzfreien Download im geforderten Dateiformat zur Verfügung gestellt. Da BBO auf BPMN 2.0 basiert und eine detaillierte Dokumentation aufweist, ist sie definitiv für die Wiederverwendung in unserem Projekt relevant.

Cyc ist eine umfangreiche Wissensbasis, die für unser Projekt relevante wiederverwendbare Konzepte beinhaltet. Da wir jedoch nicht im Besitz der für die Nutzung benötigten Lizenz sind, steht uns kein Download zur Verfügung. Zudem basiert Cyc nicht auf BPMN und ist somit für unser Projekt auch nicht von Bedeutung.

Name	Beschreibung	wichtige Konzepte	Jahr
BBO	Eine auf BPMN basierte Ontologie zur Repräsentation von Business-Prozessen. Dabei werden Spezifikationen für die Ausführung von Prozessen von BPMN 2.0 verwendet und mit Taxonomien, Konzepten, Beziehungen und Attributen ergänzt.	Process, Input/output specifications, Agent, Work product, Manufacturing facility	2019
Cyc	Definiert eine umfangreiche Wissensbasis, die formalisiertes Wissen für effektives Reasoning und Problemlösungen zu einer Vielzahl von Domänen enthält.	(Aggregate) Process, Script, Scene, Roles/participants, Conditions, Repetition, Properties, Identity	2017
BPMNO	Beschreibt eine Formalisierung der BPMN-Spezifikationen in Form einer Ontologie und ermöglicht damit den Aufbau eines validen BPMN-Diagramms.	Flow Objects, Connecting Objects, Swimlanes, Artifacts	2014
OdV	Eine Ontologie, die in Rechtsvorschriften vorhandene Elemente mit Hilfe einer definierten Semantik korrekt annotiert. Dabei werden etablierte Methoden bei der Arbeit mit Rechtstexten angewendet.	Antragsteller, Ausführender, Beteiligter, Rechtsanwender, Rechtsnorm, Sachverhalt, Rechtsfindung, Mittelwahl	2011
BPMO	Eine Ontologie für die Modellierung von Business-Prozessen auf semantischer Ebene, unter Einbezug von Informationen über den organisatorischen Kontext, Arbeitsabläufe und Aktivitäten.	Process, Business Activity, Task, SemanticCapability, MediationTask, DataMediator	2009
PSL	Beinhaltet eine begrenzte Menge grundlegender Konzepte zur formalen Beschreibung komplexer Prozessontologien.	Activity, Activity-occurrence, Timepoint, Object	2007
SUPER	Beschreibt die Zusammenfassung einer Vielzahl von Ontologien für das Management von Business-Prozessen unter einem Projekt. Die einzelnen Ontologien werden dabei durch die Definition einer Upper Process Ontology miteinander verbunden.	(Non)-Physical Objects, Events, States, Regions Qualities, Constructivist Situations	2007
m3po	Eine Ontologie für die Beschreibung, Ausführung und den Austausch von Business-Prozessen, unter Einbezug von Workflow-Management-Systemen und Choreografie-Beschreibungen.	Event, Activity, ActivityOccurrence, Transition, Loop, ProcessType, ProcessOccurrence	2006
DDPO	Definiert eine Ontologie, die in der Lage ist, eine Unterscheidung zwischen abstrakten und ausführbaren Plänen zu treffen und dabei Neubeschreibungen von Konzepten anderer Ontologien ermöglicht.	Endurant, Perdurant, Time Location, Agentive Physical Object, Social Object.	2003
GFO	Soll bei der Modellierung von Ontologien die Basis bilden, bei der die Theorie mit algorithmischen Verfahren kombiniert und die Interoperabilität gesteigert wird.	Process, Object, Space, Time, Situations, Configurations, Roles, Endurant, Perdurant	2003

Tabelle 4.1.: Zusammenfassung wichtiger Informationen der ausgewählten Ontologien

Die Business Process Modeling Notation Ontology (BPMNO) ist eine auf BPMN basierende Ontologie und weist durch eine zu unserem Projekt ähnliche Zielsetzung für die Wiederverwendung relevante Konzepte auf. Die Zielsetzung wird dabei durch die Formulierung von Kompetenzfragen konkretisiert, jedoch steht kein Download im geforderten Dateiformat zur Verfügung. Zudem existiert keine Dokumentation, wodurch BPMNO keine Relevanz für unser Projekt darstellt.

Die Ontologie des Verwaltungshandelns (OdV) beinhaltet durch eine zu unserem Projekt sehr ähnliche Zielsetzung viele wiederverwendbare Konzepte, jedoch ohne die Formulierung von Kompetenzfragen. Die Inhalte sind über einen Download im geforderten Dateiformat verfügbar und durch eine ausführliche Dokumentation detailliert beschrieben. Auf BPMN basiert die OdV nicht, wodurch die Wiederverwendung den letztendlichen Nutzen überschreitet und sie somit keine Relevanz für unser Projekt darstellt.

Die Business Process Modelling Ontology (BPMO) weist eine ähnliche Zielsetzung zu unserem Projekt auf, ohne die konkrete Formulierung von Kompetenzfragen. Da kein Download zur Verfügung gestellt wird, sind die vorhandenen wiederverwendbaren Konzepte nicht zugänglich. BPMO basiert nicht auf BPMN und eine Dokumentation fehlt komplett. Dadurch kommt diese Ontologie für die Wiederverwendung in unserem Projekt nicht in Frage.

Die Process Specification Language (PSL) fasst grundlegende, für die Wiederverwendung relevante Konzepte zusammen, stellt jedoch keinen Download im Dateiformat OWL zur Verfügung. Zudem ist keine Dokumentation vorhanden und PSL basiert nicht auf BPMN, wodurch sie für die Wiederverwendung in unserem Projekt nicht von Bedeutung ist.

Das Projekt Semantics Utilised for Process Management within/between Enterprises (SUPER) weist verschiedene wiederverwendbare Konzepte vor, jedoch ohne vorhandene Dokumentation. Ein Download im geforderten Dateiformat ist nicht vorhanden und somit sind die entsprechenden Konzepte auch nicht zugänglich. Zudem basiert SUPER nicht auf BPMN und kommt damit für die Wiederverwendung in unserem Projekt nicht in Frage.

Die Multi Metamodel Process Ontology (m3po) ist nicht mehr aktuell. Es ist zwar ein Download vorhanden, jedoch nur in der Web Service Modeling Language (WSML), die nicht mehr unterstützt wird. Zudem fehlt eine Dokumentation komplett und die Ontologie basiert nicht auf BPMN. Die wenigen wiederverwendbaren Konzepte sind also nicht verfügbar und damit ist m3po für unser Projekt nicht relevant.

Die DOLCE + DnS Plan Ontology (DDPO) ist aus dem Jahr 2003 und damit eine der ältesten der gefundenen Ontologien. Die vorhandenen wiederverwendbaren Konzepte sind über einen lizenzfreien Download im Dateiformat OWL erhältlich, jedoch erschwert die fehlende Dokumentation die Wiederverwendung enorm. Zusätzlich basiert DDPO nicht auf BPMN. Für unser Projekt ist die Wiederverwendung dieser Ontologie unvorteilhaft und damit nicht relevant.

Die General Formal Ontology (GFO) ist aus dem Jahr 2003 und damit ebenfalls eine der ältesten der gefundenen Ontologien. Sie weist eine übersichtliche Dokumentation

vor und die Inhalte der GFO sind über einen lizenzfreien Download im Dateiformat OWL verfügbar. Durch fehlende Kompetenzfragen und kaum wiederverwendbare Konzepte übersteigt jedoch der Aufwand der Wiederverwendung den letztendlichen Nutzen. Da die GFO nicht auf BPMN basiert, wird die Wiederverwendung zusätzlich erschwert und damit ist sie für unser Projekt nicht relevant.

In Tabelle 4.2 sind die Ergebnisse der Bewertung zusammengefasst. Der häufigste Ausschlussgrund ist, dass die wiederverwendbaren Konzepte nicht durch einen Download im Dateiformat OWL zur Verfügung stehen. Sollte doch ein Download vorhanden sein, dann ist ein weiterer häufiger Ausschlussgrund, dass die Ontologie nicht auf BPMN basiert. Aus zehn für die Analyse ausgewählten Ontologien wurde also mit BBO letztendlich eine Ontologie für die Wiederverwendung selektiert.

	Download in OWL	freie Lizenz	relevante Konzepte	basiert auf BPMN	gute Dokumentation
BBO	X	X	X	X	X
Cyc	X	-	X	-	X
BPMNO	-	X	X	X	-
OdV	X	X	X	-	X
BPMO	-	X	X	-	-
PSL	-	X	X	-	-
SUPER	-	X	X	-	-
m3po	-	X	X	-	-
DDPO	X	X	X	-	-
GFO	X	X	X	-	X

Tabelle 4.2.: Überblick über die Bewertungskriterien für jede Ontologie

4.3. Entwurf

Da wir nach der vorherigen Analyse nun BBO als Kandidat für die Wiederverwendung selektiert haben, widmen wir uns in diesem Abschnitt der Erstellung des konzeptionellen Modells. Dabei ist es notwendig, die relevanten Begriffe mit Hilfe unserer Kompetenzfragen zu identifizieren und zu definieren. Die Definitionen der entsprechenden Begriffe sind anschaulich in Tabelle 4.3 dargestellt.

Um BBO erfolgreich wiederzuverwenden, sollte anschließend ein Vergleich der Kompetenzfragen durchgeführt werden. Das ist wichtig, da durch die Identifikation gleicher oder ähnlicher Ziele die optimalen Konzepte für die Wiederverwendung ausgewählt werden können. In Tabelle 4.4 ist die Gegenüberstellung der Kompetenzfragen mit ähnlicher Zielsetzung anschaulich dargestellt.

Auf Basis des in Tabelle 4.4 durchgeführten Vergleichs, haben wir die folgenden Konzepte von BBO für die Wiederverwendung ausgewählt.

- Process: Laut BBOs Definition soll *Process* die Zerlegung eines Prozesses in seine Teilprozesse ermöglichen und darstellen können, unter Beachtung der

entsprechenden Reihenfolge bei der Ausführung. Das entspricht auch exakt unserer Definition für *Prozess*, weswegen sich dieses Konzept optimal für die Wiederverwendung eignet.

- **Task:** Als Unterklasse von *Activity* beschreibt *Task* nicht weiter unterteilbare Aufgaben, die durchzuführen sind und erlaubt die Ein- und Ausgabe von Ressourcen. Da unsere Prozesse aus atomaren Prozessschritten bestehen, eignet sich *Task* optimal für die Modellierung dieser Prozessschritte. Um den Ablauf in der korrekten Reihenfolge zu gewährleisten, verwenden wir zusätzlich *FlowNode* als Oberklasse von *Activity* und das Konzept *SequenceFlow*. *FlowNode* beinhaltet dabei alle *Activities*, aus denen sich ein Prozess zusammensetzt und durch *SequenceFlow* wird sichergestellt, dass die entsprechenden *Activities* in der richtigen Reihenfolge ausgeführt werden. In Abbildung 4.1 ist der Zusammenhang dieser Konzepte anschaulich dargestellt.

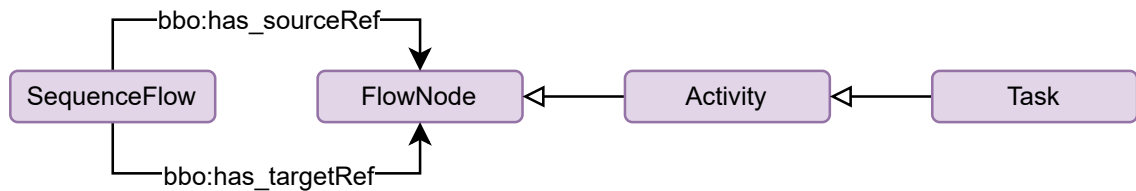


Abbildung 4.1.: Zusammenhang von Task, Activity, FlowNode und SequenceFlow

- **Resource:** BBOs Konzept *Resource* beinhaltet alle Ressourcen jeglicher Typen, auf die durch bestimmte *Activities* verwiesen werden kann. Durch die Wiederverwendung dieses Konzepts ist es uns also möglich, die benötigten Ressourcen, sprich Dokumente und Datenfelder zu modellieren. Dabei modellieren wir jeweils beide als *DataResource*, wobei *Dokument* von *DocumentResource*, einer Unterklasse von *DataResource* modelliert wird, was anschaulich in Abbildung 4.2 dargestellt ist.

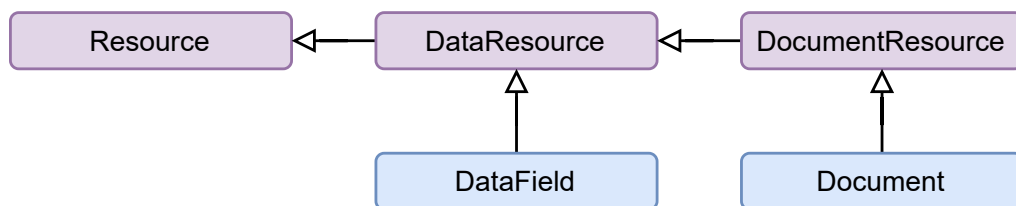


Abbildung 4.2.: Ausschnitt aus BBOs Resource Taxonomie

- **Agent:** Laut BBOs Definition ist ein *Agent* für die Ausführung einer bestimmten *Activity* verantwortlich, was ebenfalls exakt unserer Definition für *Akteur* entspricht, weswegen sich dieses Konzept optimal für die Wiederverwendung eignet.

Nach der erfolgreichen Auswahl wiederverwendbarer Konzepte, können wir nun auf Basis unserer Kompetenzfragen das konzeptionelle Modell unserer Ontologie erstellen, das in Abbildung 4.3 anschaulich dargestellt ist. Mit Hilfe dieses Modells werden

die entsprechenden Konzepte und ihre Beziehungen untereinander visualisiert und so der Aufbau unserer Ontologie verdeutlicht.

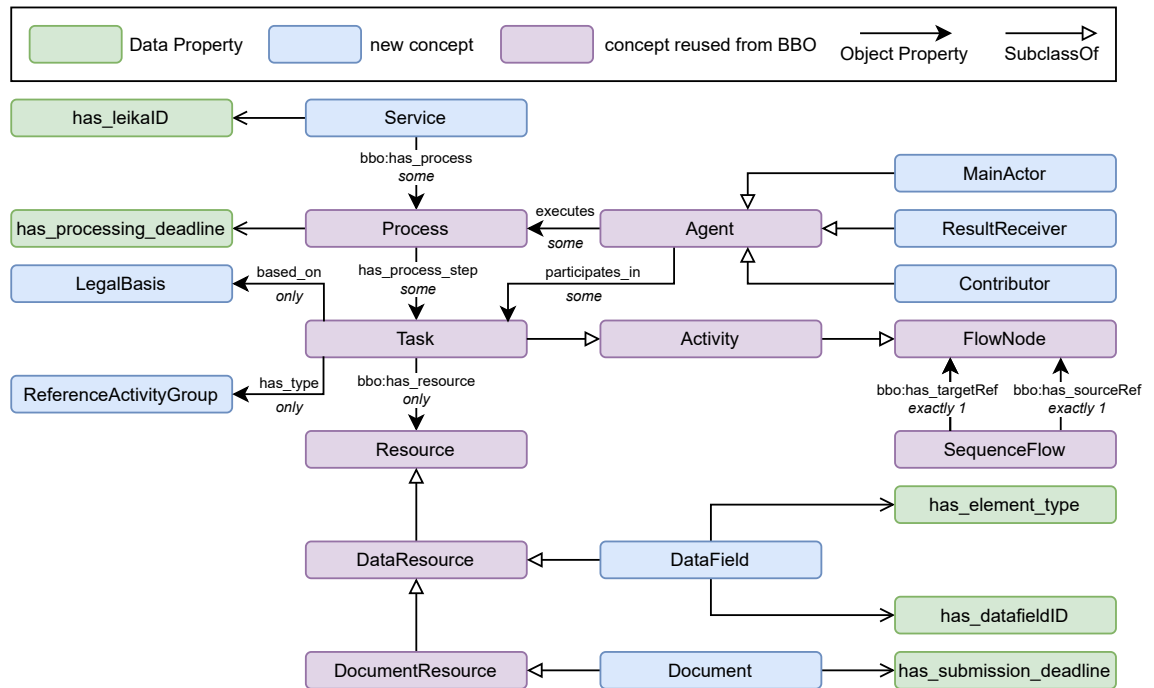


Abbildung 4.3.: Konzeptionelles Modell unserer Ontologie

Der Ausgangspunkt wird durch das Konzept *Service* beschrieben und beinhaltet alle in der Ontologie verfügbaren Leistungen. Jede Leistung ist durch den Leistungskatalog (LeiKa), der umfassende Informationen zu verschiedenen Verwaltungsleistungen enthält, mit einem individuellen Schlüssel (*has_leikaID*) verbunden. Eine Leistung besteht aus einer bestimmten Schrittfolge von Prozessen und jedem Prozess wird durch die entsprechende Beziehung (*bbo:has_process*) die korrekte Leistung zugeordnet. Der Zeitraum der Durchführung eines Prozesses ist zusätzlich durch eine Bearbeitungsfrist (*has_processing_deadline*) begrenzt und jeder Prozess ist ebenfalls in seine Teilschritte zerlegbar. Dabei wird jedem Prozessschritt durch die entsprechende Beziehung (*has_process_step*) der richtige Prozess zugeordnet. Innerhalb einer Leistung kann jedoch ein Prozessschritt auch mehreren Prozessen zugeordnet werden. Außerdem gehört jeder Prozessschritt zu einer konkreten Referenzaktivitätengruppe (RAG) und basiert auf einer bestimmten Handlungsgrundlage. Um die Ausführung in der richtigen Reihenfolge zu gewährleisten, verwenden wir die Konzepte *SequenceFlow* und *FlowNode*. Prozessschritte sind zudem in der Lage benötigte Ressourcen, wie Dokumente oder Datenfelder, anzufordern. Die verschiedenen Ressourcen sind als Unterklasse von *Resource* modelliert. Dabei haben wir uns für die Konzepte *DataResource* und *DocumentResource* der BBO Resource-Taxonomie entschieden, da sie unsere Definitionen für *Dokument* und *Datenfeld* am besten repräsentieren. Dokumente sind zusätzlich an eine bestimmte Abgabefrist (*has_submission_deadline*) gebunden und Datenfelder werden durch einen eindeutigen Schlüssel (*has_datafieldID*) gekennzeichnet. Zudem wird Datenfeldern, durch die Data Property *has_element_type*, ein konkreter Datentyp zugeordnet.

Ein Akteur hat nun die Möglichkeit, Prozesse durchzuführen, oder sich an Prozessschritten zu beteiligen, beide dieser Beziehungen sind relevant. Falls beispielsweise ein Prozessschritt mehreren Prozessen zugeordnet ist und dieser Prozessschritt in jeden Prozess von einem unterschiedlichen Akteur durchgeführt wird, dann lässt sich ohne die Relation *executes* keine eindeutige Aussage treffen, welcher Akteur den Prozessschritt ausführt. Ohne die Relation *participates_in* ist es hingegen nicht möglich, einem Akteur einen bestimmten Prozessschritt zuzuweisen. Die verschiedenen Akteure sind als Unterklassen von *Agent* modelliert. Die Kardinalitäten *only*, *some* und *exactly* der Object Properties werden im nächsten Kapitel erläutert.

wichtige Begriffe	Definition
Leistung	<i>Leistungen</i> beschreiben das Handeln öffentlicher Verwaltungen.
LeiKaID	Die <i>LeiKaID</i> beschreibt den eindeutigen, durch den Leistungskatalog festgelegten Identifikator für Verwaltungsleistungen.
Prozess	<i>Prozesse</i> beschreiben Verwaltungsabläufe, die von bestimmten Personen in einer sich wiederholenden Reihenfolge unter konkreten Vorgaben (bspw. Rechts- und Verwaltungsvorschriften) und unter Nutzung von Hilfsmitteln (bspw. Formulare) bearbeitet werden.
Entscheidungsfrist	<i>Entscheidungsfristen</i> sind an Prozesse gebunden und legen einen Zeitraum fest, in dem die Durchführung eines Prozesses beendet werden muss.
Prozessschritt	<i>Prozessschritte</i> beschreiben Aktionen oder Handlungen der Verwaltung oder der Akteure.
Ressource	<i>Ressourcen</i> beschreiben benötigte Informationen bestimmter Prozessschritte, bei denen es sich entweder um Dokumente oder um Datenfelder handelt.
Datenfeld	<i>Datenfelder</i> sind benötigte Bestandteile beim Aufbau von Formularen.
DatenfeldID	Die <i>DatenfeldID</i> beschreibt den eindeutigen Identifikator für Datenfelder.
Dokument	<i>Dokumente</i> werden zwischen Akteuren ausgetauscht und sind Gegenstand eines Prozessschrittes.
Abgabefrist	<i>Abgabefristen</i> sind an Dokumente gebunden und legen einen Zeitraum fest, in dem ein Dokument eingereicht werden muss.
Akteur	Ein <i>Akteur</i> ist Teilnehmer des Verwaltungsprozesses und übernimmt entweder die Rolle des Ergebnisempfängers, der Mitwirkenden, oder des Hauptakteurs.
Ergebnisempfänger	Der <i>Ergebnisempfänger</i> ist eine Person oder ein Unternehmen, das die Verwaltungsleistung in Anspruch nimmt.
Mitwirkende	<i>Mitwirkende</i> sind externe Behörden, die bei der Verwaltungsleistung mit einbezogen werden.
Hauptakteur	Der <i>Hauptakteur</i> ist eine Behörde, die für die Leistungserbringung verantwortlich ist.
Handlungsgrundlage	<i>Handlungsgrundlagen</i> sind Verweise auf bereits bestehende Gesetze.
Referenzaktivitätengruppe	<i>Referenzaktivitätengruppen</i> lassen sich aus konkreten Vorgaben (bspw. Rechts- und Verwaltungsvorschriften) ableiten, können thematisch zusammengehörige Prozess-Elemente bündeln und spiegeln den Ablauf der Leistungserstellung wieder.

Tabelle 4.3.: Definitionen relevanter Begriffe [42]

Unsere Kompetenzfragen	BBO Kompetenzfragen
Welche <i>Prozessschritte</i> sind zur Bearbeitung einer <i>Leistung</i> notwendig?	In welche Teilschritte wird eine <i>Aktivität</i> zerlegt?
Welche <i>Bearbeitungsfrist</i> hat ein <i>Prozess</i> ?	Was ist die Dauer einer bestimmten <i>Aktivität</i> ?
Was ist der erste/letzte <i>Prozessschritt</i> eines <i>Prozesses</i> ?	Was ist die erste/letzte <i>Aktivität</i> eines bestimmten <i>Prozesses</i> ?
Welche(r) <i>Akteure/Hauptakteur/Ergebnisempfänger/Mitwirkende</i> führen/-führt welche <i>Prozesse</i> aus?	Wer soll/kann eine bestimmte <i>Aktivität</i> ausführen?
Welche <i>Ressourcen/Datenfelder/Dokumente</i> sind zur Bearbeitung einer <i>Leistung</i> notwendig?	Welche <i>Ressourcen</i> werden für eine <i>Aktivität</i> benötigt?

Tabelle 4.4.: Vergleich unserer Kompetenzfragen mit BBO

5. Implementierung

In diesem Kapitel wird erläutert, wie die Ontologie technisch entwickelt wurde. Als Arbeitsumgebung haben wir uns für Protégé [43] entschieden. Bei dieser Software handelt es sich um einen frei verfügbaren Editor für die Modellierung von Ontologien. Nach unserer Methodologie *Ontology Development 101*, wird dabei als erstes die Klassenhierarchie erstellt. Anschließend werden die einzelnen Klassen über Relationen, den sogenannten *Object Properties*, in Verbindung gesetzt. Danach folgt die Erstellung sogenannter *Data Properties*, für die genauere Beschreibung bestimmter Klassen. Abschließend wird unsere Ontologie anhand eines Fallbeispiels mit Individuen populiert. Diese Schrittfolge entspricht exakt den folgenden vier Unterabschnitten. Als Namespace für unsere Ontologie haben wir uns für *ProOnto.org* mit dem Kürzel *pro* entschieden.

5.1. Klassenhierarchie

Damit die Wiederverwendung von BBO gewährleistet wird, muss die entsprechende OWL-Datei [44] in unsere Arbeitsumgebung importiert werden, was den Zugriff auf jedes Konzept von BBO ermöglicht. Da wir nur einzelne spezifische Konzepte von BBO wiederverwenden, ist es sinnvoll auch nur genau diese zu übernehmen, um die Übersichtlichkeit zu verbessern und die Größe der Ontologie zu beschränken. Das bedeutet, dass alle für uns nicht relevanten Konzepte entfernt werden.

Für die Erstellung der Klassenhierarchie wird anschließend für jedes Konzept in unserem Entwurf der Ontologie eine eigene Klasse eingefügt. Um die relevanten Konzepte, *Process*, *Task*, *Resource* und *Agent* von BBO und ihre Funktion erfolgreich wiederzuverwenden, erstellen wir eigene Konzepte als Unterklassen der entsprechenden BBO-Konzepte. Durch Vererbung wird also sichergestellt, dass unsere Konzepte die benötigten Funktionen der jeweiligen Oberklasse besitzen und da wir somit die Oberklassen nicht verändern, wird auch die Interoperabilität nicht eingeschränkt. Die Konzepte *Activity*, *FlowNode* und *SequenceFlow* werden von BBO übernommen, ohne die Erstellung weiterer Unterklassen. Die finale Klassenhierarchie in Protégé ist anschaulich in Abbildung 5.1 dargestellt.

5.2. Object Properties

Nach der erfolgreichen Implementierung der Klassenhierarchie ist es nun wichtig, die einzelnen Klassen durch Relationen in Beziehung zu setzen. Dabei wird für jede

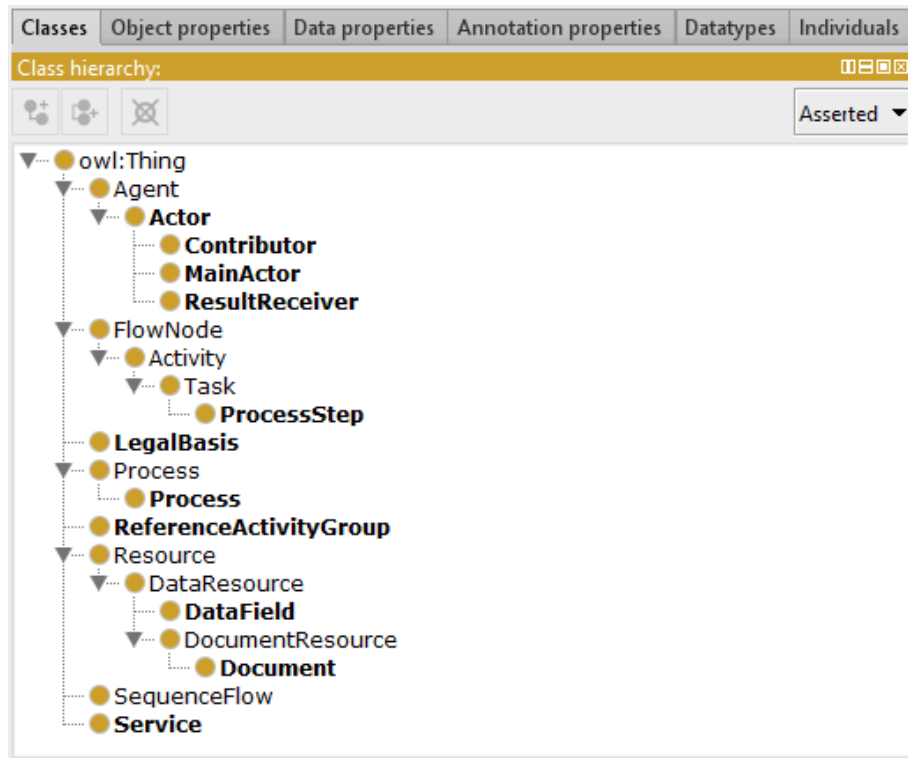


Abbildung 5.1.: Klassenhierarchie unserer Ontologie

Relation in unserem Entwurf der Ontologie eine eigene Relation in Protégé erstellt. Um die Relationen *has_process* und *has_resource* von BBO erfolgreich wiederzuverwenden, spezifizieren wir diese *Object Properties* durch die Erstellung eigener Subrelationen. Dadurch wird die Funktionalität von BBO nicht eingeschränkt und die Interoperabilität wird weiterhin gewährleistet. Die restlichen, nicht relevanten *Object Properties* von BBO werden entfernt. Anschließend wird für jede Relation eindeutig spezifiziert, welche Klassen durch die entsprechende Relation in Beziehung gesetzt werden. Dafür werden von Protégé *Domain* und *Range* zur Verfügung gestellt. Sobald man für eine bestimmte Relation eine bestimmte Klasse als *Domain* festlegt, wird eine Instanz, die das Subjekt dieser Relation bildet, der in *Domain* festgelegten Klasse zugeordnet. Wenn also beispielsweise das Szenario *Hans hat_kind Peter* existiert und für die Relation *hat_kind* die Klasse *Person* als *Domain* festgelegt wird, dann wird *Hans* als Instanz der Klasse *Person* zugeordnet. Sobald man jedoch für eine bestimmte Relation eine bestimmte Klasse als *Range* festlegt, dann wird eine Instanz, die das Objekt dieser Relation bildet, der in *Range* festgelegten Klasse zugeordnet. Wenn wir also erneut von dem Szenario *Hans hat_kind Peter* ausgehen und für die Relation *hat_kind* die Klasse *Person* als *Range* festgelegt wird, dann wird *Peter* als Instanz der Klasse *Person* zugeordnet.

Um zwei Klassen final in Beziehung zu setzen, muss man für die als *Domain* festgelegte Klasse ein Axiom erstellen. In Abbildung 5.2 ist dieser Vorgang beispielhaft dargestellt. Im Beispiel wird die Relation *has_process* zwischen der als *Domain* festgelegten Klasse *Service* und der als *Range* festgelegten Klasse *Process* thematisiert. Da *Service* die *Domain* der Relation bildet, wird das Axiom auch zur Klasse *Service*

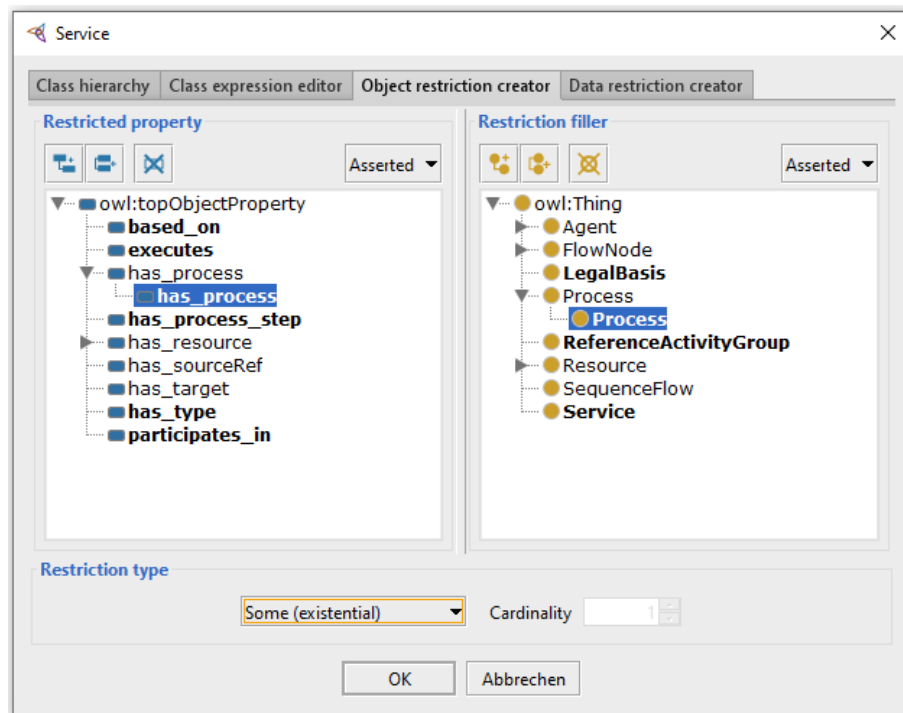


Abbildung 5.2.: Erstellung eines Axioms anhand der Relation *has_process*

hinzugefügt. Anschließend wird die entsprechende *Object Property* *has_process* und die als *Range* festgelegte Klasse *Process* ausgewählt. Zum Schluss trifft man eine Aussage über die Quantoren und die Kardinalität der Relation. Dabei hat man die folgenden fünf Möglichkeiten, die anhand des Beispiels erklärt werden.

- *Some (existential)*: Eine Leistung hat **mindestens** einen Teilprozess, kann aber auch mehrere Teilprozesse haben.
- *Only (universal)*: Eine Leistung hat **ausschließlich** Teilprozesse, es existieren keine weiteren Relationen mit der Property *has_process*.
- *Min (cardinality)*: Eine Leistung hat **mindestens** die als Kardinalität festgelegte Anzahl an Teilprozessen.
- *Exactly (cardinality)*: Eine Leistung hat **genau** die als Kardinalität festgelegte Anzahl an Teilprozessen.
- *Max (cardinality)*: Eine Leistung hat **maximal** die als Kardinalität festgelegte Anzahl an Teilprozessen.

Die finale Hierarchie der implementierten Relationen ist anschaulich in Abbildung 5.3 dargestellt.

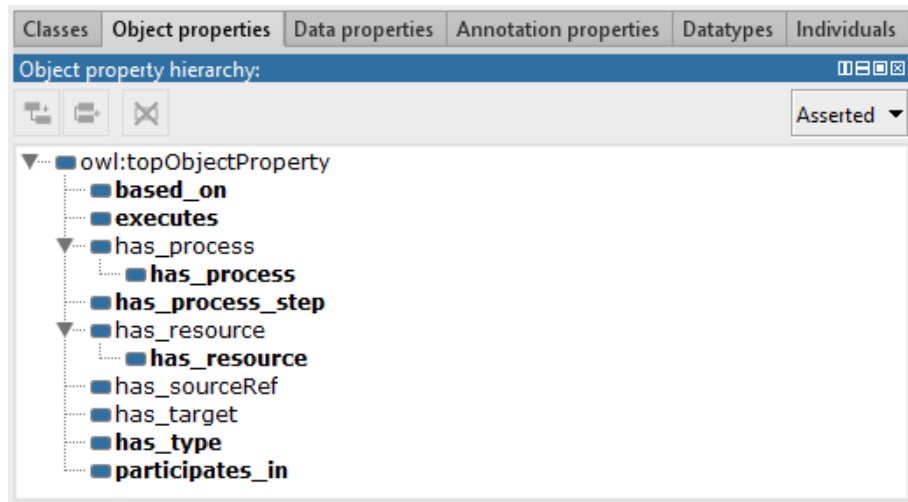


Abbildung 5.3.: Hierarchie der Object Properties unserer Ontologie

5.3. Data Properties

Nachdem die Klassen untereinander in Beziehung gesetzt werden, fügen wir die benötigten *Data Properties* ein. Dabei ist es durch *Data Properties* möglich, einer Klasse bestimmte Werte zuzuordnen, um die entsprechende Klasse noch spezifischer zu beschreiben. Bei der Erstellung einer *Data Property* muss sie also einer bestimmten Klasse zugeordnet und ein entsprechender Datentyp festgelegt werden. Dafür wird von Protégé, genau wie bei dem Einfügen der *Object Properties*, *Domain* und *Range* zur Verfügung gestellt. Als *Domain* wird die zugehörige Klasse festgelegt und als *Range* der entsprechende Datentyp. In Tabelle 5.1 sind alle *Data Properties*, ihre dazugehörigen *Klassen* und die entsprechenden Datentypen dargestellt.

Data Property	Domain (Klasse)	Range (Datentyp)
has_leikaID	Service	string
has_processing_deadline	Process	dateTime
has_element_type	DataField	int
has_datafieldID	DataField	string
has_submission_deadline	Document	dateTime

Tabelle 5.1.: *Domain* und *Range* der benötigten *Data Properties*

Da es sich bei einer Frist um ein bestimmtes Datum handelt, verwenden wir für die Modellierung von *has_processing_deadline* und *has_submission_deadline* den Datentyp *dateTime*. Für die Modellierung der beiden Identifikatoren *has_leikaID* und *has_datafieldID* verwenden wir den Datentyp *string*, wodurch wir für Instanzen der Klassen *Service* und *DataField* eindeutige Zeichenketten festlegen können. Die beiden *Data Properties* *has_leikaID* und *has_datafieldID* werden zusätzlich als Unterklassen der von BBO zur Verfügung gestellten *Data Property id* modelliert, um die Interoperabilität zu erhöhen. Zudem übernehmen wir die *Data Property name*, ebenfalls für die Erhöhung der Interoperabilität. Die restlichen, nicht relevanten *Data Properties* von BBO werden entfernt.

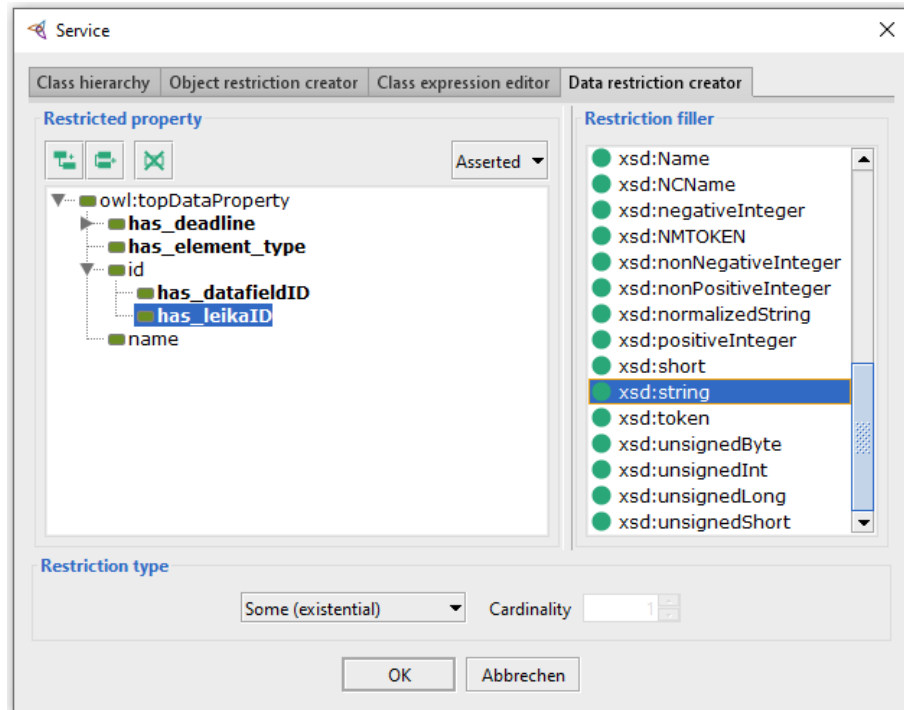


Abbildung 5.4.: Erstellung eines Axioms anhand der *Data Property* *has_leikaID*

Um die Erstellung einer *Data Property* abzuschließen, muss genau wie bei den *Object Properties* für die als *Domain* festgelegte Klasse ein Axiom erstellt werden. Dieser Vorgang ist in Abbildung 5.4 beispielhaft dargestellt. Im Beispiel wird die *Data Property* *has_leikaID* der Klasse *Service* thematisiert. Da *Service* die *Domain* der Relation bildet, wird das Axiom auch zur Klasse *Service* hinzugefügt. Danach wird die entsprechende *Data Property* *has_leikaID* und der als *Range* festgelegte Datentyp *string* ausgewählt. Abschließend trifft man auch hier eine Aussage über die Quantoren und die Kardinalität der *Data Property*, was bereits ausführlich thematisiert wurde.

In Abbildung 5.5 ist die finale Hierarchie der *Data Properties* anschaulich dargestellt.

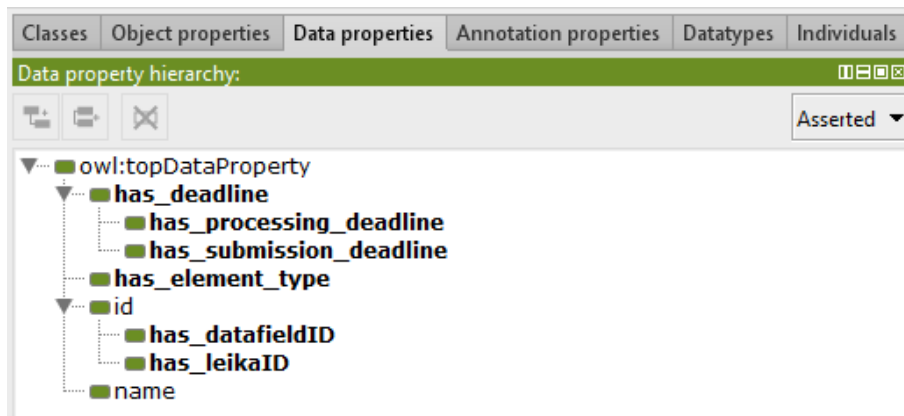


Abbildung 5.5.: Hierarchie der *Data Properties* unserer Ontologie

5.4. Individuen

Mit der Implementierung der *Data Properties* ist das Gerüst der Ontologie abgeschlossen. Im letzten Schritt wird die Ontologie nun mit Instanzen populiert. Dieser Entwicklungsschritt wurde von Mitarbeitern des Projekts durchgeführt und ausführlich auf einer Website [34] dokumentiert. Wir werden daher auf die Population der Ontologie mit Individuen nur verkürzt eingehen.

Um die Ontologie zu populieren nutzen wir *Apache Jena* [45], ein frei verfügbares Java-Framework, was die Generierung der Individuen im RDF-Format ermöglicht. Da wir mit FIM-Prozessen arbeiten, werden die entsprechenden FIM-Artefakte, *XProzess* [46] und *XDatenfeld* [47] als Input verwendet. Dabei enthält *XProzess* einen annotierten BPMN-Prozess und *XDatenfeld* enthält den Aufbau für ein Formular.

Für die Generierung der Individuen werden die XML-Dateien *XProzess* und *XDatenfeld* geparkt, wobei Java-Klassen erstellt werden, die als Schnittstelle in andere Dateiformate agieren. Dabei werden die Klassen nach Vorbild des Entwurfs der Ontologie erstellt.

6. Evaluation

In diesem Kapitel wird unsere Ontologie evaluiert. Dabei wird im ersten Teil die Fähigkeit unserer Ontologie überprüft, die von uns aufgestellten Kompetenzfragen mit Hilfe von SPARQL-Anfragen zu beantworten. Dabei verwenden wir *Apache Jena Fuseki* [48], einen SPARQL-Server, der das Hochladen von Ontologien und die anschließende Abfrage dieser mittels SPARQL ermöglicht. Im zweiten Teil wird unsere Ontologie durch OOPS! validiert. OOPS! hilft dabei, nach Fertigstellung der Ontologie häufig auftretende Fehler bei der Modellierung von Ontologien zu erkennen und anschließend zu beseitigen.

6.1. SPARQL-Anfragen

Da unsere Ontologie anhand eines Fallbeispiels populiert wurde, wurden den Klassen konkrete Individuen zugeordnet. Ziel ist es also für die in den Kompetenzfragen geforderten Klassen, die entsprechenden Individuen auszugeben. Die einzelnen Kompetenzfragen und die dazugehörigen SPARQL-Anfragen sind im Folgenden aufgelistet. Jede der SPARQL-Anfragen folgt dabei der gleichen Struktur. Als erstes werden die benötigten URIs durch die Definition von Präfixen abgekürzt, um die Anfrage übersichtlicher zu gestalten. Durch das Präfix *rdf* ist es uns möglich, auf die RDF-Semantik und die RDF-Syntax zuzugreifen und durch die Präfixe *bbo* und *pro* können wir auf entsprechende Informationen der BBO-Ontologie und unserer eigenen Ontologie zugreifen. Anschließend werden durch *SELECT* alle benötigten RDF-Daten, die die festgelegte *WHERE*-Bedingung erfüllen, tabellarisch ausgegeben. Die Ergebnisse jeder Anfrage wurden manuell auf Korrektheit überprüft und in GitLab [49] dokumentiert.

CQ1: Welche Leistungen werden angeboten?

Durch die folgende SPARQL-Anfrage geben wir alle Instanzen der Klasse *Leistung* aus. Dabei verwenden wir *rdf:type*, um anzugeben, dass eine Information eine Instanz einer Klasse ist und spezifizieren die Klasse durch *www:Leistung*.

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX bbo: <http://BPMNbasedOntology#>
PREFIX pro: <http://www.ProOnto.org/#>

SELECT *
WHERE {
  ?Leistung rdf:type pro:Leistung
}

```

CQ2: Welche Prozesse sind zur Bearbeitung einer Leistung notwendig?

Die folgende SPARQL-Anfrage gibt alle Prozesse aus, die über die entsprechende Beziehung *hat_prozess* mit der im *FILTER* festgelegten Leistung verbunden sind.

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX bbo: <http://BPMNbasedOntology#>
PREFIX pro: <http://www.ProOnto.org/#>

SELECT ?Prozess
WHERE {
  FILTER(?Leistung=pro:leistung_99006028261000)
  ?Leistung pro:hat_prozess ?Prozess.
}

```

CQ3: Welche Prozessschritte sind zur Bearbeitung einer Leistung notwendig?

Mit der folgenden SPARQL-Anfrage werden alle Prozessschritte ausgegeben, die über die Beziehung *hat_prozessschritt* mit den Prozessen verbunden sind, die über die Beziehung *hat_prozess* mit der im *FILTER* festgelegten Leistung verbunden sind.

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX bbo: <http://BPMNbasedOntology#>
PREFIX pro: <http://www.ProOnto.org/#>

SELECT ?Prozessschritt
WHERE {
  FILTER(?Leistung=pro:leistung_99006028261000)
  ?Leistung pro:hat_prozess ?Prozess.
  ?Prozess pro:hat_prozessschritt ?Prozessschritt
}

```

CQ4: Was ist der erste/letzte Prozessschritt eines Prozesses?

Durch die folgende SPARQL-Anfrage geben wir jeweils den ersten und letzten Prozessschritt eines Prozesses aus. Dabei verwenden wir die BBO-Konzepte *StartEvent*

und *EndEvent*, die jeweils den Anfang und das Ende eines Prozesses darstellen. Da es sich bei diesen Konzepten jedoch nicht um Prozessschritte handelt, verwenden wir die Beziehungen *bbo:has_next_flownode*, um nach dem *StartEvent* den ersten Prozessschritt darzustellen und *bbo:has_previous_flownode*, um vor dem *EndEvent* den letzten Prozessschritt darzustellen.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX bbo: <http://BPMNbasedOntology#>
PREFIX pro: <http://www.ProOnto.org/#>

SELECT ?erster_Prozessschritt ?letzter_Prozessschritt
WHERE {
  ?StartEvent rdf:type bbo:StartEvent.
  ?EndEvent rdf:type bbo:EndEvent.
  ?StartEvent bbo:has_nextFlowNode ?erster_Prozessschritt.
  ?EndEvent bbo:has_previousFlowNode ?letzter_Prozessschritt
}
```

CQ5: Welche LeikaID hat eine Leistung?

Die folgende SPARQL-Anfrage gibt die LeikaID aus, die über die Data Property *hat_leikaID* mit der im *FILTER* angegebenen Leistung verbunden ist.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX bbo: <http://BPMNbasedOntology#>
PREFIX pro: <http://www.ProOnto.org/#>

SELECT ?ID
WHERE {
  FILTER(?Leistung=pro:leistung_99006028261000)
  ?Leistung pro:hat_leikaID ?ID
}
```

CQ6: Welche Bearbeitungsfrist hat ein Prozess?

Mit der folgenden SPARQL-Anfrage wird die Bearbeitungsfrist ausgegeben, die über die Data Property *hat_bearbeitungsfrist* mit dem im *FILTER* angegebenen Prozess verbunden ist. Die Ausgabe bleibt bei dieser SPARQL-Anfrage jedoch leer, da in unserem Fallbeispiel keine Instanz mit einer Bearbeitungsfrist modelliert ist.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX bbo: <http://BPMNbasedOntology#>
PREFIX pro: <http://www.ProOnto.org/#>

SELECT ?Bearbeitungsfrist
WHERE {
  FILTER(?Prozess=pro:prozess_99006028261000)
  ?Prozess pro:hat_bearbeitungsfrist ?Bearbeitungsfrist
}
```

CQ7: Welche Ressourcen/Datenfelder/Dokumente sind zur Bearbeitung einer Leistung notwendig?

Durch die folgende SPARQL-Anfrage werden alle Ressourcen ausgegeben, die über die Beziehung *hat_ressource* mit den Prozessschritten verbunden sind, die über die Beziehung *hat_prozessschritt* mit den Prozessen verbunden sind, die über die Beziehung *hat_prozess* mit der im *FILTER* festgelegten Leistung verbunden sind. Durch *DISTINCT* werden dabei Wiederholungen bei der Ausgabe vermieden. Nach erstmaliger Ausführung war die Ausgabe leer und die gewünschten Ressourcen wurden nicht angezeigt. Das lag daran, dass wir innerhalb der Ontologie nicht die von uns definierte Beziehung *hat_ressource* zwischen den Konzepten *Prozessschritt* und *Ressource* verwendet haben, sondern *has_resource* von BBO. Nachdem dieser Fehler behoben war, funktionierte die SPARQL-Anfrage fehlerfrei.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX bbo: <http://BPMNbasedOntology#>
PREFIX pro: <http://www.ProOnto.org/#>

SELECT DISTINCT ?Resource
WHERE {
  FILTER(?Leistung=pro:leistung_99006028261000)
  ?Leistung pro:hat_prozess ?Prozess.
  ?Prozess pro:hat_prozessschritt ?Prozessschritt.
  ?Prozessschritt pro:hat_ressource ?Resource
}
```

CQ8: Welche Abgabefrist hat ein Dokument?

Mit der folgenden SPARQL-Anfrage werden alle Dokumente und die dazugehörigen Abgabefristen, die über die Data Property *hat_abgabefrist* mit dem entsprechenden Dokument verbunden sind, ausgegeben. Die Ausgabe bleibt bei dieser SPARQL-Anfrage jedoch leer, da in unserem Fallbeispiel keine Instanz mit einer Abgabefrist modelliert ist.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX bbo: <http://BPMNbasedOntology#>
PREFIX pro: <http://www.ProOnto.org/#>

SELECT *
WHERE {
  ?Dokument pro:hat_abgabefrist ?Abgabefrist
}
```

CQ9: Welche DatenfeldID hat ein Datenfeld?

Durch die folgende SPARQL-Anfrage werden alle Datenfelder und die dazugehörigen DatenfeldIDs, die über die Data Property *hat_datenfeldID* mit dem entsprechenden Datenfeld verbunden sind, ausgegeben.


```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX bbo: <http://BPMNbasedOntology#>
PREFIX pro: <http://www.ProOnto.org/#>

SELECT *
WHERE {
    ?Datenfeld pro:hat_datenfeldID ?ID
}
```

CQ10: Welche(r) Akteure/Hauptakteur/Ergebnisempfänger/Mitwirkende führen/führt welche Prozesse aus?

Die folgende SPARQL-Anfrage gibt alle Akteure und die dazugehörigen Prozesse aus, die über die Beziehung *fuehrt_aus* mit dem entsprechenden Akteur verbunden sind.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX bbo: <http://BPMNbasedOntology#>
PREFIX pro: <http://www.ProOnto.org/#>

SELECT DISTINCT *
WHERE {
    ?Akteur pro:fuehrt_aus ?Prozess
}
```

CQ11: Welche(r) Akteure/Hauptakteur/Ergebnisempfänger/Mitwirkende beteiligen/beteiligt sich an welchen Prozessschritten?

Mit der folgenden SPARQL-Anfrage werden alle Akteure und die dazugehörigen Prozessschritte ausgegeben, die über die Beziehung *beteiligt_sich_an* mit dem entsprechenden Akteur verbunden sind.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX bbo: <http://BPMNbasedOntology#>
PREFIX pro: <http://www.ProOnto.org/#>

SELECT DISTINCT *
WHERE {
    ?Akteur pro:beteiligt_sich_an ?Prozessschritt
}
```

CQ12: Auf welchen Handlungsgrundlagen beruht ein Prozess?

Durch die folgende SPARQL-Anfrage werden alle Handlungsgrundlagen ausgegeben, die über die Beziehung *basiert_auf* mit den Prozessschritten verbunden sind, die über die Beziehung *hat_prozessschritt* mit dem im *FILTER* festgelegten Prozess verbunden sind.

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX bbo: <http://BPMNbasedOntology#>
PREFIX pro: <http://www.ProOnto.org/#>

SELECT DISTINCT ?Handlungsgrundlage
WHERE {
  FILTER(?Prozess=pro:prozess_99006028261000)
  ?Prozess pro:hat_prozessschritt ?Prozessschritt.
  ?Prozessschritt pro:basiert_auf ?Handlungsgrundlage
}

```

CQ13: Auf welchen Handlungsgrundlagen beruht welcher Prozessschritt?

Die folgende SPARQL-Anfrage gibt alle Prozessschritte und die dazugehörigen Handlungsgrundlagen aus, die über die Beziehung *basiert_auf* mit dem entsprechenden Prozessschritt verbunden sind.

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX bbo: <http://BPMNbasedOntology#>
PREFIX pro: <http://www.ProOnto.org/#>

SELECT *
WHERE {
  ?Prozessschritt rdf:type pro:Prozessschritt.
  ?Prozessschritt pro:basiert_auf ?Handlungsgrundlage
}

```

CQ14: Welche Referenzaktivitätengruppen beinhaltet ein Prozess?

Mit Hilfe der folgenden SPARQL-Anfrage werden alle Referenzaktivitätengruppen ausgegeben, die über die Beziehung *hat_typ* mit den Prozessschritten verbunden sind, die über die Beziehung *hat_prozessschritt* mit dem im *FILTER* festgelegten Prozess verbunden sind.

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX bbo: <http://BPMNbasedOntology#>
PREFIX pro: <http://www.ProOnto.org/#>

SELECT DISTINCT ?RAG
WHERE {
  FILTER(?Prozess=pro:prozess_99006028261000)
  ?Prozess pro:hat_prozessschritt ?Prozessschritt.
  ?Prozessschritt pro:hat_typ ?RAG
}

```

CQ15: Welcher Referenzaktivitätengruppe entspricht welcher Prozessschritt?

Durch die folgende SPARQL-Anfrage werden alle Prozessschritte und die dazugehörige Referenzaktivitätengruppe ausgegeben, die über die Beziehung *hat_typ* mit dem entsprechenden Prozessschritt verbunden ist.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX bbo: <http://BPMNbasedOntology#>
PREFIX pro: <http://www.ProOnto.org/#>

SELECT *
WHERE {
  ?Prozessschritt pro:hat_typ ?RAG
}
```

6.2. Ontology Pitfall Scanner!

Für die Evaluation mit OOPS! laden wir unsere Ontologie auf der zur Verfügung gestellten Webseite hoch. Anschließend wird der Scanner gestartet, der auftretende Probleme und Fehler bei der Modellierung ausgibt. Jeder der ausgegebenen Vorschläge wird dabei einer der drei folgenden Kategorien zugeordnet.

- *Critical*: Fehler, die definitiv behoben werden müssen, da sie die Konsistenz und Funktion der Ontologie einschränken.
- *Important*: Fehler, die zwar nicht die Funktion der Ontologie einschränken, aber dennoch behoben werden sollten.
- *Minor*: Fehler, die in keiner Weise die Funktion der Ontologie einschränken, aber nach Behebung für eine einheitliche und gut strukturierte Ontologie sorgen.

Der in Abbildung 6.1 dargestellte Pitfall weist auf eine fehlende *Domain* und *Range* der *Object Properties* *hasTargetRef*, *hasSourceRef* und der *Data Property* *hatFrist* hin. Da es sich bei den *Object Properties* *hasTargetRef* und *hasSourceRef* um Bestandteile von BBO, einer Ontologie außerhalb unserer Reichweite, handelt, können wir den Pitfall für diese beiden Beziehungen nicht bearbeiten. Die *Data Property* *hatFrist* hingegen haben wir bewusst ohne *Domain* modelliert, um sie für konkrete Sub-Properties zu spezifizieren. So haben die beiden Sub-Properties *hatBearbeitungsfrist* und *hatAbgabefrist* jeweils unterschiedliche *Domains*, weswegen man für die Oberklasse *hatFrist* keine konkrete *Domain* festlegen kann.

In Abbildung 6.2 ist der Pitfall dargestellt, der auftritt, falls ein Bestandteil der Ontologie als Klasse genutzt wird, jedoch nicht eindeutig als Klasse deklariert wurde. Die Klassen, bei denen der Pitfall auftritt, sind ausschließlich Bestandteile von BBO und damit erneut außerhalb unserer Reichweite, weswegen wir diesen Pitfall für die angegebenen Klassen nicht bearbeiten können.

Results for P11: Missing domain or range in properties. 3 cases | Important

Object and/or datatype properties without domain or range (or none of them) are included in the ontology.

- This pitfall appears in the following elements:
 - > http://BPMNbasedOntology#has_targetRef
 - > http://BPMNbasedOntology#has_sourceRef
 - > http://www.semanticweb.org/#hat_frist

Abbildung 6.1.: Pitfall 11: Fehlende Domain/Range bei Properties

Results for P34: Untyped class. 6 cases | Important

An ontology element is used as a class without having been explicitly declared as such using the primitives owl:Class or rdfs:Class. This pitfall is related with the common problems listed in [8].

- This pitfall appears in the following elements:
 - > <http://BPMNbasedOntology#DocumentResource>
 - > <http://BPMNbasedOntology#Task>
 - > <http://BPMNbasedOntology#Process>
 - > <http://BPMNbasedOntology#DataResource>
 - > <http://BPMNbasedOntology#Agent>
 - > <http://BPMNbasedOntology#Resource>

Abbildung 6.2.: Pitfall 34: Nicht-deklarierte Klasse

Der in Abbildung 6.3 dargestellte Pitfall ist sehr ähnlich zu seinem Vorgänger, nur dass der Fokus hier auf den Beziehungen zwischen den Klassen liegt. Ein Bestandteil der Ontologie wird also als Beziehung genutzt, jedoch nicht eindeutig als diese deklariert. Erneut befinden sich in der Ausgabe ausschließlich Bestandteile von BBO, weswegen wir auch diesen Pitfall für die angegebenen Beziehungen nicht bearbeiten können.

Results for P35: Untyped property. 3 cases | Important


An ontology element is used as a property without having been explicitly declared as such using the primitives rdf:Property, owl:ObjectProperty or owl:DatatypeProperty. This pitfall is related with the common problems listed in [8].

- This pitfall appears in the following elements:
 - > http://BPMNbasedOntology#has_process
 - > <http://BPMNbasedOntology#id>
 - > http://BPMNbasedOntology#has_resource

Abbildung 6.3.: Pitfall 35: Nicht-deklarierte Beziehung

Vor der Veröffentlichung der Ontologie sollte eine entsprechende Lizenz gewählt werden. Das sollte jedoch erst nach Abschluss der Modellierung aller Verwaltungsleistungen innerhalb der Ontologie erfolgen. Da wir aktuell ausschließlich ein Fallbeispiel modelliert haben, ist der in Abbildung 6.4 dargestellte Pitfall für uns aktuell nicht relevant.

Da in der Ausgabe der Pitfalls ausschließlich BBO-Konzepte aufgelistet wurden, hat die Evaluation mit Hilfe von OOPS! gezeigt, dass durch die Implementierung der von uns benötigten Konzepte keine Fehler aufgetreten sind.

Results for P41: No license declared. ontology* | Important 

The ontology metadata omits information about the license that applies to the ontology.

*This pitfall applies to the ontology in general instead of specific elements.

Abbildung 6.4.: Pitfall 41: Keine Lizenz festgelegt

7. Fazit

In dieser Arbeit haben wir die schrittweise Entwicklung einer Ontologie als Wissensbasis für Verwaltungsprozesse im Anwendungsbereich öffentlicher Verwaltungen vorgestellt. Die Umsetzung dieser Ontologie wurde durch die Methodologien *Ontology Development 101* und *CLeAR* in einzelne Schritte unterteilt, die im Verlauf der Entwicklung nacheinander abgearbeitet wurden. Für die Implementierung der Ontologie haben wir ontologische Ressourcen bereits bestehender Ontologien analysiert und wiederverwendet. Die finale Ontologie haben wir anschließend im Dateiformat OWL gespeichert, ohne sie zu veröffentlichen. Für eine Veröffentlichung sind noch einige Verbesserungsvorschläge und Ergänzungen zu implementieren, die ausführlich im Kapitel Ausblick thematisiert werden. Die Qualität der Ontologie haben wir durch die erfolgreiche Evaluation mit Hilfe von SPARQL-Anfragen und OOPS! nachgewiesen. In dieser Arbeit wurde also ein weiterer Grundstein für das langfristige Ziel der Digitalisierung von Verwaltungsleistungen gelegt.

8. Ausblick

Wie im Fazit bereits erwähnt, wird unsere Ontologie vorerst nicht veröffentlicht. Vorschläge für zukünftige Änderungen und Ergänzungen thematisieren wir in diesem Abschnitt. Da in unserer Ontologie aktuell ausschließlich ein Fallbeispiel einer Leistung modelliert ist, sollte die Ontologie in Zukunft noch durch weitere Fallbeispiele ergänzt werden. Dafür notwendig sind die Vereinheitlichung der entsprechenden Leistungen und die Erstellung weiterer benötigter Klassen und Eigenschaften, sodass unsere Ontologie irgendwann alle verfügbaren Verwaltungsleistungen beinhaltet. Zudem ist die Erstellung einer URI und einer entsprechenden Nutzungslizenz für die eindeutige Identifikation und Kennzeichnung im semantischen Web von Bedeutung. Erst danach ist die Ontologie für eine Veröffentlichung bereit. Ein weiterer denkbarer Ausblick ist die automatische Population der Ontologie auf Basis unstrukturierter Informationen aus Rechtstexten. Zudem könnte die Dokumentation der Ontologie durch die Erstellung eines Glossars, das zu jedem Konzept detaillierte Informationen enthält, verbessert werden.

Literaturverzeichnis

- [1] Bundesministerium des Innern und für Heimat. Stand: 31.08.2022. URL: <https://www.onlinezugangsgesetz.de/Webs/OZG/DE/startseite/startseite-node.html>.
- [2] Friedrich-Schiller Universität Jena. *Arbeitsgruppe offenes Design digitaler Verwaltungsarchitekturen*. Stand: 13.03.2023. URL: <https://www.opendva.uni-jena.de/>.
- [3] Zhan Cui, Ernesto Damiani und Marcello Leida. „Benefits of Ontologies in Real Time Data Access“. In: *2007 Inaugural IEEE-IES Digital EcoSystems and Technologies Conference*. 2007, S. 392–397. DOI: 10.1109/DEST.2007.372004.
- [4] Tobias Bürger und Elena Simperl. „Measuring the Benefits of Ontologies“. In: *On the Move to Meaningful Internet Systems: OTM 2008 Workshops*. Hrsg. von Robert Meersman, Zahir Tari und Pilar Herrero. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, S. 584–594. ISBN: 978-3-540-88875-8.
- [5] María Poveda-Villalón, Asunción Gómez-Pérez und Mari Carmen Suárez - Figueroa. „OOPS! (Ontology Pitfall Scanner!): An On-line Tool for Ontology Evaluation“. In: *International Journal on Semantic Web and Information Systems (IJSWIS)* 10.2 (2014), S. 7–34.
- [6] IONOS SE. *Was bedeutet Semantic Web?* Stand: 13.12.2022. Sep. 2021. URL: <https://www.ionos.de/digitalguide/online-marketing/suchmaschinen-marketing/semantic-web/>.
- [7] Christine Hoyland u. a. „The RQ-Tech Methodology: A New Paradigm for Conceptualizing Strategic Enterprise Architectures“. In: *Journal of Management Analytics* 1 (Mai 2014), S. 55–77. DOI: 10.1080/23270012.2014.889912.
- [8] Heike Müller. *Erstellung von Bibliographien auf der Basis von XML und XSL*. Informationswirtschaft (Bachelor, Diplom). 2003.
- [9] Pascal Hitzler u. a. „Einfache Ontologien in RDF und RDF Schema“. In: *Semantic Web: Grundlagen*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, S. 33–88. ISBN: 978-3-540-33994-6. DOI: 10.1007/978-3-540-33994-6_4. URL: https://doi.org/10.1007/978-3-540-33994-6_4.
- [10] Tim Berners-Lee, Roy T. Fielding und Larry M. Masinter. *Uniform Resource Identifier (URI): Generic Syntax*. RFC 3986. Jan. 2005. DOI: 10.17487/RFC3986. URL: <https://www.rfc-editor.org/info/rfc3986>.

- [11] Martin J. Dürst und Michel Suignard. *Internationalized Resource Identifiers (IRIs)*. RFC 3987. Jan. 2005. DOI: 10.17487/RFC3987. URL: <https://www.rfc-editor.org/info/rfc3987>.
- [12] Alexander Maedche und Boris Motik. „Repräsentations- und Anfragesprachen für Ontologien - eine Übersicht.“ In: *Datenbank-Spektrum* 6 (Jan. 2003), S. 43–53.
- [13] Brian McBride. „The Resource Description Framework (RDF) and its Vocabulary Description Language RDFS“. In: *Handbook on Ontologies*. Hrsg. von Steffen Staab und Rudi Studer. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, S. 51–65. ISBN: 978-3-540-24750-0. DOI: 10.1007/978-3-540-24750-0_3. URL: https://doi.org/10.1007/978-3-540-24750-0_3.
- [14] Deborah L. McGuinness und Frank van Harmelen. *OWL Web Ontology Language*. Stand: 13.12.2022. 2004. URL: <https://www.w3.org/TR/2004/REC-owl-features-20040210/#s1.3>.
- [15] The W3C SPARQL Working Group. *SPARQL 1.1 Overview*. Stand: 13.12.2022. März 2013. URL: <https://www.w3.org/TR/sparql11-overview/>.
- [16] Wolfgang Hesse. *Ontologie(n)*. Stand: 19.09.2022. Juli 2005. URL: <https://gi.de/informatiklexikon/ontologien>.
- [17] Heiner Stuckenschmidt. „Symbole, Objekte und Konzepte“. In: *Ontologien: Konzepte, Technologien und Anwendungen*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, S. 5–25. ISBN: 978-3-540-79333-5. DOI: 10.1007/978-3-540-79333-5_1. URL: https://doi.org/10.1007/978-3-540-79333-5_1.
- [18] Malte Rehbein. „Ontologien“. In: *Digital Humanities: Eine Einführung*. Hrsg. von Fotis Jannidis, Hubertus Kohle und Malte Rehbein. Stuttgart: J.B. Metzler, 2017, S. 162–176. ISBN: 978-3-476-05446-3. DOI: 10.1007/978-3-476-05446-3_11. URL: https://doi.org/10.1007/978-3-476-05446-3_11.
- [19] Cecilia Reyes-Pena und Mireya Tovar-Vidal. „Ontology: components and evaluation, a review“. In: *Research in Computing Science* 148.3 (2019), S. 257–265.
- [20] Robert Stevens, Carole A. Goble und Sean Bechhofer. „Ontology-based knowledge representation for bioinformatics“. In: *Briefings in bioinformatics* 1.4 (2000), S. 398–414.
- [21] Mariano Fernández-López und Asunción Gómez-Pérez. „Overview and analysis of methodologies for building ontologies“. In: *The Knowledge Engineering Review* 17.2 (2002), S. 129–156. DOI: 10.1017/S0269888902000462.
- [22] Natalya Noy und Deborah McGuinness. „Ontology Development 101: A Guide to Creating Your First Ontology“. In: *Knowledge Systems Laboratory* 32 (Jan. 2001).
- [23] Asuncion Gomez-Perez und Mari Carmen Suárez-Figueroa. „NeOn Methodology for Building Ontology Networks: a Scenario-based Methodology“. In: (Jan. 2009).

- [24] Aldo Gangemi und Valentina Presutti. „Ontology Design Patterns“. In: Mai 2009, S. 221–243. DOI: 10.1007/978-3-540-92673-3_10.
- [25] Patricia M. C. Campos u. a. „Finding reusable structured resources for the integration of environmental research data“. In: *Environmental Modelling & Software* 133 (2020), S. 104813.
- [26] Elena Pâslaru-Bontaş. „A Contextual Approach to Ontology Reuse“. Diss. 2007. URL: <http://dx.doi.org/10.17169/refubium-12546>.
- [27] Asunción Gómez-Pérez. „Evaluation of ontologies“. In: *International Journal of intelligent systems* 16.3 (2001), S. 391–409.
- [28] Janez Brank, Marko Grobelnik und Dunja Mladenic. „A survey of ontology evaluation techniques“. In: *Proceedings of the conference on data mining and data warehouses (SiKDD 2005)*. Citeseer Ljubljana Slovenia. 2005, S. 166–170.
- [29] Stephen A. White. „Introduction to BPMN“. In: *Ibm Cooperation 2.0* (2004), S. 0.
- [30] Marco Rospocher, Chiara Ghidini und Luciano Serafini. „An ontology for the business process modelling notation“. In: *Frontiers in Artificial Intelligence and Applications* 267 (Jan. 2014), S. 133–146. DOI: 10.3233/978-1-61499-438-1-133.
- [31] Amina Annane, Nathalie Aussenac-Gilles und Mouna Kamel. „BBO: BPMN 2.0 Based Ontology for Business Process Representation“. In: Sep. 2019. DOI: 10.34190/KM.19.113.
- [32] Thomas Off. „Durchgängige Verfolgbarkeit im Vorfeld der Softwareentwicklung von E-Government-Anwendungen : ein ontologiebasierter und modellgetriebener Ansatz am Beispiel von Bürgerdiensten“. doctoralthesis. Universität Potsdam, 2012.
- [33] Andreas Schminck, Rami-Habib Eid-Sabbagh und Mathias Weske. „eGovernment Process Knowledge Ontology - Business Process Knowledge Interdependencies in the Public Administration“. In: *GI-Jahrestagung*. 2013.
- [34] Max Raupach. Stand: 14.03.2023. URL: <https://processkg.simplex.fmi.uni-jena.de/>.
- [35] Cynthia Matuszek u. a. „An Introduction to the Syntax and Content of Cyc.“ In: Jan. 2006, S. 44–49.
- [36] Liliana Cabral, Barry Norton und John Domingue. „The Business Process Modelling Ontology“. In: *Proceedings of the 4th International Workshop on Semantic Business Process Management*. SBPM '09. Heraklion, Greece: Association for Computing Machinery, 2009, S. 9–16. ISBN: 9781605585130. DOI: 10.1145/1944968.1944971. URL: <https://doi.org/10.1145/1944968.1944971>.
- [37] Craig Schlenoff u. a. *The process specification language (PSL) overview and version 1.0 specification*. US Department of Commerce, National Institute of Standards und Technology, 2000.

-
- [38] Armin Haller, Eyal Oren und Paavo Kotinurmi. „m3po: An Ontology to Relate Choreographies to Workflow Models“. In: Okt. 2006, S. 19–27. DOI: 10.1109/SCC.2006.65.
- [39] Claudio Masolo u. a. *Ontology Library. WonderWeb Deliverable D18 (ver. 1.0, 31-12-2003)*. 2003.
- [40] Heinrich Herre u. a. „General Formal Ontology (GFO) - A Foundational Ontology Integrating Objects and Processes [Version 1.0]“. In: (Juli 2006).
- [41] Föderale IT-Kooperation. Stand: 26.01.2023. URL: <https://fimportal.de/>.
- [42] Föderale IT-Kooperation. Stand: 13.03.2023. URL: <https://fimportal.de/glossar>.
- [43] Mark A. Musen. „The protégé project: a look back and a look forward“. In: *AI matters* 1.4 (2015), S. 4–12.
- [44] Amina Annane. Stand: 14.03.2023. URL: https://github.com/AminaANNANE/BBO_BPMNbasedOntology/blob/master/BBO.owl.
- [45] The Apache Software Foundation. *Apache Jena*. Stand: 14.03.2023. URL: <https://jena.apache.org/>.
- [46] Koordinierungsstelle für IT-Standards. *XProzess*. Stand: 14.03.2023. URL: <https://www.xrepository.de/details/urn:xoev-de:mv:em:standard:xprozess>.
- [47] Koordinierungsstelle für IT-Standards. *XDatenfeld*. Stand: 14.03.2023. URL: <https://www.xrepository.de/details/urn:xoev-de:fim:standard:xdatenfelder>.
- [48] The Apache Software Foundation. *Apache Jena Fuseki*. Stand: 14.03.2023. URL: <https://jena.apache.org/documentation/fuseki2/>.
- [49] Jonas Hoyer. Stand: 17.03.2023. URL: https://git.rz.uni-jena.de/fusion/teaching/thesis/hoyer_jonas_bachelor_thesis/-/tree/main/query_results.

Abbildungsverzeichnis

2.1. Semantic Web Stack [7]	11
2.2. XML-Dokument	11
2.3. Allgemeine Syntax einer URI	12
2.4. Beispiel für ein grafisches Datenmodell in RDF	14
2.5. Iterativer Ablauf von CLeAR	20
2.6. Ablauf Ontology Evaluation Method	22
4.1. Zusammenhang von Task, Activity, FlowNode und SequenceFlow	34
4.2. Ausschnitt aus BBOs Resource Taxonomie	34
4.3. Konzeptionelles Modell unserer Ontologie	35
5.1. Klassenhierarchie unserer Ontologie	40
5.2. Erstellung eines Axioms anhand der Relation <i>has_process</i>	41
5.3. Hierarchie der Object Properties unserer Ontologie	42
5.4. Erstellung eines Axioms anhand der <i>Data Property has_leikaID</i>	43
5.5. Hierarchie der Data Properties unserer Ontologie	43
6.1. Pitfall 11: Fehlende Domain/Range bei Properties	52
6.2. Pitfall 34: Nicht-deklarierte Klasse	52
6.3. Pitfall 35: Nicht-deklarierte Beziehung	52
6.4. Pitfall 41: Keine Lizenz festgelegt	53

Tabellenverzeichnis

2.1. OWL-Subklassen [14]	15
2.2. Ontologie-Eigenschaften für spezielle Anwendungsszenarien [26]	23
4.1. Zusammenfassung wichtiger Informationen der ausgewählten Ontologien	31
4.2. Überblick über die Bewertungskriterien für jede Ontologie	33
4.3. Definitionen relevanter Begriffe [42]	37
4.4. Vergleich unserer Kompetenzfragen mit BBO	38
5.1. <i>Domain</i> und <i>Range</i> der benötigten <i>Data Properties</i>	42

Selbstständigkeitserklärung

Ich erkläre, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Seitens des Verfassers bestehen keine Einwände die vorliegende Bachelorarbeit für die öffentliche Benutzung im Universitätsarchiv zur Verfügung zu stellen.

Ort, Abgabedatum, Unterschrift des Verfassenden