Technische Universität Dresden

Fakultät Bauingenieurwesen
Institut für Bauinformatik
Prof. Dr.-Ing. habil. Karsten Menzel
Prof. Dr.-Ing. Raimar J. Scherer

Diplomarbeit

zur Erlangung des akademischen Grades
Diplomingenieur für Bauingenieurwesen

Erarbeitung einer Methode zur Transformation semantischer Daten in Brückenmodellen vom IFC Format in eine Ontologie

(Development of a method to transform semantic data of bridge models from IFC format into an ontology)

Taras Kozak

Hochschullehrer: Prof. Dr.-Ing. Raimar J. Scherer
Betreuer: Dr.-Ing. Peter Katranuschkov, Dipl.-Ing. Al-Hakam Hamdan

Dresden, 16. April 2019

Erklärung zur Anfertigung der Diplomarbeit

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ähnlicher Form einer anderen Prüfungsbehörde vorgelegt und ist auch noch nicht veröffentlicht worden.

Ort, Datum

Unterschrift

Abstract

Das Hintergrundschema des IFC-Datenmodells sieht kaum den Einsatz der Herleitung des impliziten Wissens vor. Weitherin wurde das IFC-Format nicht für die Findung und Ableitung neuer Informationen aus dem Datenmodell ausgelegt. Dies hat ein paar Arbeiten mit dem Einsatz von Ontologien zur Folge gehabt. Dank der ersten effizienten Ergebnisse, ist das Interesse an deren Einsatz im Bauwesen zunehmend gestiegen.

Allerdings fehlt dem Brückenbau noch der IFC-Standard, weshalb die Modellierungen mit sehr geringem semantischen Hintergrund laufen. Die Brückenmodelle verfügen größtenteils nur über Geometrien und wenige Meta Daten. Deshalb können weitere Bearbeitungsmöglichkeiten erst durch manuelle Manipulationen gewährleistet werden.

Der Fokus dieser Diplomarbeit liegt in der semantischen Verarbeitung der IFC- Brückenmodelle mittels Ontologien unter dem Einsatz der OWL-Websprache. Es wurde die terminologische Komponente (TBox) für semantisch relevante Bauelemente von Brücken der Spannbeton- und Verbundbauweise erstellt. Desweiteren wurden die statischen und räumlichen Relationen, sowie deren Attribute erstellt. Die Einstellung der spezifischen Relationseigenschaften hat die Ergebnisse der Schlussfolgerung durch die Herleitung der nützlichen Tripel deutlich vergrößert. Die Attribute sind an die DIN 1076 angelehnt, was den Einsatz bei der Brückenprüfung und -sanierung ermöglicht. Abschließend wurden die SWRL-Regeln mit weiteren SPARQL-Abfragen von Informationen durchgeführt.

Der Einsatz der OWL als Webstandard, weist ein Potential und die Möglichkeit für Verbesserungen der Interoperabilität auf. Weitere Untersuchungen des Linked Data Einsatzes sind sinnvoll.

Abstract

The background scheme of the IFC data model hardly provides for the use of deriving implicit knowledge. Furthermore, the IFC format was not designed to find and deduce new information from the data model. This has resulted in some work on the use of ontologies. Thanks to the first efficient results, interest in their use in construction has increased.

However, bridge building still lacks the IFC standard, which is why the modeling runs with very little semantic background. For the most part, the bridge models have only geometries and a few meta data. Therefore, further editing options can be guaranteed only by manual manipulation.

The focus of this diploma thesis is based on the semantic processing of IFC bridge models by means of ontologies using the OWL web language. The terminological component (TBox) for semantically relevant components of bridges of prestressed concrete and composite construction was drawn up. Additionally, the static and spatial relations as well as their attributes were created. The adjustment of the specific relation properties has significantly increased the results of the conclusion by the derivation of the useful triples. The attributes are based on DIN 1076, which enables use in bridge testing and rehabilitation. Finally, the SWRL rules were executed with further SPARQL queries of information.

The use of the OWL as a web standard has a potential and the potential for improving interoperability. Further investigations of the Linked Data application make sense.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listenverzeichnis

Abkürzungsverzeichnis

3D	dreidimensional
ARGE	Arbeitsgemeinschaft ARGE BIM4INFRA2020
BRIDGE–Ontologie	Die Erarbeitete im
BIM	Building Information Modelling
BMVI	Bundesministerium für Verkehr und digitale Infrastruktur
bSI	bildingSMART International
IFC	Industry Foundation Classes
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
OGC	Open Geospatial Consortium
OWL	Web Ontology Language
SPARQL	Web Ontology Language
SWRL	Web Ontology Language
usw.	und so weiter
z. B.	zum Beispiel

Glossar

BRIDGE–Ontologie Erarbeitet im Rahmen dieser Arbeit Ontologie für Beschreibung von Elementen in Brückenbauwerken

DatatypeProperty Eine abstrakte Beziehung (Rolle), die Individuen mit Datenwerten verbindet. Siehe Kapitel 2.5.1.

Infra Room Arbeitsgemeinschaft ARGE BIM4INFRA2020

Meta-Daten Metadaten sind Daten über Daten.

ObjectProperty Eine abstrakte Beziehung (Rolle), die Individuen miteinander verbindet. Siehe Kapitel 2.5.1.

OWL Die Web Ontology Language ist eine Sprache, die über eine wohldefinierte Semantik verfügt und für die Modellierung der Ontologien konzipiert wurde.

RDF Resource Description Framework ist eine formale Sprache für die Beschreibung strukturierter Informationen [2].

Reasoner Reasoner ist Teil von Software, die im Stande ist logische Konsequenzen von behaupteten Fakten und Axiomen zu generieren.

SWRL Semantic Web Rule Language ist eine Sprache für die Erstellung von logischen Regeln.

Schlussfolgerung Schlussfolgerung ist eine Anwendung von Ableitungsregeln auf formale Semantik, um aus bestehenden Aussagen neue zu generieren [63].

URI Uniform Resource Identifier ist einheitlicher Bezeichner für Ressourcen, der nach einer bestimmten Standard-Syntax zusammengesetzt wird. Diese Nummer ist eineindeutig, sodass jede betrachtete Ressource eineindeutig ansprechbar ist [1].

W3C Das World Wide Web Consortium ist das Gremium für die Standardisierung der Technologien im World Wide Web.

XML eXtensible Markup Language. XML ist eine Markup-Sprache, die für die Anreicherung von Text-Dokumenten mit der zusätzlichen Informationen verwendet wird [2]

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Der Begriff BIM beschreibt eine Methode, die eine optimierte Zusammenarbeit zwischen mehreren Projektbeteiligten über alle Projektphasen gewährleistet. Die BIM-Methode hat sich als ein vielversprechender Ansatz gezeigt, sodass ihre Entwicklung und Implementierung in Deutschland gefordert und gefördert wird. Durch das Bundesministerium für Verkehr und digitale Infrastruktur (BMVI) wird ab Ende 2020 der Einsatz der BIM-Methode bei allen neu zu planenden Infrastrukturprojekten verbindlich einzusetzen [35].

Die Interoperabilität soll in der BIM-Methode mit Hilfe von speziell dafür entwickelten von buildingSMART Datenformat IFC *(Industry Foundation Classes)* gewährleistet werden. Der IFC-Standard für Brücken ist bereits in der Entwicklung, wobei die Vorabzug-Version von IFC-Bridge *IFC4.2* schon veröffentlicht wurde [47] [53].

Mit der offiziellen Einführung der IFC-Bridge wird es möglich auch in den Brückenmodellen den semantischen Hintergrund mitzubenutzen. Jedoch, als ein objektorientiertes Datenmodell, strukturiert IFC die Daten hauptsächlich für den Austauschzweck, und weniger fürs Verstehen des Wissens in der Domäne. Die Informationen werden durch relativ komplexe Strukturen repräsentiert [23], was die Verarbeitung der semantischen Informationen im IFC-Datenmodell erschwert.

Durch die Entwicklung der Semantic Web Technologien und steigender Standardisierung im Web kommen die semantische Anwendungen in mehreren Bereichen öfter zum Einsatz. Sie können die effizientere Datenverarbeitung gewährleisten, indem die Informationen in maschinenlesbarer Form dargestellt werden. Dies ermöglicht mittels logischen Schlussfolgerungen das neue implizite Wissen zu gewinnen. Dabei steigt das Interesse an Linked Data Prinzipien, die die Möglichkeit geben, Daten aus mehreren Domänen untereinander zu verknüpfen und von semantischen Abfragen zu profitieren. Da im Baubranche das Problem der Datenheterogenität besteht, ist so ein Prinzip vom Interesse.

1.2 Problemstellung

Mitte der 90-er Jahren wurde von verschiedenen Interessengruppen des Baubereichs eine Initiative bezüglich der Interoperabilität im Bauwesen ergriffen. Das Ziel war die Erarbeitung eines programmunabhängiges Formats für einen einheitlichen Datenaustausch. Die Erstellung eines komplett neuen Austauschstandards explizit für Baubereich wurde jedoch von vielen als zu langwierig empfunden. Deswegen wurde beschlossen das neue Format mittels der EXPRESS-Modellierungssprache auf der Basis von STEP-Standard *(Standard for the exchange of product model data)* zu entwickeln. Mit der schon existierten ISO-Normierung des STEP-Standards (DIN EN ISO 10303) konnten die Umstände bei der Standardisierung während des aufwändigen bürokratischen Verfahrens vermieden werden [7]. Abschließend wurde das programmunabhängige Format mit der Entwicklung des IFC-Standards *(Industry Foundation Classes)* auf der STEP-Basis begonnen.

Eines der Ziele bei der Erarbeitung von IFC war die Erstellung des semantischen Hintergrundes, der als ein Kernbestandteil einer „intelligenten“ Bauwerksinformationsmodellierung ist [7]. Es gibt viele Konzepte in dem IFC-Datenmodell, welche es ermöglichen das Bauwerk semantisch zu beschreiben. Trotz einer großen Anzahl von Konzepten im IFC-Datenmodell, kann allerdings der aktuelle Bedarf an Semantik innerhalb von IFC nicht abgedeckt werden. Des Weiteren sind diese Informationen nicht formal und können auf verschiedene Art und Weise implizit repräsentiert werden, was zu Zweideutigkeit führen kann [24].

Viele nützliche Relationen und Attribute, die bereits explizit im IFC-Datenmodell definiert sind, können schwer in der Praxis abgerufen und verwendet werden. Darüber hinaus sind die IFC-Daten wegen des IFC-Schemas limitiert, welches nicht flexibel genug für die Integration und Verarbeitung der Daten aus externen Quellen ist [25].

Der semantische Teil des Schemas im IFC wird durch die Einschränkungen seines Hintergrundschemas auf der STEP-Basis begrenzt. Da bei der Erstellung des STEP-Standards nur ein schwacher mathematischer Logikhintergrund berücksichtigt wurde, beschränkt dies dementsprechend die Verarbeitung der semantischen Informationen im IFC-Datenmodell. Da das Schema des STEP-Formates nicht dafür ausgelegt wurde, ist die Erweiterung vom IFC-Schema auf zusätzliche Konstrukte für die Wissensverarbeitung folglich nicht möglich [23].

Außerdem wurde das IFC-Format hauptsächlich für den Austausch konzipiert. Dabei wurde der Einsatz von Informationsabfragen nicht berücksichtigt. Dies macht das Durchsuchen der Informationen im IFC-Datenmodell kompliziert [26].

Die Methoden der Wissensherleitung und des Einsatzes von einfachen Regeln stehen im Rahmen von IFC nicht zur Verfügung [4]. Allerdings bieten die Ontologien dank der aussagekräftigen Logik in der Basis wirksame Mechanismen für die Herleitung des impliziten Wissens. Von besonderer Bedeutung ist der Einsatz von Ontologien zusammen mit den Linked Data Prinzipien, welche eine offene Umgebung für die Freigabe, Integration, sowie Datenverlinkung aus verschiedenen Domänen anbieten [23]. Dabei werden die Daten einheitlich mittels fürs Web empfohlener Weltstandards, wie OWL und RDF, repräsentiert.

Die oben genannten Möglichkeiten haben auch das Interesse der Bauindustrie geweckt. Sie stellen eine deutliche Verbesserung der Wissensverarbeitung und eine Wissensherleitung dar. Die Datenrepräsentation mittels der Weltstandards des World Wide Web Consortiums (W3C) ermöglicht die Verwendung der eigenen kompatiblen Standards. Von Interesse ist die SPARQL-Sprache, die das Finden und Abfragen von Informationen aus verschiedenen ggf. externen Datenquellen ermöglicht.

1.3 Zielsetzung der Arbeit

Ziel der Arbeit ist die Erarbeitung einer Methode für die semantische Verarbeitung von IFC-Brückenmodellen.

Seit einigen Jahren, gewinnen das Semantic Web und die Linked Data Technologien zunehmend die Aufmerksamkeit als Wissensmodellierungsmethode im Bauwesen. Einige Prototypen wurden bereits entwickelt und sind unter folgender Liste einsehbar [27].

Es soll überprüft werden, was die semantischen Technologien können, welche Eigenschaften sie besitzen und welche Vor- und Nachteile diese mit sich bringen. Für den Bereich von Brücken existieren noch keine terminologischen Komponenten. In meiner Diplomarbeit werde ich diese Komponenten erstellen und die Methoden der Wissensherleitung überprüfen.

Zur Durchführung wird ein Brückenmodel im IFC-Format in eine Ontologie konvertiert. Am Modell wird kontrolliert, ob die semantische Verarbeitung möglich und unter welchen Bedingungen diese durchgeführt werden kann.

1.4 Aufbau der Arbeit

Die Diplomarbeit besteht aus neun Kapiteln. Das zweite Kapitel besteht aus den theoretischen Grundlagen für die Wissensrepräsentation, den Semantic Web Technologien, den Ontologien, sowie der OWL-Sprache. Im dritten Kapitel wird der aktuelle Stand der Technik und Forschung von BIM in der Kombination mit Brückenbau, sowie im Kontext von Semantic Web vorgestellt. Im folgenden vierten Kapitel wird die Erarbeitung der BRIDGE–Ontologie und ihre Möglichkeiten ausführlich erklärt. Anschließend werden im fünften Kapitel die Regeln vorgestellt, die als weiteres Werkzeug für Logische Folgerungen dient. Im sechsten Kapitel werden verschiedene Abfragen vorgestellt, welche die Funktionstüchtigkeit der Ontologie darstellen. Das neunte Kapitel dient der Zusammenfassung und dem Ausblick.

2 Theoretische Grundlagen

Für die Erstellung des Zusammenhangs zwischen dem Thema der Arbeit müssen die theoretische Grundlagen bezüglich Semantic Web Technologien erklärt werden. Bevor auf die Semantik und Repräsentationssprachen von Ontologien eingegangen wird, soll zuerst erklärt werden, wie Informationen und Daten vom Computer interpretiert und verarbeitet werden.

Bemerkung: *Im Gebiet der Informatik und Computertechnik wurden viele Innovationen im internationalen Raum entwickelt. Dabei wurde vorwiegend die englische Sprache als lingua franca verwendet. Des Weiteren ist die Syntax sehr vieler Programmiersprachen auf Englisch.*

Die Übersetzungen entsprechen gelegentlich nicht ganz der Bedeutung des englischen Begriffes, weswegen eine Zweideutigkeit auftreten kann. Damit die Informationen eindeutig verstanden werden, ist es sinnvoll sich auf ein Namenssystem zu einigen. Daher werden einige Begriffe im Rahmen dieser Arbeit auf Englisch verwendet. Zu dem Begriff wird bei dem ersten Gebrauch im Text eine Erklärung, sowie eine Übersetzung vorgestellt. Weiterhin sind diese Begriffe auch im Glossar vorhanden.

2.1 Information und Wissen

Unter Daten versteht man Informationen verschiedener Arten, wie z. B. Zahlen, Wörter, Texte, Video bzw. Audiodateien usw. Wenn man beispielsweise ein Zahl betrachtet, kann sie in Abhängigkeit vom Zusammenhang verschiedenste Bedeutungen annehmen, wie z. B. eine Koordinate, ein Abstand, ein Gewicht, ein Alter, eine Last usw. D. h. die Daten nehmen völlig andere Bedeutungen an in verschiedenen Kontexten, sodass die Interpretation sehr stark vom Kontext abhängt, in dem sie verwendet werden [1].

Der Computer kann die Informationen sehr gut speichern, aber das Wissen nicht interpretieren, was eine Weiterverarbeitung nicht möglich macht.

Beispiel: Es wurde in einem Wohnhausprojekt eine Zahl „-70.5“ mit Hilfe des Computers in einer Datei erkannt. Ohne jeglichen Kontext, erschließt sich kaum weiteres Wissen aus dieser Information. Das Minus Zeichen kann bedeuten, dass es sich beispielsweise um eine Höhenquote oder eine negative Drucklast handeln. Falls ein kleiner Buchstabe „m“ zusätzlich interpretiert wird, kann vermutet werden, dass „m“ die Meter-Einheit gemeint ist. Wenn die großgeschriebene Buchstabenkombination „UK“ davor stände, könnte geschlussfolgert werden, dass es sich um die Unterkante eines Bauteils handelt. Unter der Annahme, dass die Zahl sich auf eine relative Höhenquote bezieht, kann vermutet werden, dass die „-70.5m“ eine Unterkante der Bohrpfahlgründung bezeichnet. Da höchstwahrscheinlich eine so tiefe relative Höhenquote auf die Unterkante keines anderen Bauteils verweisen würde, bedeutet das, dass die Annahme nicht verkehrt ist. Des Weiteren, lässt sich eine Aussage über den kaum tragfähigen Boden treffen, falls so eine Bohrpfahlgründung bei einem typischen Wohnhaus eingesetzt wird.

Anhand dieses Beispiels kann man sehen, dass ein fehlender Kontext selbst bei Menschen eine korrekte Interpretation erschwert. Falls der nötige Kontext vorhanden ist, können viele andere Informationen geschlussfolgert werden. Im Vergleich zu Computern, besitzen die Menschen die Fähigkeit aus der geschriebenen oder gesprochenen Information die offensichtliche, und verdeckte Bedeutung sofort zu erkennen. Damit die Computer dies trotzdem können, gibt es zwei Optionen: entweder sollen die Informationen mit Hilfe von Technologien der Künstlichen Intelligenz (KI) analysiert werden, um die bedeutungstragenden Informationsglieder zu erkennen, oder die zusätzlichen Informationen sollen in Form von beschreibenden Attributen hinzugefügt werden [1].

2.1.1 Computerverarbeitung von Informationen

In der Liste 1 sind die Informationen über ein Bauelement aufgeführt. Wir Menschen können aus dem folgenden Text sofort die Informationen interpretieren und sagen, dass es sich um eine Stahlbetonstütze A27 mit der Betongüte C30/37 handelt, die sich im Erdgeschoss im Bauabschnitt A befindet. Für den Computer ist das nicht möglich.

Liste 1: Was sehen Menschen

```
Stütze
A27
Erdgeschoss
Bauabschnitt A
Stahlbeton
C30/37
```

Die Buchstaben, Zahlen, sowie alle andere Symbole, können vom Computer nur im Sinne der binären Codierung „verstanden“ werden. Der Computer setzt den Text nach einer angegebenen Codierung in einen binären Code um, in welchem die Darstellung von Zeichen durch Menschen festgelegt wurde. Die Liste 1 ist für den Computer mit der Liste 2 vergleichbar, wo jedes Zeichen auch nach einer Codierung einen eigenen binären Code besitzt.

Liste 2: Darstellung beim Computer

```
♢♣♢♡♠▲
♦△□
▲△♢★▲♢♠▽■♢♢
■♦♢★■▽♯♣★ ⋱ ♡♦
♢♡♦∠△■♢♡▽★
♠□△♦∠
```

Nach einer Definition von syntaktischen XML-Tags und weiteren XML-Markierungen des Inhaltes kann der Computer die Art der Informationen schon unterscheiden, siehe Liste 3.

Liste 3: Was sehen Menschen nach der XML-Markierung

```
<bauelement>
        <bauteil>Stütze</bauteil>
        <position>A27</position>
        <geschoss>Erdgeschoss</geschoss>
        <bauabschnitt>Bauabschnitt A</bauabschnitt>
        <material>Stahlbeton</material>
        <betonguete>C30/37</betonguete>
</bauelement>
```

Die Inhalte zwischen den Tags sind für den Computer immer noch nicht „verständlich“, siehe Liste 4.

Liste 4: Scheinbare Darstellung beim Computer nach der XML-Markierung

```
<bauelement>
<bauteil>♢♣♢♡♠▲<\bauteil>
<position> ♦△□ <\position>
<geschoss> ▲△♢★▲♢♠▽■♢♢ <\geschoss>
<bauabschnitt> ■♦♢★■▽♯♣★ ˙·. ♡♦ <\bauabschnitt>
<material> ♢♡♦∠△■♢♡▽★ <\material>
<betonguete> ♠□△♦∠ <\betonguete>
<\bauelement>
```

Die Namen von Tags haben für den Computer trotzdem keine Bedeutung. Die Inhalte in Tags zwischen zwei einfachen Einführungszeichen werden bloß vom Computer als eine Zeichenkette verstanden. Der Computer erkennt die Paare derselben geöffneten und geschlossenen Tags und prüft die syntaktische Richtigkeit des Dokuments. Die Liste 4 ist für den Computer mit der Liste 5 vergleichbar.

Liste 5: Tatsächliche Darstellung beim Computer nach der XML-Markierung

```
< ▲△♢★▲♢ >
<♢★▲♢♠▽>♢♣♢♡♠▲<\♢★▲♢♠▽>
<♠□△♦∠> ♦△□ <\♠□△♦∠>
<■♢♡▽★> ▲△♢★▲♢♠▽■♢♢ <\■♢♡▽★>
<▽★♢♢♠□> ■♦♢★■▽♯♣★ ˙·. ♡♦ <\▽★♢♢♠□>
<♢★▲♢♠▽> ♢♡♦∠△■♢♡▽★ <\♢★▲♢♠▽>
<♠□△♦∠> ♠□△♦∠ <\♠□△♦∠>
<\ ▲△♢★▲♢>
```

Es lässt sich schlussfolgern: Damit der Computer die Informationen interpretieren und in der Zukunft operieren kann, muss die Semantik eingeführt werden.

2.1.2 Semantik vs. Syntax

Diese zwei Begriffe spielen bei der Definition von Sprachen eine zentrale Rolle. Unter der Syntax versteht man im Allgemeinen eine Menge von Regeln zur Strukturierung von Zei-

chen und weiteren Informationsgliedern, wie Zeichenkette oder Sätze. In der Informatik steht die Syntax auch im Sinne für die Menge von Regeln um die Informationen standardisiert aufzuschreiben. D. h. die Syntax bedeutet auf welche Art und Weise die Informationen geschrieben werden [2].

Der Begriff Semantik stammt aus dem Griechischen. Dieser wird in unterschiedlichen Kontexten sehr verschieden verwendet und kann am treffendsten mit dem deutschen Wort „Bedeutung“ umschrieben werden [2]. Der Gegenstand der Semantik ist die Untersuchung der durch sprachliche Zeichen bezeichneten Inhalte. Also, der Bedeutung von bestimmten Informationsgliedern [29]. In der Informatik versteht man unter dem Begriff Semantik die Bedeutung von Daten und Informationen [7].

Syntax steht also für die normative Struktur von Daten, welche erst durch die Semantik eine Bedeutung erhält [2].

4+)=(	**3+4=12**	**3+4=7**
syntaktisch falsch	syntaktisch richtig	syntaktisch richtig
semantisch falsch	semantisch falsch	semantisch richtig

Tabelle 1: Unterschied Semantik vs. Syntax. Beispiel aus [63]

2.1.3 Wissensrepräsentation

Das menschliche Wissen kann in verschiedenen Formen repräsentiert werden. Die Verteilung des menschliches Wissens nach [5] ist in der Tabelle 2 aufgeführt.

Kategorie	**Erklärung**
Prozedurales Wissen	Wie/Auf welche Weise soll etwas gemacht/gelöst werden? D. h. nach welchen Regeln, Strategien, Prozeduren.
Deklaratives Wissen	Enthält vorhandene Informationen (Details, Fakten) über ein konkretes Thema.
Meta-Wissen	Wissen über Wissen. Ermöglicht die Klassifizierung des Wissens nach der Art. Das beschleunigt das Auffinden der Informationen, indem irrelevantes Wissen für das Lösen des konkreten Themas am Anfang ausgelassen wird.
Ungenaues und unsicheres Wissen	Die Informationen, die unklare Angaben, Fakten usw. enthalten, wie z. B. klein, warm, groß. (anstatt eine quantitative Werte mit Einheiten zu enthalten)
Ontologisches Wissen	Enthält Konzepten, Relationen zwischen Konzepten, Eigenschaften der Relationen, Axiome, Einschränkungen. Das ontologische Wissen überlappt sich mit anderen Wissenstypen.

Tabelle 2: Arten des menschlichen Wissens [5]

Die möglichen Wissensrepräsentationssprachen werden als logikbasiert, regelbasiert oder visuell definiert [5]. Besonders sinnvoll ist die Anwendung von Ontologien, da sie anhand

von logikbasierten Wissensrepräsentationssprachen dargestellt werden. Ontologien unterstützen die Anreicherung mit Meta-Wissen, den Einsatz von regelbasierten Methoden und die logische Schlussfolgerung.

Bevor es mit dem Thema Ontologien weitergeht wird zunächst auf das Thema Semantic Web eingegangen

2.2 Semantic Web & Linked Open Data

Da das WEB für „Menschengebrauch“ ausgearbeitet wurde, sind die meisten Informationen im WEB nicht maschinenlesbar [5]. Dies macht das Finden der Informationen sehr aufwendig. Wie es schon im Kapitel 2.1.1 erläutert wurde, ist die fehlende Semantik ein entscheidendes Hindernis, die die Interpretation der Informationen mit Hilfe von Computern sehr aufwendig bzw. unmöglich macht.

Mit der Initiative von Tim Berners-Lee wurde der Begriff *Semantic Web* geboren und bis heute weiterentwickelt. Das Ziel von Semantic Web ist es durch Semantische Technologien die Informationen so zu behandeln, dass die Weiterverarbeitung mit Hilfe von Computern möglich wird [5]. Mit so einer Vorbereitung der Informationen, wird die Wiederverwendung der Informationen und die Interoperabilität zwischen verschiedenen Daten und Systemen erreicht. Dies ermöglicht eine Steigerung der Automatisierung durch die Übernahme eines größeren Arbeitsanteils der Informationsverarbeitung von Computern.

Unter dem Begriff Semantic Web werden die Technologien und Techniken der Wissensmodellierung zusammengefasst, dass dieses Wissen mit Hilfe von Computern verarbeitet werden kann [1]. Zu diesen Technologien gehören Wissensrepräsentationssprachen für Ontologien, Methoden und Werkzeuge zur deren Erstellung, Wartung und Anwendung. Dies ermöglicht das effiziente Finden der Informationen aus verschiedenen Quellen, derer Behandlung und Integration in andere Systeme, sowie die Inferenz des impliziten Wissens [2].

Ursprünglich wurden die Semantischen Technologien nur im WWW (World Wibe Web) eingesetzt. Später wurde das Potenzial dieser Technologie in vielen anderen Bereichen erkannt, wie z. B. Informatik, maschinelles Lernen, KI (künstliche Intelligenz), Medizin, Bauinformatik [63].

Auf der Abbildung 1 ist eine so genannte *„Semantic Web layer-cake“* (Semantic Web Kuchenschicht) dargestellt. Dabei verwenden die oberen Schichten die Implementierungen der unteren. Im Ursprung liegt die Verwendung von URIs. Darauf beruht die XML-Sprache die als die Meta-Sprache für die Repräsentation anderer Web-Sprachen dient. Mit RDF können Meta-Daten über Web-Ressourcen repräsentiert werden. Die Sprachen RDFS und OWL ermöglichen die Modellierung der Ontologien und mit der Anwendung von Regeln und Abfragen mittels der SPARQL-Abfragesprache können die Informationen effizient behandelt und gefunden werden [5].

Linked Open Data Linked Open Data (LOD) – Verknüpfte öffentlich zugängliche Daten, sin ein Teil des Semantic Webs. Linked Open Data ist eine Community-Bestrebung,

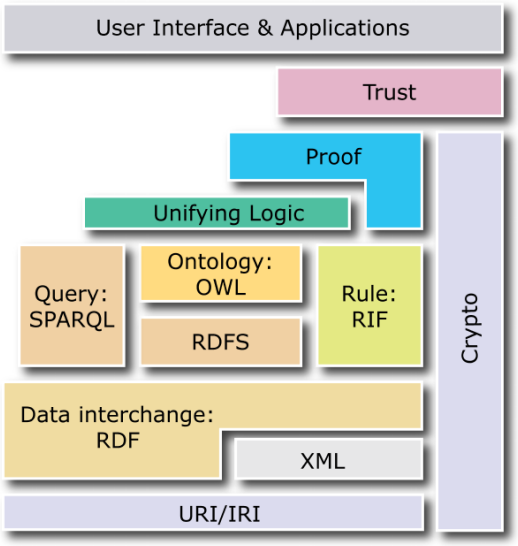


Abbildung 1: Tim Berners-Lee's Blick auf Semantic Web
Bildquelle: [66]

die großen Datensätze zu veröffentlichen und mittels semantischer Techniken miteinander zu verknüpfen versucht [54] [1]. Die Veröffentlichung der Daten in der maschinenlesbaren Form, schafft die Möglichkeit weitere Daten untereinander zu verknüpfen und von semantischen Abfragen zu profitieren [1].

Nach den Linked Data Prinzipien sollen die Daten per URIs und HTTP URIs identifizierbar sein, damit sowohl Menschen als auch Web-Agenten auf die Daten zugreifen und verweisen können. Eine URI sollte auf verwendbare Information verweisen, die mit den Standards RDF und SPARQL repräsentiert wird [1]. Je mehr Daten korrekt bereitgestellt und verknüpft werden, desto mehr Daten lassen sich effizient finden und wiederverwenden.

Die Datensätze werden nach Bereichen in einer LOD-Cloud gespeichert, wo laut `https://lod-cloud.net/` im Juni 2018 – 1.234 Datensätze in den verschiedensten Bereichen mit 16.136 Links existierten, die eine Anzahl von über 50 Milliarden RDF-Tripel aufwiesen.

Der Einsatz des Linked Data Prinzips ist für die Baubrancheinsofern von großen Interesse dass mehr Einheitlichkeit in die Datenheterogenität gebracht werden kann und eine bessere Interoperabilität gewährleistet werden kann.

2.3 Ontologien

Der Begriff Ontologie hat seinen Ursprung in der Philosophie. In der Informatik wird Ontologie oft mit der Definition nach T. R. Gruber erklärt:

> *An ontology is an explicit and formal specification of a conceptualization.*

Der Begriff wird folgendermaßen verstanden:

- Unter dem Wort *„conceptualisation“* ist ein abstraktes Modell eines Gebietes zu verstehen.

Es ist möglich über beliebige (physische und abstrakte) Objekte sowohl im Sinne konkreter Individuen zu sprechen, als auch im Sinne abstrakter Klassen. Wobei für die Erstellung einer Ontologie die Abstraktion für eine Unterteilung auf Klassen notwendig ist.

Ein Objekt kann entweder wie ein konkretes oder abstraktes Objekt betrachtet werden. Wenn man ein konkretes Objekt abstrahiert, wird es möglich ein abstraktes Modell dieses Objektes abzubilden.

Beispiel: Ein konkretes Individuum, siehe Liste 6, kann entweder, als ein abstraktes Objekt betrachtet oder in einem abstrakten Modell abgebildet werden, siehe Liste 7.

Liste 6: Beschreibung eines Individuums

```
- Stütze
- A27
- Erdgeschoss
- Bauabschnitt A
- Stahlbeton
- C30/37
```

Liste 7: Beschreibung eines Individuums

```
- Klassenname: Stütze
- Bauteil eines Bauwerks
- Dimension in der Längsrichtung ist vorwiegend viel größer als in zwei anderen
Richtungen
- Werden normalerweise Druckkräfte aufgenommen
- Die Aufnahme der Zugkräfte kann auch vorgesehen/gewährleistet werden
```

- Unter *„explicit specification“* ist eine Unterteilung in Klassen/Konzepte/Begriffe, Eigenschaften der Klassen, Attributen der Klassen, Eigenschaften der Eigenschaften, Beziehungen untereinander und Individuen zu verstehen.
- Das Wort *„formal“* bezeichnet die Voraussetzung, die bei der Beschreibung der oben genannten Begriffen, unter Verwendung der formalen Sprache (der Mathematik und Logik) gegeben sein kann, sodass die Implementierung dieser Ansätze bei Computern möglich wäre.

Es lässt sich zusammenfassen, dass Ontologie eine klassifizierte Darstellung einer Menge von Begriffen und Beziehungen zwischen einander ist und über eine klare Syntax und formale Semantik verfügt. Die Ontologie gibt eine gemeinsame Sprache und ein gemeinsames Verständnis für die Beschreibung von physischen, sowie abstrakten Objekten und Prozessen.

2.3.1 Semantische Netze

Semantische Netze sind eine Art der Wissenrepräsentationstechniken, die mittels gerichteter Graphen die Konzepte und Relationen zwischen Konzepten visuell in der Form eines Netzes darstellen [1], siehe Abbildung 2.

Dabei ist ein *Graph* eine abstrakte Struktur, die visuell eine Menge von Objekten zusammen mit den zwischen diesen Objekten bestehenden Beziehungen repräsentiert. Die Objekte werden als Knoten des Graphen, sowie die Beziehungen als Kanten des Graphen bezeichnet. Die Kanten können gerichtet oder ungerichtet sein [9]. Im Vergleich zu einem ungerichteten Graphen kann die Relation nur in eine Richtung verlaufen, siehe Abbildung 3.

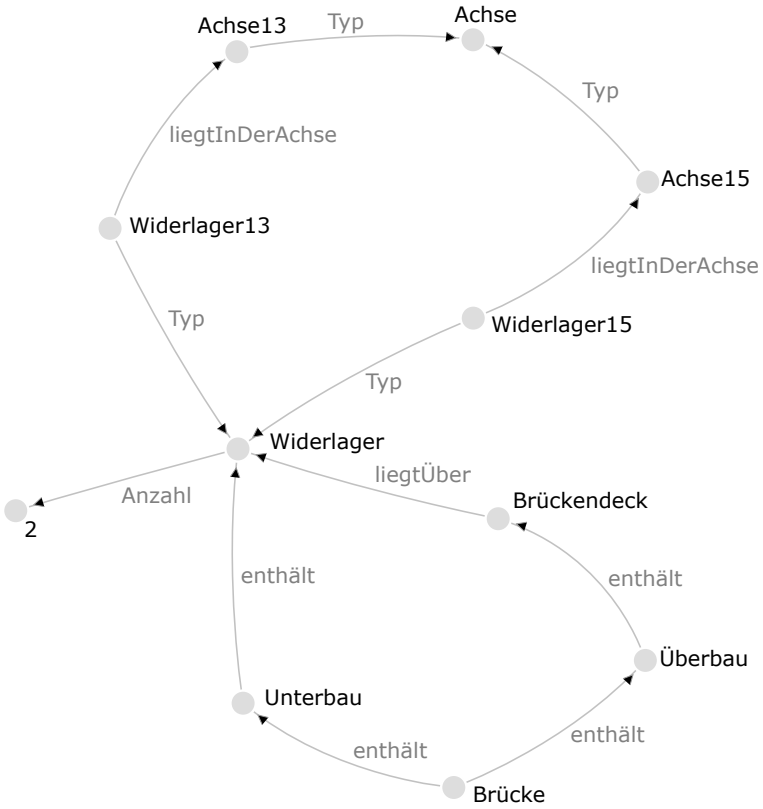


Abbildung 2: Beispiel eines semantischen Netzes

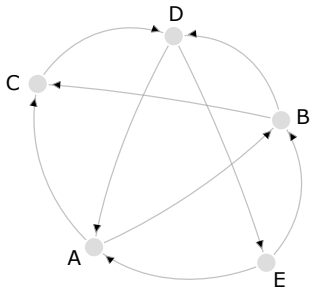


Abbildung 3: Beispiel eines gerichteten Graphen

Die Darstellung mittels Graphen ist sehr anschaulich und präzise, wobei von Menschen bloß sehr kleine Graphen mühelos verstanden werden können. Die großen Graphen mit Tausenden von Knoten und Kanten eigenen sich nicht zu zeichnerischen Repräsentation. Des Weiteren, ist die Graphendarstellung für die Verarbeitung im Computersystem ungeeignet [2].

Eine gängige Textdarstellung, die für die Computerverarbeitung taugt, ist die Verwendung der Tripel „Subjekt – Prädikat – Objekt“. Mit Hilfe der Serialisierung (Umwandlung

komplexer Datenobjekten in lineare Zeichenketten) kann der Graph in Bestandteile zerlegt werden und in der entsprechenden Reihenfolge abgespeichert werden. Die Serialisierung ist mittels RDF/XML Syntax, Notation 3 (N3), N-Triples und Turtle möglich [2]. Die Turtle-Syntax, sowie N3 und N-Triples, werden als menschenlesbar bezeichnet. Im Rahmen dieser Arbeit wird für die Beispiele die Turtle-Syntax benutzt.

Alle drei Teile eines Tripels werden durch ihre eineindeutige URI-Bezeichner (Uniform Resource Identifier) beschrieben. Nur das Objekt in einem RDF-Tripel kann gegebenfalls ein RDF-Literal sein. In der Liste 8 ist das Beispiel mit der Turtle-Syntax dargestellt. Am Anfang des Dokumentes werden die Präfixe deklariert. Dies ermöglicht die URIs abzukürzen und bessere Lesbarkeit zu erzielen.

Liste 8: Beispiel der Turtle-Syntax

```
@prefix : <http://www.semanticweb.org/taras.kozak/bridge#> .
:Gelaende :enthaelt :Bruecke .
:Bruecke :enthaelt :Unterbau .
:Bruecke :enthaelt :Ueberbau .
:Unterbau :enthaelt :Wiederlager .
```

2.3.2 Repräsentationssprachen von Ontologien

Die Ontologien können mittels verschiedener Sprachen modelliert werden, die über verschiedene Ausdrucksstärken verfügen. Die Sprachen können bedingt auf schwache und starke unterteilt werden. Zu schwachen gehören ER-Diagramme, UML-Modelle, RDFS usw. Zu den *starken* gehören die OWL-Sprache und ihre Vorgänger [32]. Die RDFS und OWL werden als Web Standards vom W3C-Konsortium empfohlen.

RDF *Resource Description Framework (RDF)* ist eine formale Sprache für die Beschreibung strukturierter Informationen [2]. Zuerst wurde RDF für die Beschreibung von Metadaten im Web konzipiert [3]. Heutzutage wird RDF in vielen Gebieten verwendet, da es eine Vielzahl von praktischen Werkzeugen dank der Unterstützung von RDF-Daten (sogenannte *RDF Stores* oder *Triple Stores*) ermöglicht [2].

Das RDF-Modell basiert auf Tripeln, die aus Subjekt, Prädikat und Objekt bestehen, siehe Abbildung 4. Eine Menge von RDF-Tripeln ergeben einen RDF-Graphen. Hierbei werden das Subjekt und das Objekt als Knoten betrachtet. Das Prädikat ist der Name der gerichteten Kante von Subjekt zu Objekt. Alle Ressourcen im RDF-Modell werden immer mit eineindeutigen URI-Bezeichner identifiziert. Dabei sind Prädikate immer URIs. Subjekte sind üblicherweise URIs, wobei sie auch unbenannte Knoten (aus dem Englischen - *blank nodes*) sein können. Objekte können entweder URIs, unbenannte Knoten oder Literale sein. Unter Literalen werden z.B. Texte, Zahlen, Datumsangaben verstanden [3].

URI *Uniform Resource Identifier (URI)* ist ein einheitlicher Bezeichner für Ressourcen, der nach einer bestimmten Standard-Syntax zusammengesetzt wird. Diese Nummer ist

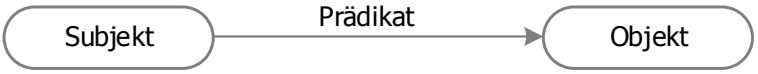


Abbildung 4: Beispiel der Triple-Repräsentation

eindeutig, sodass jede betrachtete Ressource (Subjekt, Prädikat oder Objekt) eindeutig ansprechbar ist [1].

Z. B. Im Rahmen dieser Arbeit hat die Instanz `ABUTMENT_ELEMENT_15` den folgenden URI, siehe Liste 9:

Liste 9: Beispiel eines URI-Bezeichners

```
http://www.semanticweb.org/taras.kozak/bridge#ABUTMENT_ELEMENT_15
```

RDFS *RDF-Schema (RDFS)* ist eine Erweiterungssprache zur Beschreibung der Klassen, Eigenschaften und Attributen von RDF-Ressourcen [3]. Mittels der RDFS kann die Hierarchie zwischen Klassen von Objekten und Eigenschaften in einem RDF-Modell geschaffen werden. Die Anordnung der formalen Beziehung ermöglicht zwischen Ressourcen die Erstellung von einfachen Ontologien [3].

Für die Modellierung der komplexeren Ontologien sind RDF und RDFS nicht ausdrucksstark genug. Daher wurde die ausdrucksstärkere OWL-Sprache entwickelt.

2.4 Die Web Ontology Language (OWL)

Die Web Ontology Language (OWL) ist eine Sprache, die über eine wohldefinierte Semantik verfügt und für die Modellierung der Ontologien konzipiert wurde. Die OWL, sowie RDF und RFDS, ist eine Spezifikation des World Wide Web Consortiums (W3C) [3]. Die aktuelle Version OWL2 ist seit Oktober 2009 ein W3C Standard und gilt bis jetzt als die W3C-Empfehlung [3].

Die OWL-Sprache wurde als Erweiterung der bisherigen Web-Standards XML, RDF und RDFS entwickelt. Mit der zunehmenden Ausdrucksmächtigkeit geht aber auch immer die zunehmende Komplexität einher. Die OWL-Sprache wurde gut in Bezug auf Komplexität angepasst, um die Entscheidbarkeit mit der hohen Ausdrucksfähigkeit erreichen zu können [1].

Es gibt drei verschiedene Teilsprachen der OWL: OWL Full, OWL DL und OWL Lite. Diese Teilsprachen unterscheiden sich hinsichtlich der Komplexität, Skalierbarkeit und Entscheidbarkeit. Nach [2] und [6] verfügen diese Teilsprachen über folgende Eigenschaften:

OWL Full

- sehr ausdrucksstark
- unentscheidbar
- enthält die Teilsprachen OWL DL und OWL Lite

- ist völlig semantisch und syntaktisch mit RDFS kompatibel
- wird von aktuellen Softwares nur bedingt unterstützt

OWL DL

- ausdrucksstark
- entscheidbar
- enthält die OWL Lite und ist die Teilsprache von OWL Full
- wird von aktuellen Softwares fast vollständig unterstützt

OWL Lite

- weniger ausdrucksstark
- entscheidbar
- ist die Teilsprache von OWL DL und OWL Full

Die Teilsprache OWL DL ist von besonderer Bedeutung, da sie ein Kompromiss zwischen der Ausdrucksstärke und Entscheidbarkeit ist. Sie wird detaillierter im Kapitel 2.5 beschrieben. Weiterhin wird kurz auf das Thema der Komplexitätsklassen eingegangen.

2.5 OWL DL

Die Teilsprache OWL DL basiert auf der Beschreibungslogik (die Abkürzung folgt aus dem Englischen *description logic* – DL). Die Sprachkonstrukte der OWL2 DL, die in ihrer Basis liegen, werden durch die Abkürzung $\mathcal{SHOIN}^{(\mathcal{D})}$ bezeichnet [22]. Eine maximale Erweiterung im Rahmen der Entscheidbarkeit kann zu $\mathcal{SROIQ}^{(\mathcal{D})}$ durchgeführt werden. Jede Buchstabe hat eine konkrete Bedeutung und laut [22] und [2] können sie folgend aufgeschlüsselt werden:

- $\mathcal{S}$ steht für die Beschreibungslogik $\mathcal{ALC}$ (*Attributive Concept Language with Complements*) zusammen mit Rollentransitivität
- $\mathcal{R}$ steht für komplexe Rolleninklusion (transitive, symmetrische, asymmetrische, reflexive, irreflexive Rollencharakteristiken, sowie disjunkte Rolle)
- $\mathcal{H}$ steht für Unterrollenbeziehung
- $\mathcal{O}$ steht für Nominale, die abgeschlossenen Klassen mit `owl:oneOf` Restriktionen
- $\mathcal{I}$ steht für inverse Rollen
- $\mathcal{N}$ steht für Zahlenrestriktionen
- $\mathcal{Q}$ steht für qualifizierende Zahlenrestriktionen
- $^{(\mathcal{D})}$ steht für die Datentypen

Im folgenden Kapiteln werden wichtigste Sprachkonstrukten der Teilsprache OWL DL vorgestellt. Die Informationen wurden der offiziellen W3C Spezifikation [56] entnommen.

2.5.1 Klassen, Rollen, Individuen

In OWL gilt eine klare Unterteilung von Ressourcen nach Klassen, Rollen und Individuen.

Eine *Klasse* ist ein Ressource, mit dem eine konkrete oder abstrakte Entität der realen Welt bezeichnet wird. Ein *Individuum* ist eine Instanz, die einer Klasse zugehörig ist. Eine *Rolle* ist ein Ressource, die eine verbindende Beziehung, analog wie eine gerichtete Kante in einem Graphen, zwischen Ressourcen darstellt [2], siehe Kapitel 2.3.1.

Die allgemeinste Klasse, die alles enthält ist die Klasse `owl:Thing`. Die Klasse `owl:Nothing` dagegen leer ist [2].

- `Wand`, `Decke` sind Klassen die Entitäten abstrakt beschreiben
- `WAND_W-13`, `DECKE_D-01` sind konkrete Individuen dieser Klassen
- `traegt` ist eine Rolle, mit der die Individuen verbunden werden können

Die Rollen werden in *abstrakte* und *konkrete* unterteilt. Mit einer *abstrakten* Rolle werden Individuen miteinander verbunden. Mit einer *konkreten* Rolle werden Individuen mit Datenwerten verbunden. Beide Rollenarten gehören zur `rdf:Property` [2].

Die abstrakten Rollen werden mit `owl:ObjectProperty` deklariert, während die konkreten mit `owl:DatatypeProperty`, siehe die Liste 10.

Liste 10: Beispiel. Deklaration und Anwendung der abstrakten und konkreten Rollen

```
# Deklaration der abstrakten Rolle
:traegt rdf:type owl:ObjectProperty .
# Verwendung der abstrakten Rolle
:WAND_W-13 :traegt :DECKE_D-01 .
# Deklaration der konkreten Rolle
:betonguete rdf:type owl:DatatypeProperty .
# Verwendung der abstrakten Rolle
:WAND_W-13 :betonguete "C30/37" .
```

Klassen werden in OWL mit `owl:class` und Individuen mit `owl:NamedIndividual` deklariert. Der Rollenbezug kann mittels `rdfs:domain` und `rdfs:range` explizit deklariert werden [2].

Weiter werden im Rahmen dieser Arbeit die Begriffe **Property** (Rolle), **ObjectProperty** (abstrakte Rolle) und **DatatypeProperty** (konkrete Rolle) benutzt.

2.5.2 Einfache Klassenbeziehungen

- `rdf:subClassOf`
  Auch wie in RDFS kann die Zugehörigkeit einer Klasse zu der Beziehung einer anderen `rdf:subClassOf` transitiv verstanden werden.
- `owl:disjointWith`
  Mit diesem Konstrukt werden zwei Klassen als disjunkt definiert. Das bedeutet, dass kein Individuum einer Klasse beiden Klassen zugeordnet werden kann [2].
- `owl:equivalentClass`
  Mit diesem Konstrukt kann explizit zugewiesen werden, dass eine Klasse gleichwertig einer anderen ist.

2.5.3 Beziehungen zwischen Individuen

- `owl:sameAs`
  Mit diesem Konstrukt kann explizit die Gleichwertigkeit eines Individuums zu einem anderen definiert werden.
  Z. B. `Stütze_Achse_3-A_Erdgeschoss` und `STUETZE_EG_ACHSE_3-A`
- `owl:differentFrom`
  Mit diesem Konstrukt kann explizit die Ungleichwertigkeit eines Individuums zu einem anderen definiert werden, siehe auch Kapitel 2.5.10.

2.5.4 Logische Konstruktoren auf Klassen

- `owl:intersectionOf`
  Das Konstrukt entspricht dem logischen Operator Konjunktion $\sqcap$. Dies umfasst alle Elemente, die zur gemeinsamen Schnittmenge der Klassen gehören.
- `owl:unionOf`
  Das Konstrukt entspricht den logischem Operator Disjunktion $\sqcup$. Dies umfasst alle Elemente, der betrachten Klassen.
- `owl:complementOf`
  Das Konstrukt entspricht dem logischen Operator Negation $\neg$. Dies umfasst alle Elemente, die nicht zu der entsprechenden Klasse gehören.

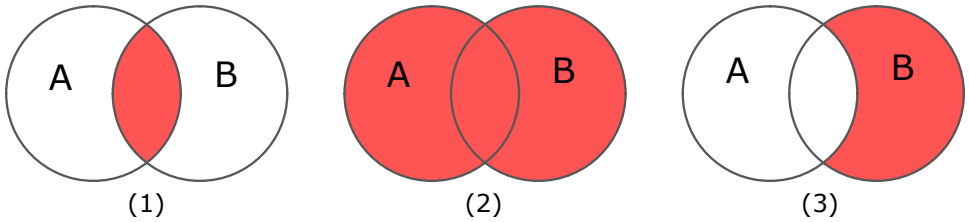


Abbildung 5: Visuelle Darstellung von: Konjunktion (1) A $\sqcap$ B; Disjunktion (2) A $\sqcup$ B; Negation (3) $\neg$ A

2.5.5 Rolleneinschränkungen

- `owl:allValuesFrom`
  Das Konstrukt entspricht dem logischen Allquantor $\forall$.
- `owl:someValuesFrom`
  Das Konstrukt entspricht dem logischen Existenzquantor $\exists$.
- `owl:cardinality` `owl:minCardinality` `owl:maxCardinality`
  Die Konstrukte erlauben die Verwendung von Kardinalitäten.

2.5.6 Rollenbeziehungen

Analog zu Klassenbeziehungen, siehe Kapitel 2.5.2, können für Properties folgende Eigenschaften definiert werden.

- `rdfs:subPropertyOf`
  Mit dem Konstrukt kann explizit die Zugehörigkeit einer Property ***A*** zu einer anderen ***B*** definiert werden, während die Anwendung der Property ***A*** die Anwendung der Property ***B*** impliziert, aber nicht umgekehrt. Beispiel, siehe Liste 12.
- `owl:equivalentProperty`
  Mit dem Konstrukt kann explizit die Gleichwertigkeit einer Property ***A*** zu einer anderen ***B*** definiert werden, während die Anwendung der Property ***A*** die Anwendung der Property ***B*** impliziert, und umgekehrt.
- `owl:inverseOf`
  Mit dem Konstrukt kann einer Property explizit eine andere inverse Property zugewiesen werden. Jede ObjectProperty kann eine entsprechende inverse ObjectProperty haben. D.h. wenn ein Individuum ***A*** sich mit einer Property auf ein anderes Individuum ***B*** bezieht, wird die inverse Property den Bezug zwischen dem Individuum ***B*** auf ***A*** erstellen. Z. B. `bears` und `restOn`.
- `owl:disjointOf` Mit dem Konstrukt kann einer Property explizit eine andere disjunkte Property zugewiesen werden. Falls für das Paar von Individuen **A** und **B**, ist der Fall wenn: Property1(**A**, **B**) und Property2(**A**, **B**) nicht möglich, sollen die Properties als disjunkt definiert werden.

2.5.7 Rolleneingeschaften

Die OWL2 ermöglicht die folgenden Eigenschaften der ObjectProperties:

- `owl:FunctionalProperty` Diese Eigenschaft hat die Beschränkung, dass ein Individuum nur eine ausgehende Relation besitzen kann. ***Beispiel:*** ein simples Bauteil kann nur aus einem Material bestehen. Bei der Einstellung der `hatMaterial` ObjectProperty als FunctionalProperty mit der Anordnung zu einem Individuum `STAHLBETON_C25/30` und gleichzeitig zu einem Individuum `STAHLBETON_C30/37`, wird es ohne Unique Name Assumption, siehe Kapitel 2.5.10, impliziert das diese Individuen dasselbe Individuum ist. Wenn diese Individuen explizit als `owl:differentFrom` definiert werden, wird das zur Inkonsistenz in der Ontologie führen [57].
- `owl:InverseFunctionalProperty` Diese Eigenschaft besitzt den selben Hintergrund wie `owl:FunctionalProperty`, jedoch kann das Individuum nur so eine **eingehende** Relationen besitzen.
- `owl:ReflexiveProperty` Die Eigenschaft ermöglicht die Anordnung der Relation von einem Individuum zu sich selbst.
- `owl:IrreflexiveProperty` Die Eigenschaft verbiete die Anordnung der Relation von einem Individuum zu sich selbst.
- `owl:SymmetricProperty` Bei der Anordnung ObjectProperty mit der Eigenschaft, wird definiert, wenn ein Individuum **A** durch eine ObjectProperty mit eines anderen

Individuums **B** verbunden ist, bedeutet das, dass die gleiche Verbindung von **B** zu **A** durch dieselbe ObjectProperty impliziert wird.
Z. B. `SEIL istVerbundenMit BRÜCKENDECK`.

- `owl:AsymmetricProperty` Mit der Eigenschaft, wird definiert, wenn ein Individuum **A** durch eine ObjectProperty einem anderes Individuums **B** verbunden ist, bedeutet das, dass die Verbindung des Individuums **B** durch dieselbe ObjectProperty mit **A** nicht möglich ist.
  Z. B. `STÜTZE hatMaterial STAHL_S355`.
- `owl:TransitiveProperty` Diese Eigenschaft gewährleistet die Transitivität, Beispiel siehe Liste 11.

Liste 11: Beispiel `owl:TransitiveProperty`

```
#behauptete Tripel
:UNTERBAU :enthält :FELD_3 .
:FELD_3 :enthält :BRÜCKENDECK_SEGMENT_7 .
#geschlussfolgertes Tripel
:UNTERBAU :enthält :BRÜCKENDECK_SEGMENT_7 .
```

2.5.8 Property Chain Axiom

Property Chain Axiom kann mit Rolle-Kette-Axiom übersetzt werden. Es gibt zwei Arten von Property Chain Axiom: *SubObjectPropertyOf* und *ObjectPropertyChain*.

Das SubObjectPropertyOf Axiom erlaubt, dass eine ObjectProperty einer anderen ObjectProperty untergeordnet werden kann [56].

Liste 12: Beispiel SubObjectPropertyOf

```
:BRUECKENTEIL_WEST :hatDehnungsfuge :FUGE_ACHSE_15
"impliziert ->"
:BRUECKENTEIL_WEST :hatFuge :FUGE_ACHSE_15
```

Die erste Aussage impliziert die folgende Aussage, da der Besitz einer Dehnungsfuge den Besitz einer Fuge voraussetzt.

Die komplexere Form von *SubObjectPropertyOf* Axiom ist *ObjectPropertyChain*. Dies sagt aus, wenn ein Individuum X sich durch eine Reihe von ObjectProperties (PropertyA, PropertyB ... PropertyX) auf ein Individuum Y bezieht, bezieht sich ebenfalls X auf Y durch das *ObjectPropertyChain* Axiom [56]. Anders gesagt - die Anwendung einer Reihe der ObjectProperties impliziert eine andere ObjectProperty, siehe Liste 13.

Liste 13: Beispiel ObjectPropertyChain

```
:hatStahlgüte "und" :istAußenbauteil "impliziert" :hatInstandhaltungskosten
```

Es ist anzumerken, dass die willkürliche Anwendung des *Property Chain Axiom* leicht zur Unentscheidbarkeit führen kann. Damit die Entscheidbarkeit erhalten bleibt, müssen einige

Einschränkungen eingehalten werden [55]. Um im Rahmen der OWL DL zu bleiben, sollen laut [59] einige Einschränkungen bei den Aussagen eingehalten werden, z.B. Einhaltung der Regularität von *ObjectPropertyChain*. Hierauf soll im Folgenden nicht weiter eingegangen werden, da es den Rahmen dieser Arbeit sprengen würde. Weitere Informationen können [59] entnommen werden.

2.5.9 Open World Assumption vs. Closed World Assumption

Bei *Open World Assumption* wird immer davon ausgegangen, dass die vorhandene Wissensbasis nie vollständig ist und potentiell immer erweitert werden kann [2]. Wenn nichts über die Existenz einer Ressource gesagt werden kann, bedeutet dies nicht, dass sie nicht existiert, sondern, dass deren Existenz in dieser Wissensbasis nicht explizit konstatiert wurde.

In konventionellen Programmiersprachen, Wissens- und Datenbasen hingegen, wird mit der *Closed World Assumption* gearbeitet. Das bedeutet, wenn das Vorhandensein von einer Ressource nicht festgestellt wurde, bedeutet dies eindeutig, dass die Ressource in dieser Wissensbasis nicht existiert.

2.5.10 Unique Name Assumption (UNA)

In der OWL wird die *Unique Name Assumption (UNA)* (Einzigartige Namen Annahme) nicht verwendet. Das bedeutet, dass verschiedene Namen sich auf ein Individuum beziehen können, wenn sie nicht explizit als *DifferentIndividuals* definiert wurden. Ebenfalls können verschiedene Individuen als ein einziges Individuum betrachtet werden, wenn sie explizit als dasselbe Individuum definiert wurden [57].

3 Stand der Forschung und Technik

In diesem Kapitel wird der aktuelle Stand der Forschung und Technik von BIM in Bezug auf den Brückenbau, sowie der Kombination von BIM und Semantic Web Technologien in Bezug auf die Baubranche allgemein vorgestellt.

3.1 BIM in Deutschland

Im Jahre 2015 wurde vom Bundesministerium für Verkehr und digitale Infrastruktur (BMVI) der Stufenplan für die Einführung von BIM in der Baubranche veröffentlicht. Der Fokus des BMVI liegt momentan auf Großprojekten [35]. Die Pläne namens *Aktionsplan Großprojekte*, sowie *Leitfaden Großprojekte* wurden vom BMVI mit dem Ziel erarbeitet, die Grundlagen für die Anwendung der BIM-Methode in den öffentlichen Großprojekten zu ermöglichen [39]. Letztendlich soll ab Ende 2020 die BIM-Methode nach dem BMVI bei allen neu zu planenden Infrastrukturprojekten verbindlich eingesetzt werden [35].

Zukünftig soll BIM in der ganzen Baubranche angewandt werden. Dabei sollen zukünftig Klein- und Großprojekte, Hochbau und Infrastruktur, Neubau und Bauen im Bestand unter dem Einsatz von BIM geplant werden und über den gesamten Lebenszyklus des Gebäudes hinweg benutzt werden [8].

Mit dem Einsatz von BIM wird die Planung digital durchgeführt. Dabei wird eine Datenbasis geschaffen, die ganze Abläufe und Informationen verbindet, dass alle Projektbeteiligten darauf zugreifen können. Dadurch werden alle projektbezogenen Informationen transparent miteinander vernetzt, sodass Auswirkungen einer Änderung bei allen Beteiligten durch ständige Aktualisierung schnell sichtbar sind. Dadurch werden die Zeitpläne, Kosten und Risiken präzise und schneller ermittelt, sodass die notwendige Optimierung früher durchgeführt werden kann [35].

Um die Einführung von BIM in dem Infrastrukturbereich zu forcieren, hat das BMVI im Oktober 2016 die Arbeitsgemeinschaft *ARGE BIM4INFRA2020* beauftragt, wichtige Voraussetzungen für die Umsetzung des BIM-Stufenplans zu schaffen. Anhand von initiierten Pilotprojekten sollten Erfahrungen beim Einsatz der BIM-Methode im Infrastrukturbau gesammelt werden [41].

Der Stufenplan sah eine Vorbereitungsphase und Pilotphase mit der weiteren Einführung von BIM vor. Ursprünglich sollten vier Pilotprojekte durchgeführt werden:

- Brücke über den Petersdorfer See, A 19 Höhe Anschlussstelle Waren
- Talbrücke Auenbach, B 107 Südverbund Chemnitz
- Eisenbahnüberführung Filstal, Neubaustrecke Wendlingen-Ulm
- Eisenbahntunnel Rastatt, Neubaustrecke Karlsruhe-Basel

Später wurde die Erstauswahl um zwei weitere Projekte erweitert:

- Streckenplanung B 87n, Abschnitt Eilenburg – Mockrehna
- Ingenieurbauwerke entlang Los 5 der B31 Ost, Immenstaad-Waggershausen *(wurde nicht in die Auswertung mit einbezogen)*

In allen Pilotvorhaben wurden der Fokus auf unterschiedliche Schwerpunkte gesetzt. BIM-Abwicklungspläne wurden in unterschiedlicher Form und Detaillierung mit den projektspezifischen Zielen, BIM-Anwendungsfällen, Rollen, Zuständigkeiten und Datenübergabepunkten erstellt [37].

Das IFC-Format wurde dabei mangels Unterstützung für Infrastrukturbauvorhaben nicht eingesetzt [36]. Die Daten wurden ggf. im IFC-Format gespeichert, jedoch wurden sie nicht für weitere Anwendungen wiederverwendet. Das offizielle Release vom IFC-Format hatte zu dem Zeitpunkt das Schema, das nur für Gebäuden ausgelegt wurde, so dass die Anwendung vom IFC im Infrastrukturbereich die Arbeitsweise noch erschwert hätte. Daher konnten nicht alle Vorteile durch den Einsatz der BIM-Methode genutzt werden. Trotzdem haben die oben genannte Pilotprojekte vielversprechende Ergebnisse bezüglich der Effizienz der BIM-Methode gezeigt und haben als Grundlage für die Erstellung erster Handlungsempfehlungen und Leitfäden gedient [36].

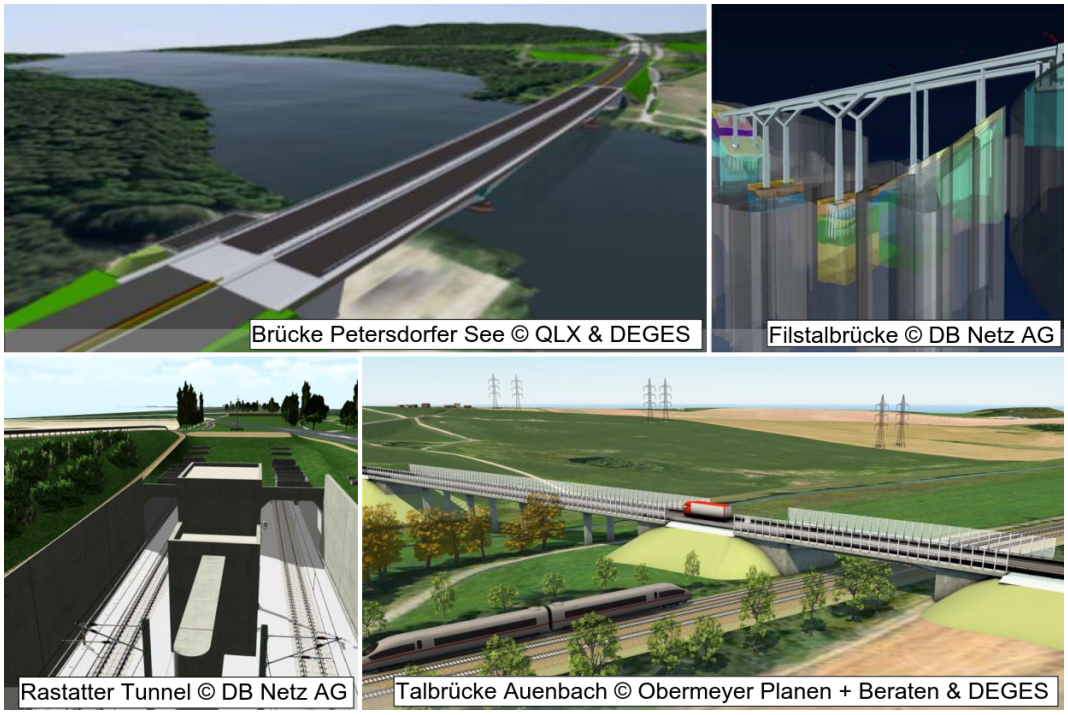


Abbildung 6: Visualisierung der Pilotprojekte: Brücke Petersdorfer See, Brücke Filstal, Tunnel Rastatt, Brücke Auenbachtal [36]

3.2 IFC & Infrastruktur

3.2.1 Infra Room, buildingSMART International

Infra Room (Infrastructure Room) wurde bei buildingSMART International (bSI) für die internationale Entwicklung des Infrastrukturteils im IFC-Schema gebildet, siehe den Entwicklungsplan auf der Abbildung 7. Dabei wurden die Entwicklungen von Projekten *IFC-Bridge*, *IFC-Road*, *IFC-Rail*, *IFC-Ports/Waterways* und *IFC-Tunnel* vorgesehen [46].

Da die Trassierung im modernen Straßen-, Brücken- und Tunnelbau sehr wichtig ist, wurde als erstes mit den Projekten *IFC-Infra Overall Architecture* und *IFC-Alignment* begonnen.

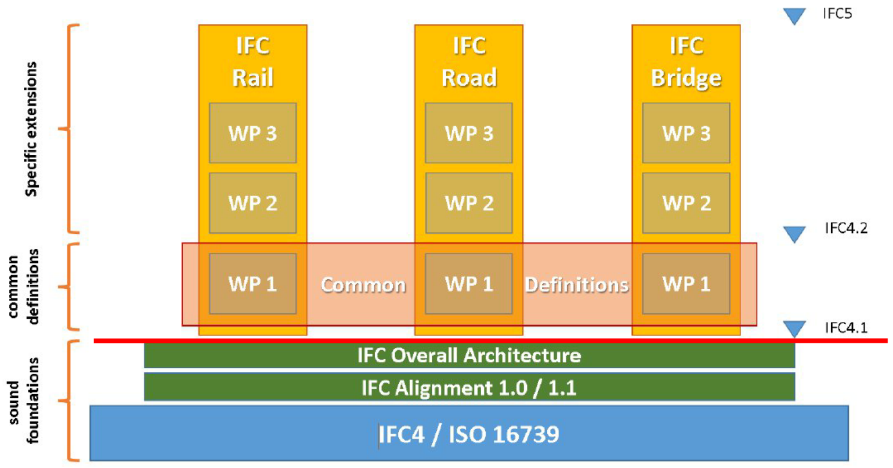


Abbildung 7: IFC-Entwicklungsplan [44]

Sie bilden eine Basis des strukturierten Vorgehen für die Erweiterung der IFC im Bereich Infrastruktur im Zusammenhang aller Teile. IFC-Alignment ist eine Erweiterung für den Datenaustausch zur digitalen Beschreibung von Trassierungsbauwerken [46].

Damit die Trassenkrümmung im Grund-, sowie Aufriss exakt berücksichtigt werden kann, ist der Einsatz von Geodaten erforderlich. Daher ist für die zukünftige Interoperabilität beim Geodatenaustausch in der Trassierungsentwicklung eine enge Kooperation mit dem *Open Geospatial Consortium* (OGC) vorgesehen [46].

Die Entwicklung der IFC-Alignment ermöglicht eine Trasse bezogen parametrische Modellierung. Im Brückenbaubereich ist dieses Thema momentan sehr aktuell.

Das Projekt IFC-Alignment hat in der Version IFC4.1 bereits den Status eines offiziellen bSI-Standards erreicht [42].

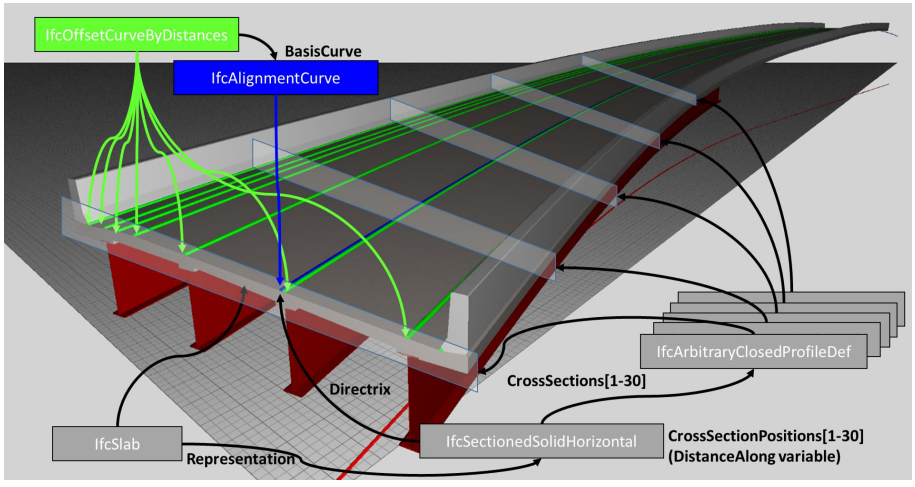


Abbildung 8: Auszug aus Dokument des Expertentreffens *IFC-Bridge*. Teil - *IFC-Alignment* [51]

Laut [52] wurde im Januar 2017 die Entwicklung der *IFC-Bridge* Erweiterung gestartet. Die Projektdauer wurde laut [48] für 2 Jahre ausgelegt mit dem Ziel, eine exakte geometrische und semantische Beschreibung von Brücken zu ermöglichen [48]. Ein bereits entwickeltes Schema *IFC-BRIDGE V2 Revision 2 (R2)* des Vorreiterteams aus Frankreich&Japan

hat als Hauptgrundlage für die *IFC-Bridge* Entwicklung gedient, siehe Kapitel 3.2.3.

Laut [45] wurden für das Projekt insgesamt 180.000,-€ gesammelt. Die Absichtserklärung wurde von entsprechenden Organisationen aus Frankreich, Deutschland, Schweden, Finnland, Japan, sowie buildingSMART International unterschrieben [45].

Momentan befindet man sich offiziell in der Ebene IFC4.1, siehe Abbildung 7, wobei Ende 2018 ein Vorabzug der *IFC-Bridge* IFC4.2 veröffentlicht wurde [47] [53].

3.2.2 IFC & Infrastruktur in Deutschland

Nach dem Start von Projekten in *Infra Room* wurde mittlerweile vom BMVI die Mitentwicklung des internationalen IFC-Standards im Infrastrukturbereich in Deutschland gefördert [42]. Das Projekt namens *IFC-INFRA* sollte eine Mitwirkung bei jedem Teilprojekt der IFC-Entwicklung der Infrastruktur gewährleisten. Dafür sollte die Ausbildung der deutschen Expertengruppen für jedes Teilprojekt vorgesehen werden.

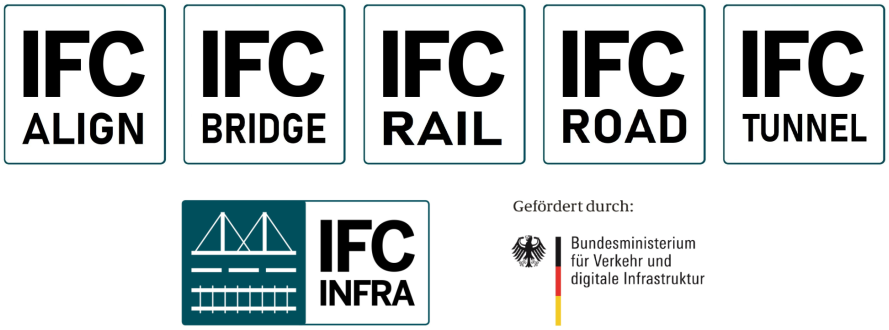


Abbildung 9: Logos der laufenden Projekten von *IFC-INFRA* [42]

Im April 2017 wurde laut [44] die deutsche Expertengruppe für *IFC-BRIDGE* gebildet.

Dafür wurden folgende Projektpartner beauftragt:

- Ruhr-Universität Bochum – Lehrstuhl für Informatik im Bauwesen
- planen-bauen 4.0 GmbH
- Technische Universität München – Lehrstuhl für Computergestützte Modellierung und Simulation
- AEC3 Deutschland GmbH

Die Expertengruppe sollte folgende Aufgaben erfüllen:

- Mitentwicklung eines internationalen Standards im Infrastrukturbereich im Rahmen eines offiziellen bSI-Projekts
- Vertretung auf internationaler Ebene
- Entsendung von Mitgliedern der ARGE zu Standardisierungstreffen
- Beitrag zur Erweiterung der IFC und Spiegelung der internationalen Vorschläge mit deutschen Interessen

Als weitere Ziel des BMVI ist die Übertragung des Standards von buildingSMART über

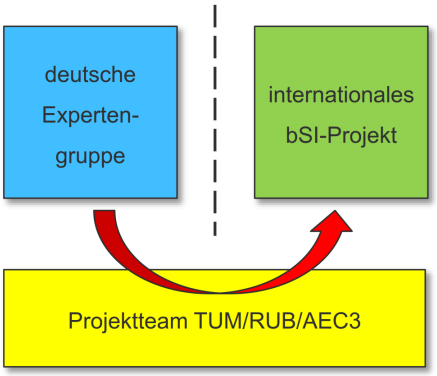


Abbildung 10: Begleitung der internationalen Entwicklung [44]

ISO16739:ed2 und CEN16739:ed2 zu DIN16739:ed2 [44]. Laut [49] sollte Ende Februar 2018 der Standardisierungsprozess gestartet werden.

3.2.3 IFC-BRIDGE V2

Im Jahre 2002 hat ein französisches Team eine Erweiterungsschema des IFC-Formates für Brückenbauwerke *IFC-BRIDGE* entwickelt. Das Schema hat auf IFC von International Alliance for Interoperability (seit 2008 zu buildingSMART umbenannt), sowie auf OA-EXPRESS von SETRA (Französisches Amt für Verkehr-, Strassenuntersuchung und deren Gestaltung) basiert [11].

Mittlerweile hat das japanische Team am *Muroran Institute of Technology* in Kooperation mit *Japan Prestressed Concrete Contractors Association* ein Erweiterungsmodell für das IFC-Schema für vorgespannte Brücken entwickelt [11].

Am Anfang wussten beide Forschungsgruppen nicht über die gegenseitigen Entwicklungen Bescheid, obwohl die Herangehensweisen sehr ähnlich waren. Die Entwicklung lief parallel und unabhängig voneinander, bis beide Teams den jeweiligen Leistungen erfahren haben. Die Forschungsgruppen haben sich danach dazu entschlossen zu kooperieren, die Leistungen zusammenzuführen und ein gemeinsames Modell zu erstellen. Das Ziel dabei war, die Grundlagen der Erweiterung für das IFC-Schema zu schaffen, die später zu einem Weltstandard wurden [11].

Nach der Zusammenarbeit haben im Jahre 2005 die Forschungsteams als Ergebnis die entstandene gemeinsame Endversion *IFC-BRIDGE V2 Revision 2 (R2)* vorgestellt.

3.3 BIM im Kontext von Semantic Web

Die Nutzung von Semantic Web Technologien ermöglicht die Darstellung der menschenlesbaren Daten auch in der maschinenlesbaren Form. Dies ermöglicht einen weiteren Entwicklungsschritt in der Automatisierung, indem mehr Prozesse bezüglich der Informationsverarbeitung von Computern übernommen werden können.

In der *Linked Data* Technologie wurde ein vielversprechendes Potenzial erkannt, indem heterogene Daten aus verschiedenen Web-Quellen unter einander verknüpft werden konn-

ten [17]. Seitdem haben *Linked Data* Technologien in vielen anderen Bereichen, inklusive Bauwesen, Anwendung gefunden. Im Kontext wurde so ein Verfahren als ein interessanter Ansatz identifiziert, mit dem eine bessere Interoperabilität und ein flexiblerer Datenaustausch erreicht werden kann [18]. Infolgedessen wird in der Baubranche eine zunehmende Anzahl von Software für Informationsmanagement und Informationsaustausch in Anlehnung an *Linked Data* Technologien eingesetzt [14].

Die Verbindung zwischen Semantic Web Technologien und dem IFC-Format kann durch die Verwendung von Ontologien repräsentiert werden. Laut [15] wurden in den letzten Jahren ein paar Ontologien im Baubereich und FM (Facility Management) entwickelt. Die bedeutende Implementierung dabei war die *ifcOWL* Ontologie von Pauwels P. und Terkaj W. [14]. Im Rahmen dieser Arbeit wurde das EXPRESS-Schema in eine Ontologie transformiert. Die *ifcOWL* wurde nach dem etablierten Standard der OWL-Sprache (Web Ontology Language) erstellt. Mit der Implementierung der *ifcOWL* von Pauwels P. und Terkaj W. konnten einige Aussagen getroffen werden. Als erstes, ist die Anwendung der OWL–Ontologien mit dem Einsatz des etablierten IFC-Formates möglich. Des Weiteren weist die Verwendung von Ontologien viele Möglichkeiten in Hinsicht auf Datenverteilung, Datenspeicherung, Erweiterbarkeit der Datenmodellen, Abfragen und Schlussfolgern von Informationen, Konsistenzprüfung von Modellen, sowie Wissensherleitung auf [14].

Abschließend wurden von Pauwels P. und Terkaj W. die Empfehlungen für den Einsatz von Ontologien mit dem IFC-Format formuliert. Erstens, soll die Ontologie mit der Sprache OWL2 DL umgesetzt werden, da die DL-Teilsprache entscheidbar ist, siehe auch Kapitel 2.3.2. Weiterhin soll die Ontologie mit dem originalen EXPRESS-Schema so übereinstimmend sein, wie möglich. Letztendlich soll die Ontologie primär, und nicht sekundär, genutzt werden, um die Umwandlung von IFC-Instanzen-Datei in gleichwertige RDF-Graphen zu ermöglichen.

Von Mendes de Farias T., Roxin A. und Nicolle C. [15] wurde ein Verbesserungsvorschlag zur *ifcOWL* Ontologie mit einer semantischen Adaptation gebracht, was eine bedeutende Verbesserung bei der Durchführung von SPARQL-Abfragen von 90% bis zu 95% aufwies. Dabei haben Pauwels P., Roxin A. [16] diesen Ansatz als zu komplex und kaum für die Anwendung in der Praxis beschrieben und haben den Vorschlag für die Reduktion der Anzahl von RDF-Graphen gemacht, indem alle geometrische Daten, sowie zusätzliche Zwischenbeziehungen zwischen Objekten ausgelassen werden.

Darauf folgend wurde von Rasmussen M. H., Pauwels P., Lefrançoi M. et al. [13] die *BOT* (The Building Topology Ontology) Ontologie veröffentlicht. *BOT* ist eine einfache und erweiterbare Basis-Ontologie für die Verwendung mit anderen auf konkrete Domäne bezogene, Ontologien. Dies folgt allgemeine Prinzipien von W3C, bei welchen die Informationen wiederverwendet werden und das Schema nicht komplexer gehalten wird als nötig [19]. Mit BOT wurden Grundkonzepte eines typischen Gebäudes abdeckt, und zwar: Element, Gelände, Zone, Räume, Schnittstellen zwischen benachbarten Zonen bzw. Elementen, sowie Relationen zwischen diesen Konzepten, siehe Abbildung 11.

Eine der ersten Arbeiten mit der semantischen Anreicherung von Brückenmodellen erfolgte innerhalb des *SeeBridge* Projekts [28]. SeeBridge ist die Abkürzung von *Semantic*

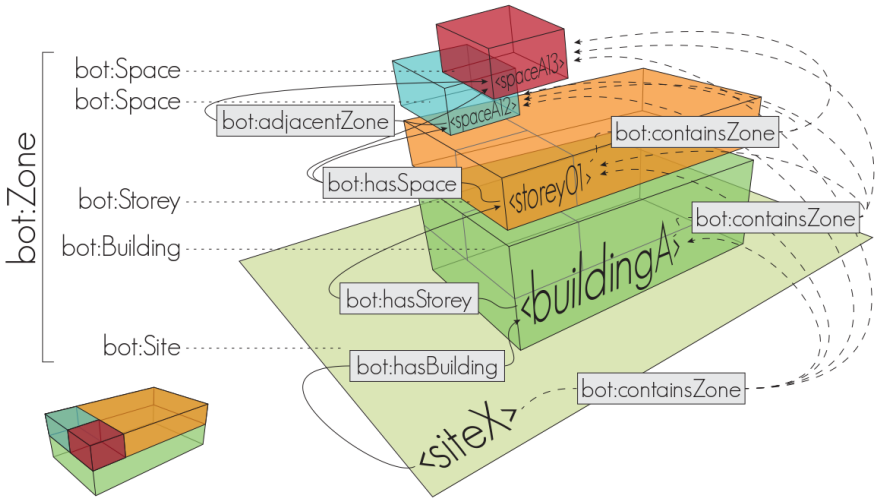


Abbildung 11: Zonenverbindung in *BOT* [13]

Enrichment Engine for Bridges. Innerhalb von SeeBridge wurde eine Methode zur automatischen Verarbeitung von BIM-Modellen von Massivbrücken im Bestand vorgeschlagen. Die BIM-Modelle enthalten exakte Informationen über die Geometrien, sowie semantische Informationen über ihre Komponenten und Schäden.

Im Paper von Ren G., Ding R., Li H. [20] ist die semantische Verarbeitung im Bereich von Bestandsbrücken erfolgt. Der Ansatz von Regeln hat die automatische Klassifizierung von Schäden ermöglicht.

Fazit Es lässt sich zusammenfassen, dass zu diesem Zeitpunkt keine Projekte gefunden wurden, die sich auf die ausführliche semantische Beschreibung von Brückenbauten mit der folgender Verarbeitung der Semantik fokussiert hatten. Dies ist die Motivation für die vorliegenden Arbeit.

4 Erarbeitung der BRIDGE–Ontologie

Als BRIDGE–Ontologie wird die im Rahmen dieser Arbeit erarbeitete Ontologie bezeichnet. Sie fasst die relevanten semantischen Bauelemente der Brücken, die in Spannverbund-, sowie Verbundbauweise konzipiert werden, zusammen. In diesem Kapitel wird der Erarbeitungsprozess im Detail erläutert.

Die Erarbeitung einer Ontologie kann grundsätzlich in zwei Phasen unterteilt werden. Die erste Phase ist die Analyse der Voraussetzungen, der Ontologie. Die zweite Phase ist der Erstellungszyklus unter Berücksichtigung der festgelegten Voraussetzungen [32]. Der Erstellungsprozess wird nicht als abgeschlossen betrachtet, da das Wissen (TBox, sowie ABox) in der Ontologie normalerweise ständig angepasst wird.

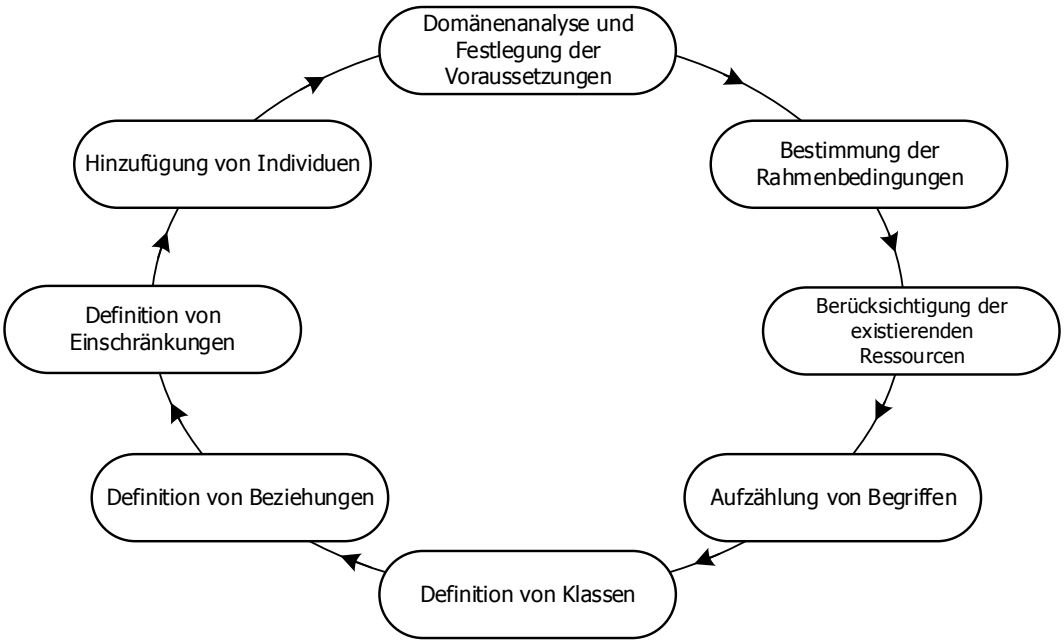


Abbildung 12: Erarbeitungsyzklus einer Ontologie laut [32]

4.1 Grundlagen für die Erarbeitung

4.1.1 Analyse der Voraussetzungen

Vor der Erarbeitung einer Ontologie müssen die Voraussetzungen und erwarteten Ergebnisse analysiert werden. Danach sollen Rahmenbedingungen festgelegt werden.

Eine herkömmliche Praxis von *Ontology Engineering* ist eine anfängliche Definition der möglichen Abfragen an das Ontologiemodell. Die Ontologie soll diese Abfragen beantworten können, wobei die Möglichkeiten nicht weiter über die Grenzen dieser Beantwortung hinausgehen sollen [32]. Der Hauptprinzip lautet: nicht komplexer als nötig und so einfach wie möglich.

Zunächst soll der Zielbereich und seine Grenze definiert werden. In welchen Ausmaßen kann die Ontologie erweitert werden? Welche Anforderungen an die Berechnungszeit be-

stehen und welche Berechnungsressourcen stehen zur Verfügung? Damit kann die nötige Komplexitätsklasse bestimmt werden, nach der die Aussagekraft der Sprache ausgewählt werden kann.

4.1.2 Erarbeitungazyklus einer Ontologie

Bei jeder Ontologie ist immer eine Taxonomie vorgesehen. Der Begriff *Taxonomie* lässt sich durch das Wort *Ordnungsgesetz* erklären. Taxonomie ordnet und verbindet Klassen, zwischen denen die Beziehung *Oberklasse — Unterklasse* definiert ist [1]. Dabei vererbt die Unterklasse alle Eigenschaften der Oberklasse. Von daher, soll die Unterteilung Oberklasse — Unterklasse widerspruchsfrei sein, um Konflikte bei der Klassenverwendung zu vermeiden [5]. Abschließend, soll die Taxonomie logisch strukturiert und leicht nachvollziehbar sein.

Weiterhin stellen die Klassen eine Konzeption dar, was den wesentlichste Charakteristika dieser Klasse dient. Dabei soll das Wort an sich (Klassenname) bloß als eine sprachliche Einheit gelten, die im Klassennamen durch ein anderes Synonym ersetzt werden kann (und immer noch derselben Konzeption zu Grunde liegt). Es lässt sich schlussfolgern: Es muss die Konzeption der Klasse möglichst genau abgebildet werden, ohne den festen Bezug zu dem Wort (Klassenname) zu haben.

Auf der Abbildung 12 ist der Erarbeitungszyklus für eine Ontologie dargestellt. Der Punkt *Berücksichtigung der existierenden Ressourcen* sieht die Verwendung schon existierender ontologischer Bibliotheken vor. Als Beispiel für die BRIDGE–Ontologie, ist die Einbindung einer externen Ontologie für Materialien sehr sinnvoll.

4.1.3 TBox vs. ABox

Die Aussagen werden in einer Ontologie in zwei Gruppen eingeteilt, und zwar in die sogenannte *TBox* und *ABox*, siehe Abbildung 13. Die TBox enthält das terminologische Schemawissen in der formalen Darstellung, das als Vokabular für die Erstellung der ABox dient. ABox bezeichnet konkretes Wissen über Instanzen [2].

TBox beschreibt abstrakt die diskutierte Domäne in einer formalen Form und prädestiniert für die Erstellung der ABox, die das konkrete Wissen über konkrete Instanzen in der Welt enthält. Die Liste 14 stellt den Unterschied zwischen diesen Begriffen dar.

Liste 14: Beispiel der TBox und ABox

```
# TBox
:Pile # Bohrpfahl als eine abstrakte Klasse
:Footing # Gründung als eine abstrakte Klasse
:bears # ObjectProperty
# ABox (Instanzen)
:PILE_7_AXIS_14 # eine konkerete Bohrpfahl in einer konkreten Brücke
:FOOTING_AXIS_14 # ein konkretes Gründungsbauteil in einer konkreten Brücke
# ABox (behauptetes Tripel)
:PILE_7_AXIS_14 :bears :FOOTING_AXIS_14 .
```

```
# ABox (geschlussfolgertes Tripel)
:FOOTING_AXIS_14 :restOn :PILE_7_AXIS_14 .
```

Abbildung 13: TBox und ABox

4.1.4 Rahmenbedingungen

Die BRIDGE–Ontologie dient zur Beschreibung der semantisch relevanten Bauelemente von Brücken. Im Rahmen dieser Arbeit wird die Taxonomie der Bauelemente sich auf die Brücken mit der Spannbeton- bzw. Verbundbauweise beziehen. Der Grund dafür ist, dass heutzutage die Brückenbauten größtenteils in diesen Bauweisen konzipiert werden.

Als Modellierungssprache wird OWL DL verwendet.

4.1.5 Wahl der Modellierungssprache

Laut [1] weist die RDFS-Sprache folgende Beschränkungen auf:

- Es fehlen die lokalen Restriktionen für Rollen (ObjectProperty / DatatypeProperty). Z. B. die ObjectProperty `hatGüte` soll bei der Anwendung an Stahlbetonbauteilen den `rdf:range` – `Betongüte` haben, sowie bei der Anwendung an Stahlbauteilen – `Stahlgüte`.
- Es fehlen Zahlenrestriktionen sowie die Restriktion auf Existenz. Z. B. jedes simples Bauteil verfügt über exakt ein Material. Ein Knoten kann nicht mehr als über sechs Lagerreaktionen verfügen.
- Es fehlen transitive, symmetrische, asymmetrische, reflexive und irreflexive Charakteristiken für ObjectProperties, sowie die Möglichkeit disjunkte Ressourcen definieren zu können.

Die symmetrischen und transitiven ObjectProperties sind essenzielle Relationen im Bauwesen, da sie den Kontakt von Bauteilen bzw. Durchleitung von Lasten darstellen. Die Restriktionen stellen aber auch eine wichtiges Konstrukt. Der Verzicht darauf wäre im Rahmen der BRIDGE–Ontologie nicht erwünscht, da das Ziel auch ist die Möglichkeitsgrenzen zu untersuchen. Darüber hinaus determinieren diese Beschränkungen die erwünschte Funktionsliste.

Als Modellierungssprache für Ontologie wurde schließlich OWL DL verwendet um mit der hohen Aussagekraft über die Entscheidbarkeit zu verfügen.

4.1.6 Namenskonvention

Innerhalb der BRIDGE–Ontologie werden die URI-Namen von Ressourcen auf Englisch geschrieben, damit die erarbeitete Ontologie auch in der internationalen Gemeinschaft genutzt werden kann. Dabei wird die sogenannte *CamelBack* Notation verwendet. CamelBack ist eine Schreibweise für Phrasen, mit der jedes folgende Wort großgeschrieben wird, wobei die Leerzeichen ausgelassen werden, z. B. `SoWirdEsGeschrieben`. Weiterhin wird auch zwischen *UpperCamelCase* und *lowerCamelCase* unterschieden. Dabei werden die Anfangsbuchstaben entweder groß- (bei UpperCamelCase) oder kleingeschrieben (bei lowerCamelCase).

In der BRIDGE–Ontologie wird für die Klassennamen die UpperCamelCase- und die lowerCamelCase-Notation für die Namen von ObjectProperties und DatatypeProperties genutzt. Die Individuen, sowie die Werte von Data typeProperties werden großgeschrieben, wobei die Wörter mit Unterstrichen getrennt werden, siehe Legende in der Tabelle 3.

Klasse:	`GirderDeck`	UpperCamelCase
ObjectProperty:	`isReinforcedWith`	lowerCamelCase
DatatypeProperty:	`elem_reinforcementTask`	lowerCamelCase
Wert der DatatypeProperty:	`STATICALLY_REQUIRED`	Großgeschrieben
Individuum:	`GIRDER_DECK_2`	Großgeschrieben

Tabelle 3: Legende. Schreibweise für Ressourcen in der BRIDGE–Ontologie

4.1.7 Protégé Software

Die Erarbeitung der Ontologie erfolgte mit Hilfe von *Protégé Software 5.2.0*. Eineerseits weil die Software eine Opensource und Freeware ist, die mit allen nötigen und gängigen Formaten arbeiten kann. Andererseits weil sie viele der benötigten Plugins, wie für SWRL und SPARQL, hat und einige Reasoners unterstützt. Die Schlussfolgerung wurde mit dem Reasoner *Pellet* durchgeführt.

4.2 Erstellung der TBox

Unterteilung der Ontologie Eines der Prinzipien von Linked Data Technologie, ist die Wiederverwendung der Informationen. Die Datensätze sollten je nach selektiv verwendet werden können, um die Komplexität so gering wie möglich halten zu können. Bei dem Einsatz, sollen mögliche Anwendungsfälle berücksichtigt werden.

Eine Ontologie kann nicht alle potenzielle Anwendungen berücksichtigen, da die Kombination verwendeter Klassen von der Brückenart und dem Vorhaben abhängig ist. Daher

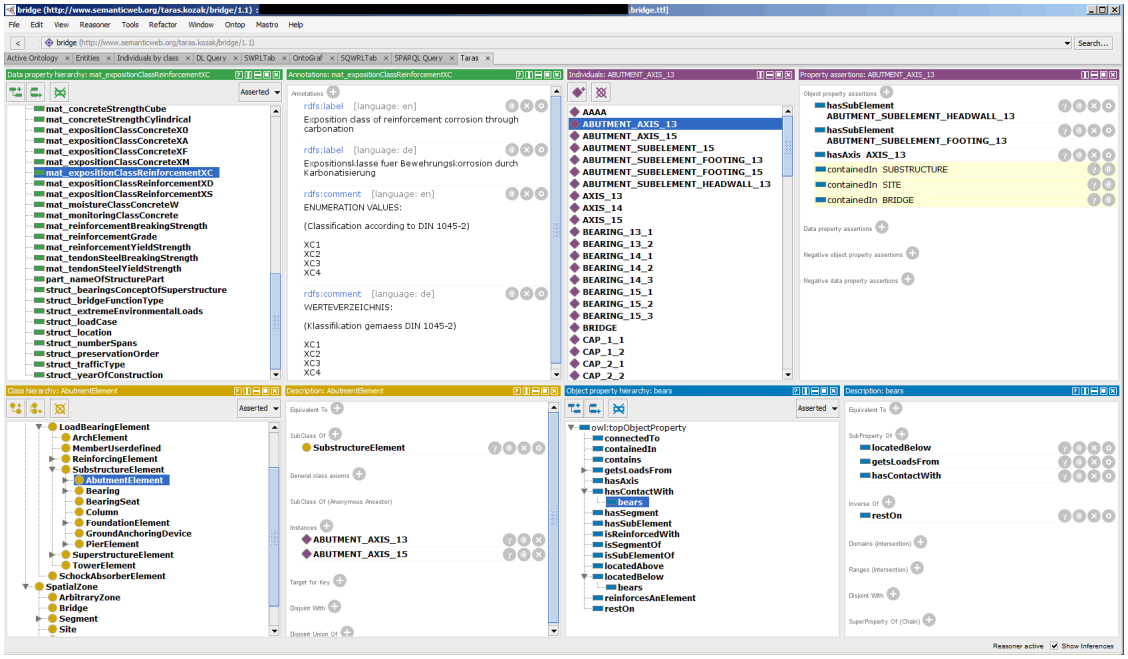


Abbildung 14: Interface von Protégé Software

muss eine qualitative Einteilung von Fall zu Fall neu getroffen werden und wird im Rahmen dieser Arbeit nicht in Teilontologien unterteilt und stattdessen als einzelne Datei betrachtet.

Erarbeitungsschritte Die Erstellung der terminologischen Komponente (TBox) sieht folgende Schritte vor:

- Definition der Begriffe in einer Domäne.
- Organisation dieser Konzepte in die Taxonomie
- Erstellung von ObjectProperties und Definition ihrer Eigenschaften
- Berücksichtigung von DatatypeProperties
- Definition von Einschränkungen

4.3 TBox: Taxonomie

Zu der Ausarbeitung der Taxonomie für die BRIDGE–Ontologie wurde ein Vorabzug von IFC-Bridge 4.2 veröffentlicht, siehe Kapitel 3.2.1. Aus jener Taxonomie wurden die Klassen `RebarArray`, `SchockAbsorberElement` und `Deviator` in die BRIDGE–Ontologie eingearbeitet.

Die vollständige Liste von Klassen ist in dem Anhang A.1 in der Tabelle 6 aufgeführt.

Grundklassen der BRIDGE–Ontologie Im Bauwesen stellt ein Bauwerksmodell, sowohl Anspruch an die räumliche, als auch an die elementbezogene semantische Klassifizierung. Die Elemente werden nach ihrer Bauteilart klassifiziert, sowie nach der Zugehörigkeit

zu verschiedenen räumlichen Zonen. Der Grund dafür ist, dass das Bauwerk in verschiedenen Fachbereichen und Bauphasen auf verschiedene Art und Weisen behandelt wird. Dadurch entsteht der Bedarf einer flexiblen Unterteilung.

In der *Bridge Topologie Ontologie (BOT)* [13] wurden Kernkonzepte (`Zone` und `Construction element`) in Anlehnung an die ISO Norm 12006-2 (*Building construction - Organization of information about construction works. Part 2: Framework for classification*) konzipiert.

Unter dem Begriff `Zone` nach ISO 12006-2 wird ein begrenzter dreidimensionaler Bereich mit einer bestimmten Funktion verstanden, der physikalisch oder abstrakt definiert werden kann. Die `Zone` kann andere Zonen, sowie Elemente enthalten. Unter `Construction element` wird eine physikalische Einheit im Bauwesen mit bestimmten Eigenschaften, Form, räumlicher Struktur und Position verstanden. Das `Construction element` muss mindestens eine Funktion in der Struktur aufweisen [65].

Die oben genannten Begriffe werden in die BRIDGE–Ontologie folgend integriert, siehe Tabelle 4.

Die BOT-Ontologie könnte als eine Hauptontologie verwendet werden, mit der Erarbeitung der Teilontologien für die Erweiterung. Aber da die räumlichen Konzepte eines Gebäudes und einer Brücken nicht übereinstimmen, ist dies nicht möglich.

ISO 12006-2	BRIDGE–Ontologie
`Zone`	`SpatialZone`
`Construction element`	`Element`

Tabelle 4: Begriffsübereinstimmung zwischen ISO 12006-2 und BRIDGE–Ontologie

Entität SpatialZone Ein konventionelles Brückenbauwerk besteht grundsätzlich aus zwei Teilen. Unterbau und Überbau. Die Aufgabe des Unterbaus ist es die Lasten des Überbaus in die Gründung abzutragen. Zum Unterbau gehören Widerlager, Gründung, Pfeiler und Teile des Bogens und Pylons unterhalb des Brückendecks.

Die Aufgabe des Überbaus ist die Gewährleistung der Fahrbahnausbildung. Die Fahrbahn wird von einem Fahrbahnträger (Brückendeck) getragen, das entweder selbsttragend ist, oder in Kombination zusätzlicher Konstruktionen funktioniert. Z. B. bei einer Schrägseilbrücke wird das Brückendeck durch das System von Seilen gehalten, die an einem Pylon befestigt werden. Zum Überbau gehören: Brückendeck, Seiltragwerk und Teile des Bogens und Pylons oberhalb des Brückendecks.

Die hierarchische Verbindung zwischen Zonen kann der Abbildung 16 entnommen werden.

Entität Element Im folgenden werden einige zugehörige Entitäten der Klasse `Element` erläutert. Des Weiteren sind die detailliertere Beschreibungen aller Entitäten mit der Übersetzung auf die deutsche Sprache im Anhang A in der Tabelle A.1 aufgeführt.

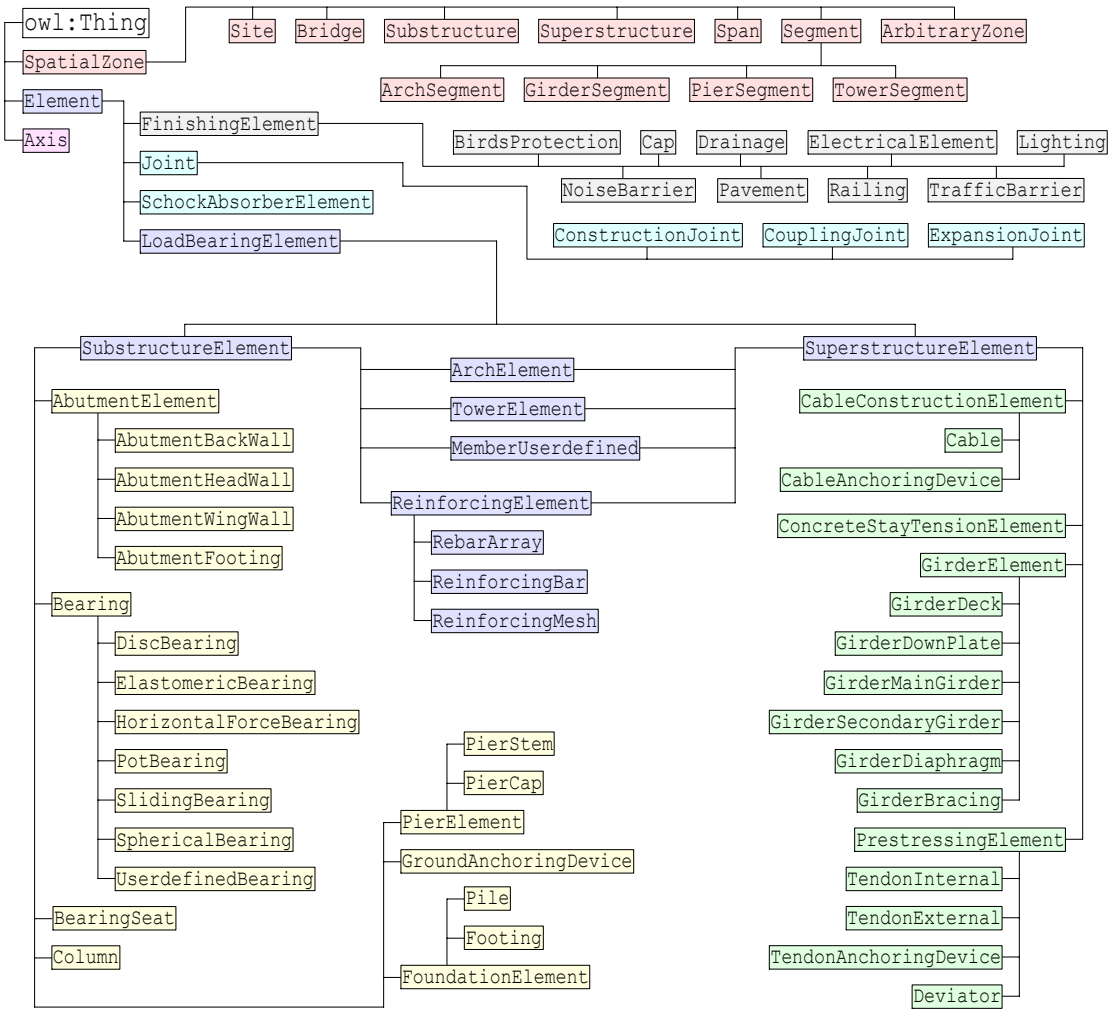


Abbildung 15: Taxonomie

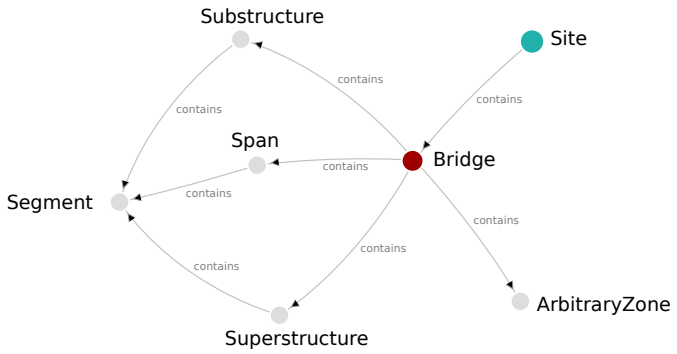


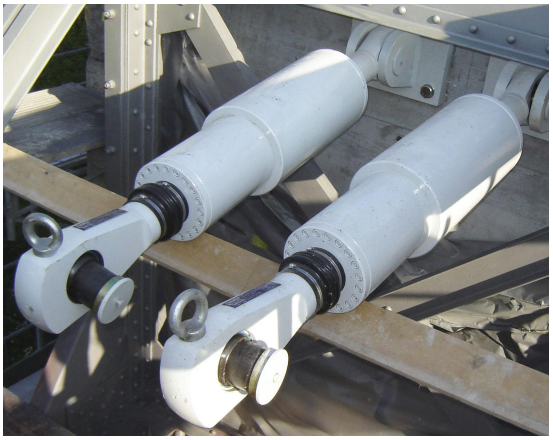
Abbildung 16: Beziehungen zwischen der Entitäten in der Klasse `SpatialZone`

Entität Joint Eine Dehnungsfuge verfügt normalerweise über eine physische Konstruktion, sodass die Klasse `ExpansionJoint` allgemein zur Klasse `Element` gehören muss. Allerdings verfügt sie über keine statische Funktion im Tragkonzept der Brücke, weshalb sie in die Klasse `LoadBearingElement` eingeordnet werden kann. Die Arbeitsfugen (`ConstructionJoint`) hingegen verfügen über keine physische Konstruktion. Die Koppelfugen (`CouplingJoint`) können zusätzliche Verankerungselemente enthalten, werden aber nicht als ein separates statisches Element betrachtet. Trotz dieser Unterschiede werden diese

Klassen zusammengeführt.

Das Beispiel einer Kragfingerfuge (eine Art der Dehnungsfuge) ist in der Abbildung 17(a) dargestellt.

(a) Kragfingerfuge



(b) Schwingungsdämpfer

Abbildung 17: Beispiel: Dehnungsfuge, Schwingungsdämpfer
Bildquelle (a): [67], Bildquelle (b): [68]

Entität SchockAbsorberElement Die Klasse Schwingungsdämpfer (`SchockAbsorberElement`) wurde aus dem Vorabzug der Taxonomie von IFC-Bridge 4.2 übernommen. Ein Schwingungsdämpfer gewährleistet die Aufnahme und Dämpfung der mechanischen Schwingungen in der Brücke.

Entität MemberUserdefined In den Bogenkonstruktionen der Brücken sind oft vertikale Zug- bzw. Druckelemente vorhanden, sogenannte Hänger und Steher. In der Abbildung 18(a) ist ein Beispiel einer Bogenbrücke dargestellt, wo sich der Bogen im Bereich des Unterbaus befindet. Für die Aussteifung sind die Stützen/Steher zusätzlich mit Querbalken verbunden. Diese Querbalken gehören auch zur Bogenkonstruktion, wobei eine Zugehörigkeit zu Hängern oder Stützen nicht gegeben ist.

Ein weiterer Punkt, sind die geneigten Streben in der Fachwerkkonstruktion einer Stahlbetonbrücke, siehe Abbildung 18(b). Ihre Einordnung in die Klasse Stütze bzw. Hänger ist auch nicht möglich.

Aus diesen Gründen werden solche Elemente neutral als eine Klasse `MemberUserdefined` behandelt. Dabei sind verschiedene Anordnungen (vertikal bzw. geneigt), sowie vorwiegende Spannungszustände (Zug bzw. Druck) innerhalb dieser Klasse möglich. Diese Eigenschaften werden von der DatatypeProperty `elem_memberType` gesteuert.

Da solche Elemente sowohl im Bereich des Überbaus, als auch in dem des Unterbaus liegen können wird die Klasse `MemberUserdefined` weder der Klasse Unterbauelement (`SubstructureElement`) noch der Klasse Überbauelement (`SuperstructureElement`) zugeordnet. Der Grund dafür ist, dass die Klassen `SubstructureElement` und `SuperstructureElement` als disjunkt behandelt werden. D.h. die Elemente die sich in einer Klasse

befinden, können in der Realität nie zu der anderen Klasse gehören. *Z. B. nach der existierenden Begriffsparadigma ist die Zugehörigkeit eines Wiederlagers zum Überbau, sowie des Brückendecks zum Unterbau ausgeschlossen.* Von daher kann die Klasse `MemberUserdefined` nicht gleichzeitig den Klassen `SubstructureElement` und `SuperstructureElement` nicht angehören, da sie sich gegenseitig ausschließen. Letztendlich, wird sie in die gleiche hierarchische Stufe wie die obengenannten Klassen eingeordnet, siehe Abbildung 15.

(a) Langwieser Viadukt, Schweiz

(b) Alfenz Brücke, Österreich

Abbildung 18: Beispiele Brücken mit `MemberUserdefined`
Bildquelle (a): [69], Bildquelle (b): [70]

Entität ArchElement Die Bogenkonstruktion einer Brücke kann sowohl im Unterbau, als auch im Überbau liegen, wobei die Anordnung in beiden Bereichen möglich ist, siehe Abbildung 19. Daher die Klasse `ArchElement` (Bogenelement) kann aus dem gleichen Grund, sowie `MemberUserdefined`, nicht den Klassen `SubstructureElement` und `SuperstructureElement` auch angehören.

(a) Guangya Brücke, China

(b) Humber Bay Arch Brücke, Kanada

Abbildung 19: Beipspiel Bogenbrücken
Bildquelle (a): [71], Bildquelle (b): [72]

Entität AbutmentElement Da das Widerlager über eine Komplexe Struktur und unterschiedliche Variationen verfügt, wurde die Klasse `AbutmentElementUserdefined` für die freie Benutzung erstellt.

Darüber hinaus kann man das der Widerlager beim modellieren als ein einziges Element ohne weitere Unterteilung behandeln. In dem Falle kann das Element der Oberklasse `AbutmentElement` zugewiesen werden.

Entitäten Column und Pier Im Rahmen der BRIDGE–Ontologie wird zwischen Stützen und Pfeiler unterschieden. Als `Column` (Stützen) werden schlanke stabförmige Bauteile bezeichnet, die über kleine Eigenlasten verfügen und nicht unbedingt lotrecht angeordnet werden können. Als `Pier` (Pfeiler) werden robuste vertikale Bauteile bezeichnet, die einen wesentlichen Anteil an Eigenlast, sowie über eine Einspannung am Fuß besitzen [64], siehe Abbildung 20.

(a) Paraná River Brücke, Argentinien

(b) Moseltalbrücke, Deutschland

Abbildung 20: Beipspiel `Column` und `PierElement`
Bildquelle (a): [73], Bildquelle (b): [74]

Entität ReinforcingElement Die Klasse `ReinforcingElement` bezeichnet die Bewehrungselemente. Die typischsten Elemente sind Bewehrungsstäbe (`ReinforcingBar`) und Bewehrungsmatten (`ReinforcingMesh`). Es werden ebenfalls Bewehrungskonstruktionen aus Stabstahl und Bügel eingesetzt, die zu einem Bewehrungskorb (`RebarArray`) zusammengeschweißt bzw. mit einem Bindedraht zusammengebunden werden [58].

Die Entität `RebarArray` wurde der Taxonomie von IFC-Bridge 4.2 übernommen.

(a)

(b)

Abbildung 21: Beispiel der Bewehrungskörbe (`RebarArray`)
Bildquelle (a): [75], Bildquelle (b): [76]

Entität ConcreteStayTensionElement `ConcreteStayTensionElement` sind massive Zugglieder, die sich im Bereich des Überbaus befinden und hauptsächlich Zugkräfte aufneh-

men. Die Zugglieder können plattenförmig, Abbildung 22(a) oder stabförmig, Abbildung 22(b), ausgebildet sein.

(a) Ganter Brücke, Schweiz

(b) Rio Magdalena Brücke, Kolumbien

Abbildung 22: Beispiel für `ConcreteStayTensionElement`
Bildquelle (a): [77], Bildquelle (b): [78]

Entität Deviator Die Klasse `Deviator` kann mit Umlenkblock übersetzt werden. Die Funktion so eines Umlenkblockes ist die Zwischenverankerung von Spanngliedern.

Abbildung 23: Umlenkblock. Klasse `Deviator`

4.4 TBox: ObjectProperties

Die vollständige Liste von ObjectProperties ist in dem Anhang A.2 in der Tabelle 7 aufgeführt.

In dem IFC-Schema werden die Beziehungen zwischen Objekten objektifiziert behandelt. D.h. Objekte werden nicht direkt miteinander, sondern durch ein dazwischengeschaltetes Objekt, welches die Beziehung selbst repräsentiert, verknüpft. Dadurch können direkt am Beziehungsobjekt beziehungsspezifische Eigenschaften gespeichert werden [7]. Bei der Ontologiemodellierung mit der OWL-Sprache werden die Beziehungen durch eigene Ressourcen `owl:ObjectProperties` repräsentiert und von Klassen-Ressourcen `owl:Class` separat behandelt. Des Weiteren wurden von Pauwels, P. und Roxin, A. [16] bei der Konvertierung von IFC-Modellen in Ontologien die Wegnamen der Beziehungsobjekte empfohlen, was die Komplexität deutlich verringert, siehe Kapitel 3.3.

Bei der Erstellung von ObjectProperties in der BRIDGE–Ontologie wurde folgenderweise vorgegangen. An der ersten Stelle wurde die Aufzählung der nötigen Beziehungen konzipiert. Folgend, wurden inverse Beziehungen erstellt. Weiterhin, wurden bei den ObjectProperties zusätzliche Charakteristiken definiert. Außerdem wurden Operationsbereiche (Domain/Range) festgelegt. Abschließend, wurden so genannte *Property Chain Axiome* erstellt.

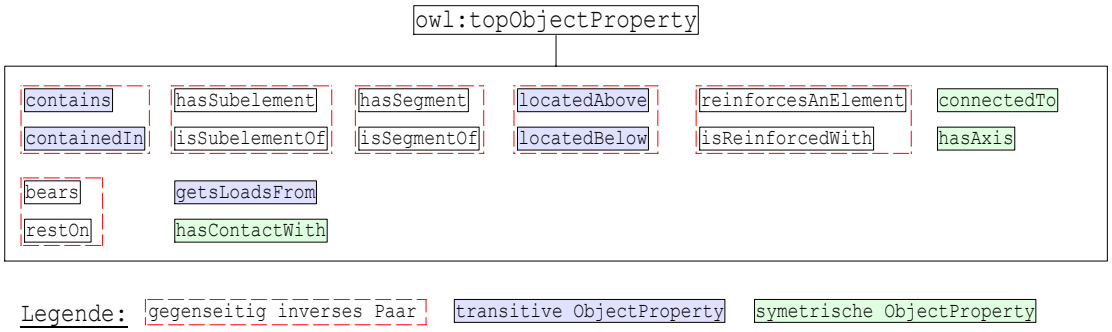


Abbildung 24: ObjectProperties in der BRIDGE–Ontologie

Grundsätzlich können ObjectProperties in zwei Gruppen gemäß ihrer Funktion unterteilt werden: räumliche (Aggregation, Zugehörigkeit, Position, Verbindung) und statische.

4.4.1 InverseOf

Bei der Erstellung einer ObjectProperty–Beziehung wurde eventuell eine zusätzliche Gegenbeziehung erstellt und als invers deklariert, siehe Liste 15.

Liste 15: Paare von inversen ObjectProperties

```
:contains owl:inverseOf :containedIn .
:hasSubelement owl:inverseOf :IsSubelementOf .
:hasSegment owl:inverseOf :IsSegmentOf .
:isReinforcedWith owl:inverseOf :reinforcesAnElement .
:locatedAbove owl:inverseOf :locatedBelow .
:bears owl:inverseOf :restOn .
```

Damit können die Beziehungen sowohl von einem Objekt aus zum Subjekt, als auch von einem Subjekt aus zum Objekt zugeordnet werden. Dies ermöglicht auch eine Schlussfolgerung des inversen Tripels, siehe Liste 16. Die ursprüngliche Richtung kann frei gewählt werden, da das Gegentripel automatisch geschlussfolgert wird.

Liste 16: Schlussfolgerung des inversen Tripels

```
# behauptetes Triple
bridgeABox:FOOTING_A14 bridge:isReinforcedWith bridgeABox:REINFORCING_BAR_07 .
# geschlussfolgertes Triple
bridgeABox:REINFORCING_BAR_07 bridge:reinforcesAnElement bridgeABox:FOOTING_A14 .
# behauptetes Triple
:BRIDGE :contains :SUPERSTRUCTURE .
# geschlussfolgertes Triple
:SUPERSTRUCTURE :containedIn :BRIDGE .
```

```
# behauptetes Triple
:PILE_15_6 :isReinforcedWith :REBAR_ARRAY_15_6 .
# geschlussfolgertes Triple
:REBAR_ARRAY_15_6 :reinforcesAnElement :PILE_15_6 .
```

4.4.2 Domain/Range

Eine Beziehung kann eigene Operationsbereiche sowohl für Objekte, als auch für Subjekte besitzen, sodass sich die Anordnung vom Objekt und Subjekt in einem Tripel auf diese Bereiche beschränkt. Der Bereich des Objektes wird als *Definitionsbereich* bzw. *Domäne* und der des Subjektes – *Wertebereich* genannt. Im Englischen werden die Wörter *domain* und *range* genutzt. In dieser Arbeit werden zukünftig die Wörter *Domäne* und *Definitionsbereich* durch `rdfs:domain` und *Wertebereich* durch `rdfs:range` ersetzt.

Beispiel: *(Das Beispiel wird sich auf das Gebiet des herkömmlichen Stahl- und Stahlbetonbaus beschränken.)* Die Beziehung `reinforcesAnElement` kann nicht auf eine Stahlstütze angewandt werden, da eine Stütze aus Stahl nicht bewehrt werden kann. Ebenfalls kann die Beziehung `isReinforcedWith` nicht zu einem Subjekt Passschraube führen. Die Anwendung der ObjectProperty `reinforcesAnElement` setzt voraus, dass das Objekt ein Bauelement aus Stahlbeton ist und sich bewehren lässt. D.h. dass `rdfs:range` für das Objekt auf Bauteile aus Stahlbeton beschränkt wird. Gleichfalls setzt die Anwendung der ObjectProperty `isReinforcedWith` voraus, dass das Subjekt ein Bewehrungselement ist. D.h. dass `rdfs:range` für das Subjekt auf Bewehrungselemente beschränkt wird.

Es ist anzumerken, dass die Definition von `rdfs:domain` und `rdfs:range` keine Restriktion an sich ist. Mit so einer Definition werden entsprechende Schlussfolgerungen bezüglich von `rdfs:domain` und `rdfs:range` durchgeführt.

Beispiel: Die Anordnung einer Passschraube als `rdfs:domain` bei der `reinforcesAnElement` Beziehung, wird nicht zu Inkonsistenz in der Ontologie führen, sondern es wird bloß geschlussfolgert, dass die Passschraube ein Bewehrungselement ist.

Die Festlegung von `rdfs:domain` und `rdfs:range` in der BRIDGE–Ontologie hat folgende Schlussfolgerungen ermöglicht, siehe Liste **??**.

Liste 17: Zuordnung eines Elements einem anderen Hauptelement

```
# behauptetes Triple
bridgeABox:ABUTMENT15 bridge:hasSubElement bridgeABox:FOOTING15 .
# geschlussfolgerte Triple
bridgeABox:ABUTMENT15 a bridge:Element .
bridgeABox:FOOTING15 a bridge:Element .
bridgeABox:FOOTING15 bridge:isSubElementOf bridgeABox:ABUTMENT15 .
```

Liste 18: Zuweisung eines Bewehrungselements einem anderen Element

```
# behauptetes Triple
bridgeABox:ELEMENT_15 bridge:isReinforcedWith bridgeABox:REINFORCINGBAR3 .
# geschlussfolgerte Triple
bridgeABox:ELEMENT_15 a bridge:Element .
bridgeABox:REINFORCINGBAR3 a bridge:ReinforcingElement .
bridgeABox:REINFORCINGBAR3 bridge:reinforcesAnElement bridgeABox:ELEMENT_15 .
```

4.4.3 Eigeschaften der ObjectProperties

Die Beziehung `bears` ist in der Realität transitiv, wird aber im herkömmlichen Bauwesen in der Bedeutung einer nicht transitiven genutzt. Eine Aussage, dass eine Bohrpfahl das Brückendeck trägt, wird nicht getroffen, da sie keinen direkten Kontakt haben, obwohl die Last tatsächlich durch ein anderes Bauteil abgeleitet wird. Wenn ein Bauteil direkt die Lasten vom anderen bekommt, wird der Sprachgebrauch des Wortes `trägt` korrekt sein.

Es wurde die zusätzliche Beziehung `getsLoadsFrom` eingeführt mit der transitive Eigenschaften korrekt verwendet werden. D.h. eine Bohrpfahl `bekommt die Lasten` vom Brückendeck.

Die ObjectProperty `hasAxis` sollte von dem Namen `situatedOnAxis` ersetzt werden. Z. B. ***Bauteil_A*** `situatedOnAxis` ***Achse_14***. Die inverse Implikation würde in dem Fall eine zusätzliche inverse Relation benötigen. Um dies zu vermeiden, wurde die Relation `hasAxis` genannt und symmetrisch eingestellt. Dies ermöglicht die Schlussfolgerung des umgekehrten Tripels, wobei sprachlich gesehen - entspricht der Name nicht komplett der Anwendung, siehe Liste 19.

Liste 19: Verwendung der `hasAxis` Relation

```
# behauptetes Triple
bridgeABox:COLUMN_14_1 bridge:hasAxis bridgeABox:AXIS_14 .
# geschlussfolgerte Triple
bridgeABox:AXIS_14 bridge:hasAxis bridgeABox:COLUMN_14_1 .
```

4.4.4 Herleitung der ObjektProperties

Im Rahmen der BRIDGE–Ontologie wurde so eine Konstruktion für die Herleitung anderer Beziehungen erstellt. Das ist eine Kombination aus *SubObjectPropertyOf*, *ObjectPropertyChain* und der Anordnung inverser Beziehungen, die eine komplexe Schlussfolgerung von anderen ObjectProperties durchführt.

Am besten wird es am nächsten Beispiel erläutert. Auf dem Bild 25 sind vier Objekte dargestellt. Zwischen diesen Objekten werden folgende Beziehungen betrachtet: `bears`, `restOn`, `locatedAbove`, `locatedBelow`, `hasContactWith`, `getsLoadsFrom`. Die bestehenden Beziehungen sind in der Liste 20 dargestellt.

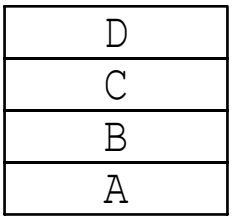


Abbildung 25: A B C D

Liste 20: Vorhandene Beziehungen, Beispiel Abbildung 25

```
#bears
:A :bears :B .
:B :bears :C .
:C :bears :D .
#restOn
:D :restOn :C .
:C :restOn :B .
:B :restOn :A .
#locatedAbove
:D :locatedAbove :C, :B, :A .
:C :locatedAbove :B, :A .
:B :locatedAbove :A .
#locatedBelow
:A :locatedBelow :B, :C, :D.
:B :locatedBelow :C, :D .
:C :locatedBelow :D .
#hasContactWith
:A :hasContactWith :B .
:B :hasContactWith :A, :C .
:C :hasContactWith :B, :D .
:D :hasContactWith :C .
#getsLoadsFrom
:A :getsLoadsFrom :B, :C, :D .
:B :getsLoadsFrom :C, :D .
:C :getsLoadsFrom :D .
```

Lediglich die Anordnung der `bears` bzw. `restOn` Beziehung ermöglicht die Schlussfolgerung der restlichen Relationen, siehe Liste 21 und Abbildung 26.

Liste 21: Ergebnis der BEARS–Property–Chain

```
# behauptete Triple
:A :bears :B .
:B :bears :C .
:C :bears :D .
# geschlussfolgerte Triple für :A
:A :getsLoadsFrom :B, :C, :D .
:A :locatedBelow :B, :C, :D .
:A :hasContactWith :B .
# geschlussfolgerte Triple für :B
:B :restOn :A .
```

```
:B :locatedAbove :A .
:B :locatedBelow :C, :D .
:B :hasContactWith :A, :C .
:B :getsLoadsFrom :C, :D .
# geschlussfolgerte Triple für :C
:C :restOn :B .
:C :locatedAbove :B, :A .
:C :locatedBelow :D .
:C :hasContactWith :B, :D .
:C :getsLoadsFrom :D .
# geschlussfolgerte Triple für :D
:D :restOn :C .
:D :locatedAbove :C, :B, :A .
:D :hasContactWith :C .
```

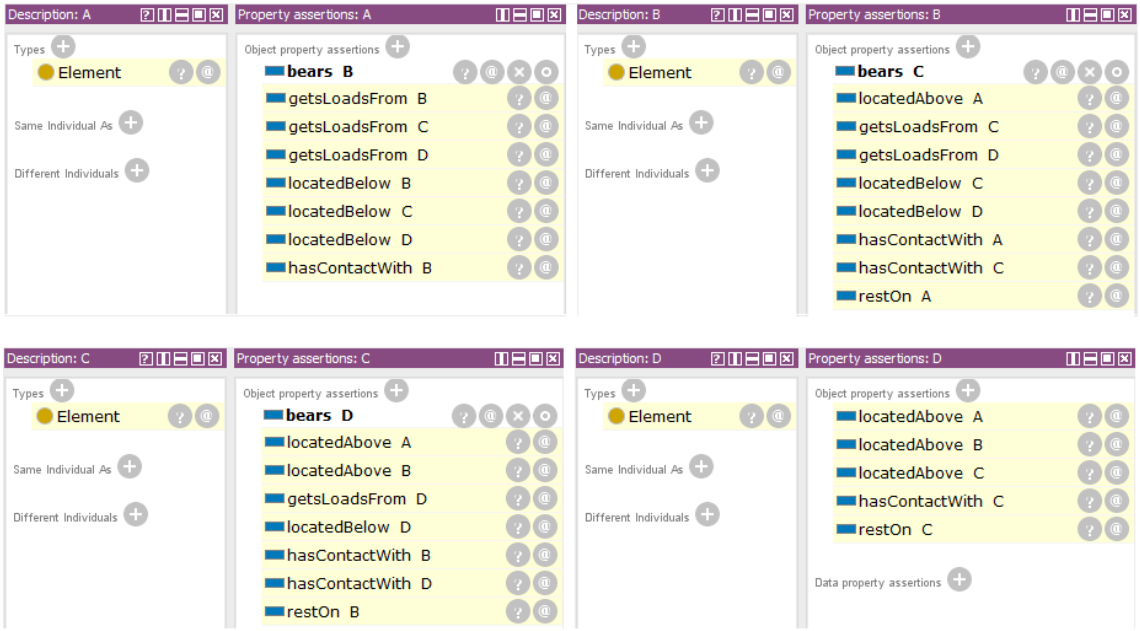


Abbildung 26: Screenshot aus Protégé

Erläuterung Unter einer Unterteilung Ober- und Unterklassen oder Ober- und Unterbeziehungen, wird folgendes verstanden. Eine Unterklasse vererbt sämtliche gesamte Eigenschaften ihrer Oberklasse. Das ist ein Prinzip, sowohl in den objektorientierten Sprachen, als auch in der Ontologiemodellierung mittels OWL. Von daher wird es bei Unterbeziehungen die Schlussfolgerung einer Oberbeziehung hervorrufen.

Da die ObjectProperty `bears` der ObjectProperty `locatedBelow`, sowie `getsLoadsFrom` untergeordnet wird, vererbt sie als die Unterbeziehung ihre gesamte Eigenschaften. Bei einer Anordnung `bears` zwischen zwei Objekten, wird ein neues Tripel mit der `locatedBelow` Beziehung impliziert. Ebenfalls wird die Unterordnung zu `hasContactWith`, die selbe Aktion bloß mit symmetrischen Eigenschaften hervorrufen.

Als Ergebnis werden von der ObjectProperty `bears` die Oberrelationen `locatedBelow`, `hasContactWith` und `getsLoadsFrom` impliziert, die die Herleitung von `locatedAbove` und `restOn` durch die Definition der inversen Beziehungen ermöglicht.

Die Beziehungen `locatedBelow`, `locatedAbove`, `hasContactWith` und `getsLoadsFrom` sind nicht für die Anwendung bei behaupteten Tripel gedacht. Bei der Anordnung der Relation `bears` werden sie automatisch geschlussfolgert. Die Relation `restOn` ist invers der Relation `bears`, sodass die Anwendung von beiden nicht nötig ist, da die inverse auch immer geschlussfolgert wird. Die Zweckdienlichkeit der Herleitung dieser ObjectProperties soll in den praktischen Anwendungen geprüft und an Anwendungsfälle angepasst werden.

Erläuterung der Property Chain Axiome In der Ontologie wurden ein *Property Chain Axiom* formuliert, siehe Liste 22.

Liste 22: Definition des Property Chain Axiomes

```
isSubElementOf ∘ hasAxis SubPropertyOf: hasAxis
```

Das Axiom kann folgend zerlegt werden, siehe Liste 23:

Liste 23: Zerlegung der Liste 22

```
A isSubElementOf B ∘ B hasAxis C ⊑ A hasAxis C
```

Das Axiom aus der Liste 23 besagt: wenn ein Oberelement ***A*** zu einer Achse gehört, gehören zu der Achse auch die weiteren Unterelemente, die von dem Element ***A*** aggregiert werden.

Es sollen dabei einige Restriktionen in dem Property Chain Axiom eingehalten werden, damit die Ontologie im Rahmen der OWL DL bleibt, siehe Kapitel 2.5.8. Mit dem OWL–Validator `http://visualdataweb.de/validator/` wurde die Ontologie bewertet und die Teilsprache OWL DL zugewiesen, siehe Abbildung 27.

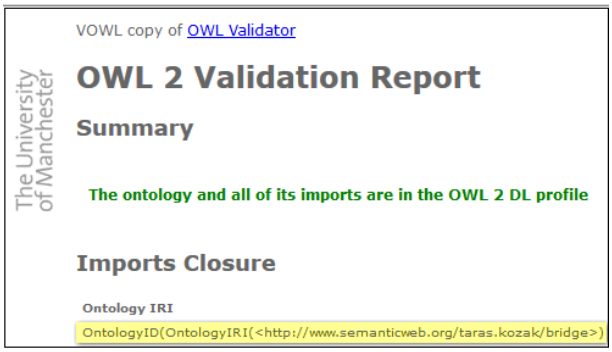


Abbildung 27: Bewertung der Ontologie

Weitere Herleitung von ObjectProeprties Mit der Anordnung von transitiven ObjectProperties, sowie ihrer inversen Paaren werden weitere Schlussfolgerungen möglich, siehe Liste 24.

Liste 24: Zuordnung einer Zone / eines Elements zu einer anderen Zone

```
# behauptete Triple
bridgeABox:BRIDGE bridge:contains bridgeABox:SUPERSTRUCTURE .
bridgeABox:SUPERSTRUCTURE bridge:contains bridgeABox:GIRDER.
```

```
# geschlussfolgerte Triple
bridgeABox:SUPERSTRUCTURE bridge:containedIn bridgeABox:BRIDGE .
bridgeABox:GIRDER bridge:containedIn bridgeABox:SUPERSTRUCTURE .
bridgeABox:BRIDGE bridge:contains bridgeABox:GIRDER .
bridgeABox:GIRDER bridge:containedIn bridgeABox:BRIDGE .
```

4.5 TBox: DatatypeProperties

Die vollständige Liste von DatatypeProperties ist in dem Anhang A.3 in der Tabelle 8 aufgeführt.

Durch die Präfixe wurden die Attribute in vier Gruppen eingeteilt. Die Gruppe `struct_` verweist auf die Zugehörigkeit zum ganzen Bauwerk. Die Gruppe `elem_` verweist auf die Zugehörigkeit zu einem Bauteil. Die Gruppe `mat_` bezieht sich auf die Materialeigenschaften. Schließlich, bezieht sich die Gruppe `doc_` auf schriftliche Dokumente.

4.6 Erstellung der ABox

Individuen Die ABox Komponente wurde anhand von einem IFC-Brückenmodell erstellt, siehe Abbildung 28. Das Modell wurde von dem Institut der Bauinformatik der Technischen Universität Dresden zur Verfügung gestellt.

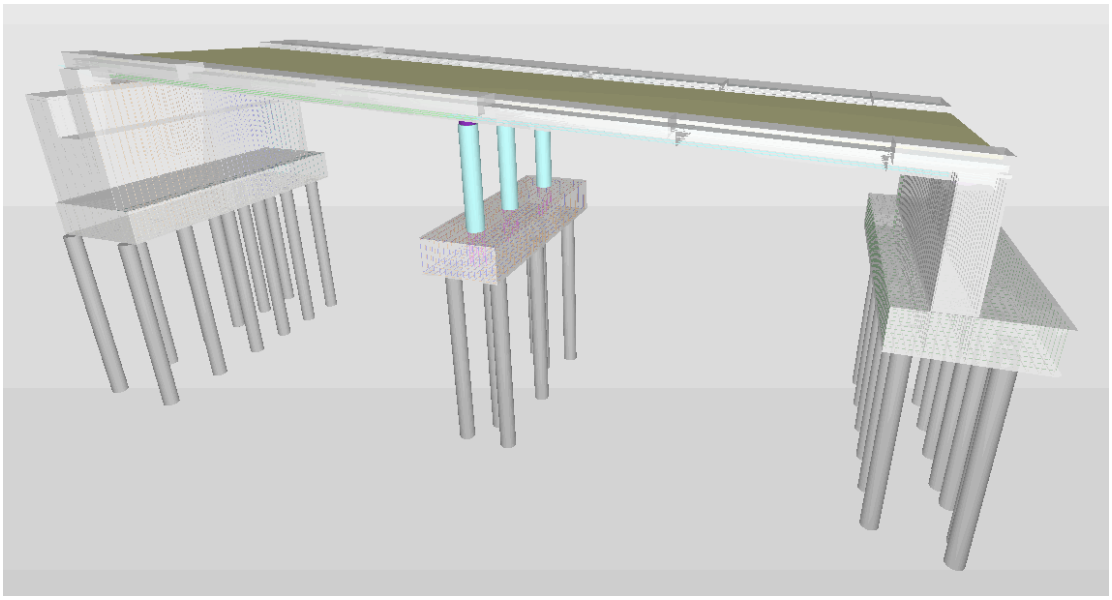


Abbildung 28: Brückenmodell für ABox
Quelle: Solibri Model Viewer

Anhand des Modells wurden die entsprechende Individuen erstellt. Die Erstellung erfolgte mit Hilfe der Regeln, aus Kapitel 5. Erstens wurden Individuen der Oberklasse `SpatialZone` mit folgenden Namen erstellt: `SITE`, `BRIDGE`, `SUBSTRUCTRE`, `SUPERSTRUCTURE`, `SPAN_1_AXIS_15_14`, `SPAN_2_AXIS_14_13`. Dann die entsprechende Bauteile, durch die Relation `contains` mit Zonen verknüpft wurden. Die zusammengesetzten Komponenten, wie Widerlager, Gründung und Brückendeck, wurden Oberelemente erstellt, die sich aus

Unterelementen mit Hilfe der Beziehung `hasSubElement` zusammensetzen. Des Weiteren wurden die Segmente des Brückendecks (Klasse `GirderSegment`) erstellt, die entsprechende Teile in sich mittels `contains` einschließen.Den Achsen wurden an erster Stelle die Komponenten zugeordnet, da die Zugehörigkeit der Unterelemente zur Achse schlussgefolgert wird. Folgend wurden die einzelnen Elemente, wie `Bearing` zugeordnet.

Statisch gesehen, wurden die Individuen lediglich mit der ObjectProperty `bears` bzw. `restOn` verbunden. Die Herleitung der anderen von der `bears` abhängigen Relation wird automatisch durchgeführt, siehe Kapitel 4.4.4.

Schließlich wurden die DatatypeProperties manchen Bauteilen zugeordnet, um die Anwendbarkeit der Abfragen des Kapitels 6 übersichtlich zeigen zu können.

Es sei gesagt, dass trotz des Fehlens von *Unique Name Assumption* in der OWL-Sprache alle Individuen als verschiedene Individuen definiert werden müssen, siehe Kapitel 2.5.10.

5 Erstellung von Regeln

Laut der Abbildung 1 stellen die Regeln (auf dem Bild – Unifying Logik) eine Schicht oberhalb der Ontologieschicht. Mittels Regeln kann die Ontologie um das weitere implizites Wissen erweitert werden [2].

Regeln sind Aussagen, die Ableitungen von neuen Fakten aus bekannten Fakten ermöglichen. Logische Regel sind Implikationen in Prädikatenlogik, die nur mit Hilfe der Logik definiert werden können [63].

Semantic Web Rule Language (SWRL) ist eine Regelsprache in der Form von Horn-Klauseln. Sie ist momentan noch kein endgültiger Standard bei W3C. Mit SWRL werden die Regel durch eine Implizierung zwischen der Antezedenz und Konsequenz definiert. Solange die Bedingungen in der Antezedenz erfüllt werden, müssen die Bedingungen in der Konsequenz erfüllt werden. Mehrere Bedingungen werden mit Konjunktion-Operator behandelt [61].

Eine detaillierte Darstellung dieses Punktes war im Rahmen dieser Arbeit nicht möglich, sodass auf die andere Regelarten und Regelsprachen nicht eingegangen wird. Das Ziel dieses Kapitel ist es die mögliche und nützliche Anwendung von Regeln zu zeigen, die als die Erweiterung des impliziten Wissens in einer Ontologie dienen können.

Regeln für BRIDGE–Ontologie Nach *ZTV-ING* (Zusätzliche Technische Vertragsbedingungen und Richtlinien für Ingenieurbauten) sind die Voraussetzungen bezüglich der Ausführung von der Umgebung abhängig. Für bewehrte Bauteile in Küstennähe, wo salzhaftige Luft vorhanden ist, werden andere Voraussetzung an Expositionsklassen gestellt. Somit lässt sich die Regel formulieren, die dies durch die Anordnung von Attributen berücksichtigen kann, siehe Liste 25.

In der Tabelle 5 ist ein Beispiel für Widerlager laut ZTV-ING aufgeführt. Die Regel, siehe Liste 25.

Liste 25: Regel 1: Vergabe von entsprechenden Werten der DatatypeProperties in Abhängigkeit von dem Wert einer anderen DatatypeProperty

```
Bridge(?b)^AbutmentElement(?x)^struct_location(?loc, ?val)
^swrlb:containsIgnoreCase(?val, "salty_air") -> mat_expositionClassReinforcementXC(?x,
"XC4") ^mat_expositionClassConcreteXF(?x, "XF1")^mat_expositionClassReinforcementXD(?x,
"XD1")^mat_expositionClassConcreteXF(?x, "XF2")^mat_moistureClassConcreteW(?x,
"WA")^mat_concreteCompressionStrengthClass(?x, "C25/30(LP)")^ elem_concreteCover(?x,
"40")^mat_monitoringClassConcrete(?x, "2")
```

Eine Regel mit SWRL wird erst dann erfüllt, wenn sowohl die Antezedenz als auch die Konsequenz erfüllt werden. Unter der Bedingung, dass nicht alle Atome der Antezedenz erfüllt werden können, kann die Konsequenz nicht erfüllt werden. Die Antezedenz bei SWRL ist nur dann erfüllt, wenn alle einzelne Bedingungen (Atome) erfüllt sind bzw. die Antezedenz leer ist.

	Widerlager *(Frost, Küstennähe)*	***Widerlager*** *(Straßenbrücken. Frost, Tausalzsprühnebel)*
Expositionsklasse durch Karbonatisierung	XC4	XC4
Expositionsklasse durch Chloride	—	XD1
Expositionsklasse durch Frostangriff	XF2	XF1
Feuchtigkeitsklasse	WA	WF
Mindestbetondeckung	C25/30(LP), C35/45	C25/30
Betondeckung	40	25
Überwachungsklasse	2	1

Tabelle 5: Bespiel. Vorasussetzungen nach ZTV-ING für Widerlager

Ein nicht erfülltes Atom in der Antezedenz verhindert die Ausführung der ganzen Regel. Daher ist die Erstellung der Regel, siehe Liste 26 nicht immer möglich. Diese Regel besagt, alle Unterklassen der Klasse `AbutmentElement` mittels ObjectProperty `hasSubElement` zu aggregieren.

Liste 26: ∗Aggregation aller Unterelementen der Klasse `AbutmentElement`

```
AbutmentBackWall(?a1)^AbutmentElementUserdefined(?a2)^AbutmentFooting(?a3)^
AbutmentHeadWall(?a4)^AbutmentWingWall(?a5)^AbutmentElement(?b)->
hasSubElement(?b,?a1)^hasSubElement(?b,?a2)^hasSubElement(?b,?a3)^
hasSubElement(?b,?a4)^hasSubElement(?b,?a5)
```

Als Lösung im Rahmen von SWRL wäre die Erstellung von mehreren Regeln, die jede Klasse einzeln enthalten möglich, siehe Liste 27. Die Regeln mit Individuen von Klassen, die in der ABox nicht vorhanden sind, werden nicht ausgeführt.

Liste 27: Regel 2: Aggregation von Individuen einer einzelnen Klasse

```
AbutmentBackWall(?a1)^AbutmentElement(?b)->hasSubElement(?b,?a1)
```

Weiter werden Regeln aufgeführt, deren Anwendung in der BRIDGE–Ontologie bei der vorhandenen Brücke durchgeführt wurde.

Liste 28: Regel 3: Erstellung einer Hierarchie zwischen `SpatialZone`–Unterklassen

```
Site(?a) ^ Bridge(?b) ^ Substructure(?c) ^ Superstructure(?d) ^ Span(?e) -> con-
tains(?a,?b) ^ contains(?b,?c) ^ contains(?b,?d) ^ contains(?d,?e)
```

Liste 29: Regel 4: Unterbau enthält immer Widerlager; Überbau entlät immer ein Brückendeck

```
Substructure(?s) ^ Superstructure(?u) ^ AbutmentElement(?b) ^ GirderElement(?g) ->
contains(?s, ?b) ^ contains(?u, ?g)
```

Es lässt sich zusammenfassen, dass die SWRL-Sprache für die Ausführung von komplizierteren Regeln nicht ausreichend ist. Es fehlt die Möglichkeit die Individuen nach einer bestimmten Klassifizierung abzurufen, was die Anwendung von Regeln nur auf alle (zusammen) Variablen einer Klasse bzw. ein konkretes Individuum beschränkt.

6 SPARQL–Abfragen

Das Finden und Abrufen der Informationen aus einer Datenbasis ist immer öfter notwendig [2]. Durch verschiedene Abfragen können die Informationen gefunden, klassifiziert, sortiert, geändert und für die weitere Verarbeitung bereitgestellt werden. Im Rahmen dieser Arbeit wird die Abfragesprache SPARQL (*SPARQL Protocol And RDF Query Language*) verwendet. Seit 2013 ist SPARQL 1.1 eine endgültige Empfehlung von W3C.

SPARQL ermöglicht vier Formen der Abfrage: `SELECT`, `ASK`, `DESCRIBE` und `CONSTRUCT`. `SELECT` wird am meisten genutzt.

Eine `SELECT` SPARQL-Abfrage besteht aus 3 Hauptteilen, die durch drei Schlüsselwörter `PREFIX`, `SELECT`, `WHERE` gekennzeichnet wird. Mit `PREFIX` wird der Namensraum deklariert. Das Schlüsselwort `SELECT` bestimmt allgemein das Ausgabeformat der Ergebnisse. Mit `WHERE` wird die eigentliche Abfrage eingeleitet [2].

SPARQL–Abfragen zu der BRIDGE–Ontologie Für die gezielten Filterung der Brückendaten in der BRIDGE–Ontologie wurden die Abfragen für die 12 folgenden Anwendungsszenarien erstellt, siehe Listen von 31 bis 42.

In den Listennamen werden die Abfragen im wesentlichen umschrieben. Dabei kann in Klammern gekennzeichnet werden, ob die Abfrage mit der Schlussfolgerung durchzuführen ist. Da die Ergebnisse der Abfragen an sich eher zweitrangig sind, werden sie hier nicht aufgeführt.

Liste 30: Erforderliche Präfixe für SPARQL-Abfragen

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX bridge: <http://www.semanticweb.org/taras.kozak/bridge#>
```

Liste 31: Angabe bezüglich der Anzahl von Achsen und Felder in der Brücke

```
SELECT (COUNT(?axis) AS ?numberOfAxis) ((?numberOfAxis - 1) AS ?numberOfSpans)
WHERE {
?axis a bridge:Axis . }
```

Liste 32: Anzeigen von Bauteilen die in der Achse 14 liegen (mit Schlussfolgerung)

```
SELECT DISTINCT ?x ?op ?axis
WHERE {
?axis a bridge:Axis .
?op a owl:ObjectProperty .
?x ?op ?axis .
FILTER(REGEX(STR(?axis), "14")) }
```

Liste 33: Berechnung des Alters von Bauteilen anhand vom Einbaujahr

```
SELECT DISTINCT ?elem ?yearOfInst ?age
WHERE {
?elem bridge:elem_yearOfInstallation ?yearOfInst
BIND((2019-?yearOfInst) AS ?age) }
```

Liste 34: Anzeige von Elementen, die ds Brückendeck tragen

```
SELECT ?bearings ?girder
WHERE {
?girder a bridge:GirderElement .
?bearings bridge:bears ?girder . }
```

Liste 35: Anzeige der vorhandenen Lagern in der Brücke mit der Zugehörigkeit zur Achse, sowohl die Anzeige von bestimmten DatatypeProperties

```
SELECT DISTINCT ?axis ?bearings ?dataProperty ?value
WHERE {
?x rdfs:subClassOf bridge:Bearing.
?bearings a ?x.
?dataProperty a owl:DatatypeProperty.
?bearings ?dataProperty ?value.
?bearings bridge:hasAxis ?axis.
FILTER( regex(STR(?dataProperty), "elem_bearingsDegreeOfFreedomDisplacement")
|| regex(STR(?dataProperty), "elem_yearOfInstallation")
|| regex(STR(?dataProperty), "elem_elementDescription")) }
ORDER BY ?axis ASC(?value)
```

Liste 36: Anzeige aller Komponenten und einfachen Elementen der Klasse `Substructure-Element` und im Falle dass die Komponenten noch aus anderen Elemente zusammengesetzt werden, werden die Unterelemente auch angezeigt

```
SELECT DISTINCT  ?elemLevel1 ?elemLevel2
WHERE {
{ bridge:BRIDGE bridge:contains ?elemLevel1 .
?elemLevel1 a bridge:SubstructureElement . }
UNION { ?elemLevel1 bridge:hasSubElement ?elemLevel2 . } }
```

Liste 37: Ausgabe von allen DatatypeProperties der BRIDGE Instanz

```
SELECT DISTINCT ?bridge ?property ?propertyValue
WHERE {
?property a owl:DatatypeProperty .
bridge:BRIDGE ?property ?propertyValue . }
```

Liste 38: Anzeige von allen ObjectProperties und DatatypeProperties über die das Element verfügt (mit Schlussfolgern)

```
SELECT DISTINCT  ?objectProperty ?object  ?dataProperty ?propertyValue
WHERE { { ?objectProperty a owl:ObjectProperty .
bridge:BEARING_13_2 ?objectProperty ?object . }
UNION {
?dataProperty a owl:DatatypeProperty .
bridge:BEARING_13_2 ?dataProperty ?propertyValue . } }
```

Liste 39: Anzeige von Bauteilen die über eine Betondeckung mehr als 30mm verfügen. Die Expositionsklasse wird mit dargestellt.

```
SELECT  DISTINCT ?stbElement ?dataProperty1 ?value1 ?dataProperty2 ?value2
WHERE {
{OPTIONAL{ ?dataProperty1 a owl:DatatypeProperty .
?stbElement ?dataProperty1 ?value1. } }
UNION
{OPTIONAL{ ?dataProperty2 a owl:DatatypeProperty .
?stbElement ?dataProperty2 ?value2 . }
FILTER (?value2>30) }
FILTER(REGEX(STR(?dataProperty1), "exposition")
|| REGEX(STR(?dataProperty2), "concreteCover") ) }
```

Liste 40: Berechnung der Anzahl von Bohrpfählen, die auf der Achse 13 liegen (mit Schlussfolgerung)

```
SELECT DISTINCT (COUNT(?piles) AS ?numberOfPiles)
WHERE {
?axis  bridge:hasAxis ?piles .
?piles a bridge:Pile .
FILTER( REGEX(STR(?axis), "13")) }
```

Liste 41: Anzeige aller räumlichen Zonen in der Ontologie (mit Schlussfolgerung)

```
SELECT DISTINCT ?zones
WHERE {
?zones  a bridge:SpatialZone . }
```

Liste 42: Anzeige von Subelemente des gesamten Brückendecks und der Widerlager

```
SELECT DISTINCT ?component ?o ?element ?axis
WHERE {
{ ?component a bridge:AbutmentElement;
bridge:hasSubElement ?element .
?o a owl:ObjectProperty .
?component ?o ?element .
?component bridge:hasAxis ?axis . }
UNION {
?component a bridge:GirderElement ;
bridge:hasSubElement ?element .
?o a owl:ObjectProperty .
?component ?o ?element . } }
ORDER BY ?1 ASC(?2)
```

7 Schlussbetrachtung

7.1 Zusammenfassung

In der vorliegenden Arbeit wurde eine Methode zur Verarbeitung der semantischen Informationen von IFC-Brückenmodellen erarbeitet und untersucht.

Es wurde sich für den Einsatz von Ontologien zur Informationsverarbeitung entschieden. Für die Modellierung der Ontologie wurde wegen ihrer starken Aussagekraft und Entscheidbarkeit die OWL DL Sprache ausgewählt.

Zuerst wurde die terminologische Komponente erstellt. Die Kernkonzepte der Ontologie stellen eine räumliche Zone und ein Element dar. Der Klasse `Element` werden die semantisch relevanten tragende und nicht tragenden Bauteile von Massivbrücken zugeordnet. Die Klasse `SpatialZone` enthält die brückenspezifischen Zonen, wie Unterbau, Überbau, Feld oder Segment.

Die Relationen lassen sich grundsätzlich auf drei Arten klassifizieren: statische, räumliche und Zugehörigkeits-Relationen. Die Attribute sind auch in Anlehnung an die DIN 1076 durchgeführt, was den Einsatz bei der Brückenprüfung und -sanierung ermöglicht.

Diverse Konstrukte, über welche die OWL DL Sprache verfügt, ermöglichten die Erstellung zusätzlicher Konstruktionen in der Ontologie, die zu einem höheren Nutzen führten. Mit dem Vererbungsprinzip und dem Einsatz des Property Chain Axioms konnten mehrere Tripel lediglich durch die Anordnung einer `bears` Relation geschlussfolgert werden.

Des Weiteren wurden Regel mittels der SWRL (*A Semantic Web Rule Language Combining OWL and RuleML*) Sprache erstellt, welche zusätzliche Wissensverarbeitung ermöglicht haben. Darauf folgend wurden diverse Abfragen mittels SPARQL-Abfragesprache konzipiert und an dem erstellten Modell angewandt.

Abschließend sei gesagt, dass das Ziel dieser Arbeit erfüllt wurde. Die Anwendung von Semantischen Technologien (die aktuell immer interessanter werden) im Thema BIM im Bezug auf Brückenbau wurde untersucht und erprobt.

Zur Ermöglichung der semantischen Beschreibung von Brückenmodellen wurde die terminologische Komponente erarbeitet, die ein großes Spektrum an Brückenbauten, sowohl im Neubau, als auch im Bestand abdeckt. Die Anwendung von Reasoner anhand der Ontologie ermöglicht die Schlussfolgerung des neu impliziten Wissens. Dies wurde bestätigt und zeigt ihre Zweckdienlichkeit. Die zahlreichen Attribute ermöglichen die Anwendung der Methode im Sinne einer Datenbank, für das Lagern von Informationen über die Brücke für die Prüfung, sowie SIB-Bauwerke momentan. Schließlich wurde ihre Funtionstüchtigkeit und Zweckdienlichkeit bewiesen, was die Grundlage für weitere Untersuchungen darstellt.

7.2 Ausblick

Die Anwendung von Ontologien setzt Kenntnisse der Aussagenlogik vorraus, was die Anwendung für Bauingenieure in der Praxis erschwert. Vorallem die effektive Anwendung des

Property Chain Axioms verlangt fortgeschrittene Kenntnisse der Logik, da ihre willkürliche Anordnung leicht zu Unentscheidbarkeit führen kann. Um die Ontologie im Rahmen der OWL DL zu halten ist es notwendig die verwendeten Konstrukte zu kennen und zu verstehen.

Des Weiteren ist es nicht sinnvoll, die Geometrie in die Ontologie mit aufzunehmen. Dafür wurden andere Formate erarbeitet , welche bereits erfolgreich eingesetzt werden. Das Hinzuladen der Geometrien führt zu einer sehr hohen Anzahl von Tripeln, womit das Modell auf Grund seiner Komplexität kaum noch zu bearbeiten wäre. Die RDF-Bausteine sind zur Beschreibung von Meta Daten konzipiert und nicht für die Repräsentation von Geometrien.

Die Verwendung von Ontologien weißt aber folgende Vorteile auf. Die Nutzung des Linked Data Prinzips ermöglicht nicht nur die Abfrage der Daten aus dem Modell, sondern ebenso aus weiteren externen Daten. Die Heterogenität wird reduziert.

Bei Ontologien prüft ein Reasoner den Inhalt auf Inkonsistenz und spürt somit vorhandene Fehler auf. Bei IFC-Modellen wird keine Prüfung auf Inkonsistenz vorgenommen, womit die Anzahl der Fehler höher liegen kann. In den Ontologien treten Inkonsistenzen nur vereinzelt auf. Da diese sichtbar sind, können die Fehler direkt behoben werden.

Durch die Standardisierung von Web-Formaten bei W3C-Konsortium kann die einheitliche Repräsentation von Daten erreicht werden, welches eine Verbesserung der Interoperabilität gewährleistet.

Darüber hinaus werden Ontologien nicht nur speziell von und für das Bauingenieurwesen entwickelt, sondern finden in allen IT basierten Branchen weite Verbreitung. Im Gegensatz dazu, wird IFC nur von Bauingenieuren entwickelt und ist speziell auf deren Bedürfnisse angepasst. Semantische Technologien werden bereits in anderen Branchen eingesetzt. Durch Adaptionen könnten diese Technologien an das Bauingenieurwesen angepasst werden.

Weiterhin bietet der Ansatz der Multimodellen die Möglichkeit, die Modelle verschiedener Fachrichtungen miteinander zu koppeln. In externen Linkmodellen, werden die Informationen der diversen Fachplaner zusammengeführt. Dies ermöglicht den Austausch der Informationen zwischen allen Projektbeteiligten [10].

Abschließend muss noch erwähnt werden, dass die Anwendung von Graphen in diesem Fall nicht möglich wäre. Die Graphdatenbank auf der RDF-Basis bestehen aus Tripel „Subjekt – Prädikat – Objekt“. Dabei sind sowohl die Knoten, als auch die Kanten des Graphen immer eine Ressource, die eine eineindeutige URI-Nummer besitzt. Die Kanten in Neo4j-Graphen können mit den Kanten in einem RDF-Graphen gleichgestellt werden. Hingegen werden die Knoten in Neo4j-Graphen und einem RDF-Graphen unterschiedlich dargestellt. In Neo4j können die Knoten als Container bezeichnet werden, die eine beliebige Anzahl von Attributen enthalten können, wobei im RDF-Tripel die Knoten die gleichen atomaren Einheiten wie die Kanten sind [62].

Literatur

[1] Dengel, A.: Semantische Technologien. Grundlagen, Konzepte, Anwendungen. Spektrum Akademischer Verlag, Heidelberg, 2012, S. 10-126, 202-242

[2] Hitzler, P.; Krötzsch, M.; Rudoplh, S.; Sure, Y.: Semantic Web. Springer-Verlag, Berlin Heidelberg, 2008, S. 9-198; 254-257

[3] Yu, L.: A Developer's Guide to the Semantic Web, Second Edition. Springer-Verlag Berlin Heidelberg, 2014. S. 121-262

[4] Scherer, R. J.; Schapke, S.: Informationssysteme im Bauwesen 1. Modelle, Methoden und Prozesse, Springer-Verlag Berlin Heidelberg, 2014, S. 87-115, 296, 360

[5] Gašević, Dragan, G.; Dragan, D.; Vladan, D.: Model Driven Engineering and Ontology Development. 2nd Edition. Springer-Verlag Berlin Heidelberg, 2009, S. 4-124

[6] Antoniou, G.; van Harmelen, F.: A Semantic Web Primer. Second edition, Massachusetts Institute of Technology, 2003, S. 119-132

[7] Borrman, A.; König, M.; Koch, C.; Beetz, J.: Building Information Modeling. Technologische Grundlagen und industrielle Praxis. Springer Vieweg Verlag, Wiesbaden, 2015, S. 83-126

[8] Hausknecht, K.; Liebich, T.: BIM-Kompendium, Building Information Modeling als neue Planungsmethode. Fraunhofer IRB Verlag, Stuttgart, 2016, S. 18-25

[9] Diestel, R.: Graphentheorie - 4. Aufl., Heidelberg, Springer, 2010, S. 2-13

[10] Fuchs, S.: Dissertation. Erschließung domänenübergreifender Informationsräume mit Multimodellen, Institut für Bauinformatik, Fakultät Bauingenieurwesen, TU Dresden, 2015, S. 45

Research Papers

[11] Yabuki, N.; Lebegue, E.; Gual, J.; Shitani, T.; Zhanatao, Z.: International colaboration for developing the bridge product model "IFC-BRIDGE". Joint International Conference on Computing and Decision Making in Civil and Building Engineering, Montréal,Canada, June 14-16, 2006

[12] Abburu, S.: A Survey on Ontology Reasoners and Comparison, International Journal of Computer Applications (0975 – 8887), Volume 57– No.17, November 2012

[13] Rasmussen, M. H.; Pauwels P.; Lefrançoi, M.; et al.: Recent changes in the Building Topology Ontology, November 2017 `https://hal-emse.ccsd.cnrs.fr/emse-01638305/`, zuletzt geprüft am 24.03.2019

[14] Pauwels, P.; Terkaj, W.: EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology. Automation in Construction, 63:100–133, 2016

[15] Mendes de Farias, T.; Roxin, A.; Nicolle, C.: IfcWoD, Semantically Adapting IFC Model Relations into OWL Properties. In Proceedings of the 32nd CIB W78 Conference on Information Technology in Construction, 175-185, 2015

[16] Pauwels, P., Roxin, A.: SimpleBIM. From full ifcOWL graphs to simplified building graphs. In Proceedings of the 11th European Conference on Product and Process Modelling (ECPPM), pages 11–18, Limassol, Cyprus, CRC Press, September 2016

[17] Nešić, S.; Rizzoli, A. E.; Athanasiadis, I. N.: Towards a Semantically Unified Environmental Information Space. Environmental Software Systems. Frameworks of eEnvironment. IFIP Advances in Information and Communication Technology, Volume 359, pp. 407-418, 2011

[18] Farias, M. T.; Roxin, A.; Nicolle, C.: A Rule Based System for Semantical Enrichment of Building Information Exchange. In CEUR Proceedings of RuleML (4th Doctoral Consortium), Volume 1211, pp. 2-9, 2014

[19] Bernadette Farias Lóscio, Caroline Burle, Newton Calegari, Annette Greiner, Antoine Isaac, Carlos Iglesias, and Carlos Laufer. Data on the Web Best Practices. W3C Recommendation, W3C, January 31 2016

[20] Ren, G.; Ding, R.; Li, H.: Building an ontological knowledgebase for bridge maintenance. Advances in Engineering Software, 130:24-40, 2019

[21] Zhang, C.; Beetz, J.: Querying Linked Building Data Using SPARQL with Functional Extensions, `https://www.researchgate.net/publication/308057797`, September 2016

[22] Krötzsch, M.; Simančík, F.; Horrocks, I.: A Description Logic Primer, Department of Computer Science, University of Oxford, UK, 3. Juni 2013 `https://arxiv.org/abs/1201.4089`, zuletzt geprüft am 29.02.2019

[23] Zhang, C.; Beetz, J.; de Vries B.: BimSPARQL: Domain-specific functional SPARQL extensions for querying RDF building data, Semantic Web. 1-27. 10.3233/SW-180297, 2018

[24] Venugopal, M.; Eastman C. M.; Sacks R.; Teizer, J.: Semantics of model views for information exchanges using the industry foundation class schema. Advanced Engineering Informatics, 26(2):411–428, 2012

[25] Beetz, J.; van Leeuwen J.; Zhang, C.; de Vries B.: IfcOWL. A case of transforming EXPRESS schemas into ontologies. Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 23(1):89–101, 2009

[26] Solihin, W.; Eastman, C.: Classification of rules for automated bim rule checking development. Automation in Construction, 53:69–82, 2015

[27] Pauwels P.; Zhang, S.; Lee Y.-C.: Semantic web technologies in AEC industry: A literature overview. Automation in Construction, 73:145–165, 2017 `https://doi.org/10.1016/j.autcon.2016.10.003`.

[28] Sacks, R.; Kedar, A.; Borrmann, A.; Ma, L.; Singer, D.; Kattel, U.: SeeBridge Information Delivery Manual (IDM) for Next Generation Bridge Inspection, 33rd International Symposium on Automation and Robotics in Construction (ISARC 2016)

Internet

[29] Semantik. Definition, Entstehung und Entwicklung, `https://www.buecher-wiki.de/index.php/BuecherWiki/Semantik`, zuletzt geprüft am 07.03.2010

[30] Entität `IfcSpatialZone`, `http://www.buildingsmart-tech.org/ifc/IFC4x1/final/html/schema/ifcproductextension/lexical/ifcspatialzone.htm`

[31] Documentation IFC4x1 Official Release `http://www.buildingsmart-tech.org/ifc/IFC4x1/final/html/`

[32] Konev, B.: Ontologie und Wissensrepräsentation. Vorlesungsreihe, `https://www.lektorium.tv/course/22781`, zuletzt geprüft am 23.02.2019

[33] OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition), `https://www.w3.org/TR/2012/REC-owl2-syntax-20121211/`, zuletzt geprüft am 01.12.2018

[34] Protégé Desktop User Documentation, `https://protegewiki.stanford.edu/wiki/Protege4UserDocs`

[35] Stufenplan / BIM-Leistungsniveau ab 2020, `https://bim4infra.de/stufenplan/`, zuletzt geprüft am 21.03.2019

[36] Wissenschaftliche Begleitung der BMVI Pilotprojekte zur Anwendung von BIM im Infrastrukturbau. Zwischenbericht, `https://www.bmvi.de/SharedDocs/DE/Anlage/Digitales/bim-materialsammlung.html`, zuletzt geprüft am 21.03.2019

[37] Wissenschaftliche Begleitung der BMVI Pilotprojekte zur Anwendung von BIM im InfrastrukturbauEndbericht. Handlungsempfehlungen, `https://www.bmvi.de/SharedDocs/DE/Anlage/DG/wissenschaftliche-begleitung-anwendung-bim-infrastrukturbau-2018.pdf?__blob=publicationFile`, zuletzt geprüft am 21.03.2019

[38] Pilotprojekte vom BMVI, `http://infrabim.de/pilotprojekte`, zuletzt geprüft am 21.03.2019

[39] Reformkommission Bau von Großprojekten, `https://www.bmvi.de/SharedDocs/DE/Artikel/G/reformkommission-bau-von-grossprojekten.html`, zuletzt geprüft am 21.03.2019

[40] Leitfaden Großprojekte, `https://www.bmvi.de/SharedDocs/DE/Anlage/VerkehrUndMobilitaet/leitfaden-grossprojekte.html?nn=12830`, zuletzt geprüft am 21.03.2019

[41] ARGE BIM4INFRA2020, `https://bim4infra.de/`, zuletzt geprüft am 21.03.2019

[42] IFC-INFRA. IFC für den Infrastrukturbereich, http://ifcinfra.de/, zuletzt geprüft am 23.03.2019

[43] IFC4.Scope, http://www.buildingsmart-tech.org/ifc/IFC4/final/html/schema/chapter-1.htm, zuletzt geprüft am 23.03.2019

[44] Dokumente und Protokolle aus Expertentreffen. Dokumente/Downloads/IFC-Bridge/1., 2., 3. Expertentreffen http://ifcinfra.de/documents/, zuletzt geprüft am 25.03.2019

[45] Dokument aus 1. Expertentreffen, Dokumente/Downloads/IFC-Bridge/1.Expertentreffen/*2017-01-23_IFC-Bridge_Begleitung_Start.pdf*, http://ifcinfra.de/documents/, zuletzt geprüft am 28.03.2019

[46] Infrastructure Room, buildingSMART International, https://www.buildingsmart.org/standards/rooms-and-groups/infrastructure-room/, zuletzt geprüft am 26.03.2019

[47] Lastest Schema 4.2, Topics, https://forums.buildingsmart.org/c/developers-ifc/schema-ifc42, zuletzt geprüft am 26.03.2019

[48] IFC-Bridge Fast Track Project, Report WP2: Conceptual Model, *2018-04-16_04_WP2_ConceptualModelReport.pdf* des 3. Expertentreffens vom 12.04.2018 http://ifcinfra.de/documents/, zuletzt geprüft am 25.03.2019

[49] Protokoll vom 2. Expertentreffen *IFC-Bridge*, *2017-09-25_00_Protokoll_Expertentreffen_IFC-Bridge.pdf* http://ifcinfra.de/documents/, zuletzt geprüft am 25.03.2019

[50] Dokument vom 2. Expertentreffen *IFC-Bridge*, *2017-09-25_04_IFC-Bridge_SpatialStructure.pdf* http://ifcinfra.de/documents/, zuletzt geprüft am 25.03.2019

[51] Dokument vom 3. Expertentreffen *IFC-Bridge*, *2018-04-16_04_IFC-Bridge_Instance_Diagrams.pdf* http://ifcinfra.de/documents/, zuletzt geprüft am 26.03.2019

[52] Files from InfraRoom administrator at buildingSMART,IFC-Bridge, Draft IFC4.2, *IR-Bridge_Overview.pdf* https://buildingsmart.sharefile.com/share/view/s619ceb7623d4b099/fod4202e-f039-49d7-8e4e-cddabe6efd3c/, zuletzt geprüft am 28.03.2019

[53] Files from InfraRoom administrator at buildingSMART,IFC-Bridge, Draft IFC4.2, https://buildingsmart.sharefile.com/share/view/s619ceb7623d4b099/fod4202e-f039-49d7-8e4e-cddabe6efd3c/, zuletzt geprüft am 28.03.2019

[54] Heath, T.; Bizer, C.: Linked Data. Evolving the Web into a Global Data Space. Morgan & Claypool, 1st Edition, 2011 http://linkeddatabook.com/, , zuletzt geprüft am 29.03.2019

[55] OWL 2 Rules, http://www.semantic-web-book.org/w/images/7/7a/OWL2-Rules-GeoS2009.pdf, zuletzt geprüft am 01.04.2019

[56] Syntax OWL https://www.w3.org/TR/owl-syntax/, zuletzt geprüft am 01.04.2019

[57] Horridge, M.: A Practical Guide To Building OWL Ontologies Using Protege 4 and CO-ODE Tools Edition 1.3 http://mowl-power.cs.man.ac.uk/protegeowltutorial/resources/ProtegeOWLTutorialP4_v1_3.pdf, zuletzt geprüft am 01.04.2019

[58] Bewehrungskorb, https://www.dornbach.com/de/baulexikon/bewehrungskorb.html, zuletzt geprüft am 08.04.2019

[59] OWL 2, Structural Specification and Functional-Style Syntax (Second Edition) https://www.w3.org/TR/owl2-syntax/#Global_Restrictions_on_Axioms_in_OWL_2_DL, zuletzt geprüft am 08.04.2019

[60] Fakten, Regeln, Klauseln, Grundbegriffe http://www.info-wsf.de/index.php/Fakten%2C_Regeln%2C_Klauseln%2C_..._Grundbegriffe%21%21%21, zuletzt geprüft am 14.04.2019

[61] SWRL: A Semantic Web Rule Language Combining OWL and RuleML https://www.w3.org/Submission/SWRL/, zuletzt geprüft am 14.04.2019

[62] Neo4j, Official Website https://neo4j.com/, zuletzt geprüft am 14.04.2019

Andere Quellen

[63] Vorlesungen Modul BIW 3-13, TU Dresden, Prof. Dr. Raimar J. Scherer

[64] Vorlesungen, Massivbrückenbau, Modul BIW 4-16, TU Dresden, Univ.-Prof. Dr.-Ing. Dr.-Ing. E. h. Manfred Curbach

[65] ISO 12006-2: Building construction - Organization of information about construction works. Part 2: Framework for classification. Second edition 2015-05-01

Bildquellen

[66] https://upload.wikimedia.org/wikipedia/commons/f/f3/Semantic_Web_Stack.png

[67] http://img.archiexpo.de/images_ae/photo-g/144233-8853295.jpg

[68] http://img.archiexpo.com/images_ae/photo-g/126411-6507243.jpg

[69] https://upload.wikimedia.org/wikipedia/commons/f/f3/Langwiesbridge.jpg

[70] https://i.pinimg.com/originals/66/1f/f2/661ff23d43e679278edcb0ee120c483c.jpg

[71] http://liuzhou.co.uk/wordpress/wp-content/uploads/2014/08/DSC04693.jpg

[72] https://d2v9y0dukr6mq2.cloudfront.net/video/thumbnail/tb5f8Tf/4k-timelapse-of-the-humber-bay-arch-bridge_hbbyptjh__F0000.png

[73] https://upload.wikimedia.org/wikipedia/commons/5/54/General_Belgrano_Bridge_and_Corrientes_beach.jpg

[74] https://farm8.static.flickr.com/7506/15464890454_5e36332f56_b.jpg

[75] http://technogroup.es/wp-content/uploads/2014/07/Fotolia_8538056_Subscription_L.jpg

[76] https://www.sfs.biz/sfs_download/media/general_media/locher_bewehrungen/bs_bb_ba/produkte/Titelbild_Bohrpfahl_1250x938_ContentHeader.jpg

[77] https://images.ecosia.org/NFlM3v0yNMyDdlCciGbdHi7goHs=/0x390/smart/http%3A%2F%2Ffiles1.structurae.de%2Ffiles%2Fphotos%2F1%2F20090712%2Fdsc07642.jpg

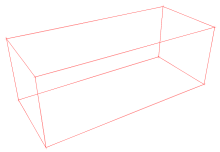
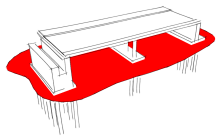
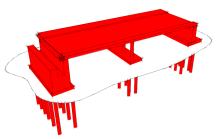
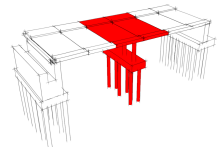
[78] https://upload.wikimedia.org/wikipedia/commons/b/b6/Bridge_over_the_Magdalena_River_%2816049525434%29.jpg

A Anhang

A.1 Beschreibung der Klassen

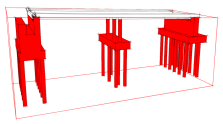
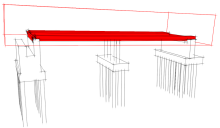
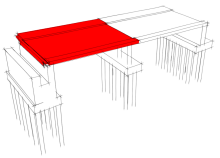
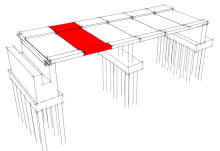
Hinweis: *Um die Eindeutigkeit bei der Beschreibung zu gewährleisten, sind die Abbildungen gegeben, die die Beschreibungen anschaulich vervollständigen. Bei Entitäten, wo keine Abbildungen vorhanden sind, sind die Informationen der Beschreibung zu entnehmen.*

Tabelle 6: Beschreibung der Klassen

№	**Klasse / Beschreibung**	**Abbildung**
1	`SpatialZone` → Räumliche Zone --- `SpatialZone` ist eine nicht hierarchische und möglicherweise eine überlappende Gliederung eines Projekts [30]. Eine räumliche Zone kann physikalische Elemente, räumliche Zonen, sowie Komponenten enthalten. Die Zugehörigkeit zu `SpatialZone` kann mit der Relation `contains` erstellt werden, siehe Tabelle 7.	
2	`Site` → Gelände --- Gelände ist eine Geländefläche, wo das Projekt ausgeführt wird. Das Gelände widerspiegelt das Relief des Geländes und kann andere für das Projekt relevante Objekte enthalten, z. B. andere Bauwerke oder geographische Objekte.	
3	`Bridge` → Brücke --- Die Klasse entspricht dem ganzen Brückenbauwerk mit allen zugehörigen Elementen, die die Funktionalität des Bauwerks gewährleisten.	
4	`ArbitraryZone` → Eine beliebige Zone --- Der Klasse können beliebige Elemente, Komponente und räumliche Zonen, vollständig oder teilweise, zugeordnet werden. Dies gewährleistet Flexibilität bei der projektspezifischen Gliederung.	

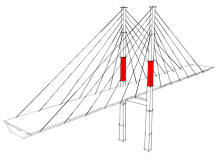
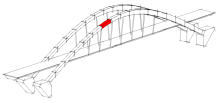
Fortsetzung auf der nächsten Seite ⇒

Tabelle 6: Beschreibung der Klassen. Fortsetzung

№	**Klasse / Beschreibung**	**Abbildung**
5	`Substructure` → Unterbau --- Der Klasse gehören Elemente, sowie andere Zonen, die sich unterhalb des Brückendecks befinden. *Z. B. Widerlager, Pfeiler, Gründung, unterer Teil des Pylons bzw. Bogens.*	
6	`Superstructure` → Überbau --- Der Klasse gehören Elemente, sowie Komponenten, die sich über dem Unterbau befinden. *Z. B. Brückendeck, Seiltragwerk, oberer Teil des Pylons bzw. Bogens.*	
7	`Axis` → Achse --- Mit der Klasse können Achsen in Querrichtung definiert werden. Im Rahmen dieser Arbeit können die Achsen nur quer definiert werden. Die Elemente bzw. Komponente die in einer Achse liegen, können durch die Relation `hasAxis` der Achse zugewiesen werden, siehe Tabelle 7.	
8	`Span` → Feld --- Die Klasse beschreibt ein Feld einer Brücke. Mittels zwei Achsen und der Relation `belongsToAxis` kann das Feld definiert werden.	
9	`GirderSegment` → Brückendecksegment --- `GirderSegment` ist ein Segment eines Brückendecks, das eine Fertigteilkomponente bzw. ein Segment eines Betonierabschnitts abbilden kann.	

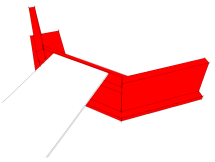
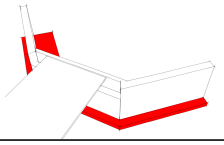
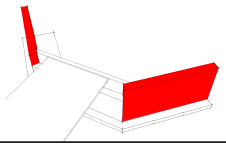
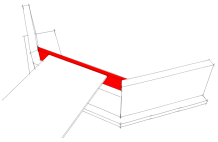
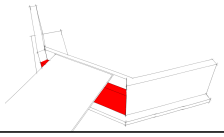
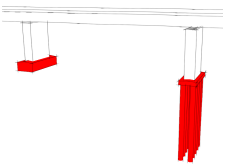
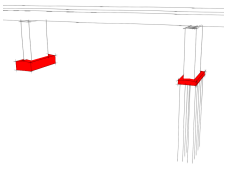
Fortsetzung auf der nächsten Seite ⇒

Tabelle 6: Beschreibung der Klassen. Fortsetzung

№	**Klasse / Beschreibung**	**Abbildung**
10	`TowerSegment` → Pylonsegment --- `TowerSegment` ist ein beliebiges Segment eines Pylons, das beispielsweise einen Betonierabschnitt abbilden kann.	
11	`ArchSegment` → Bogensegment --- `ArchSegment` ist ein beliebiges Segment eines Bogens, das eine Fertigteilelement bzw. ein Segment eines Betonierabschnitts abbildet.	
12	`Element` → Brückenelement --- Der Klasse `Element` gehört jedes physikalisches Element einer Brücke. ***Hinweis**: alle Unterklassen der Klasse `Element` sind physikalische Elemente.*	
13	`LoadBearingElement` → Tragwerkselement einer Brücke --- Der Klasse gehört jedes Element, das über eine Tragfunktion in einer Brücke verfügt.	
14	`SubstructureElement` → Unterbaülement einer Brücke --- Der Klasse gehört jedes tragendes Element des Unterbaus einer Brücke.	
15	`SuperstructureElement` → Überbaülement einer Brücke --- Der Klasse gehört jedes tragendes Element des Überbaus einer Brücke.	

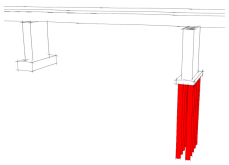
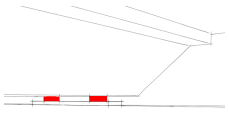
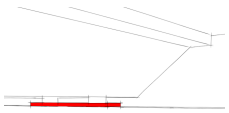
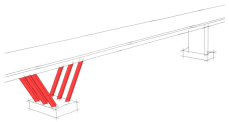
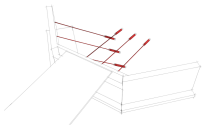
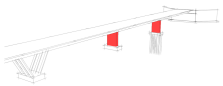
Fortsetzung auf der nächsten Seite ⇒

Tabelle 6: Beschreibung der Klassen. Fortsetzung

№	**Klasse / Beschreibung**	**Abbildung**
16	`AbutmentElement` → Widerlagerelement --- Die Klasse enthält weitere Unterelemente des Widerlagers. ***Hinweis:*** *Die Klasse dient nicht nur als eine Superklasse für die weiteren Unterelemente des Widerlagers. Sie kann auch für den gesamten Widerlager benutzt werden, der aus Vereinfachungsgründen als ein einziges Element modelliert wurde.*	
17	`AbutmentFooting` → Fundamentplatte eines Widerlagers	
18	`WingWall` → Flügelwand eines Widerlagers	
19	`ChamberWall` → Chamber wall of an abutment → Kammerwand eines Widerlagers	
20	`AbutmentBenching` → Auflagerbank eines Widerlagers	
21	`AbutmentElementUserdefined` → Ein benutzerdefiniertes Element des Widerlagers	
22	`FoundationElement` → Gründungselement --- Die Klasse enthält weitere Unterelemente einer Gründung.	
23	`Footing` → Gründung	

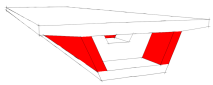
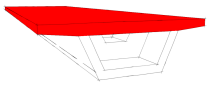
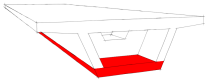
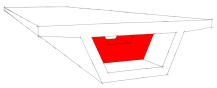
Fortsetzung auf der nächsten Seite ⇒

Tabelle 6: Beschreibung der Klassen. Fortsetzung

№	**Klasse / Beschreibung**	**Abbildung**
24	`Pile` → Pfahl	
25	`Bearings` → Lager --- Die Klasse enthält die gängigen Typen der Brückenlager: `DiscBearings` → Scheibenlager `ElastomericBearings` → Elastomerlager `HorizontalForceBearings` → Horizontalkraftlager `PotBearings` → Topflager `SlidingBearings` → Gleitlager `SphericalBearings` → Kalottenlager `BearingsUserdefined` → Benutzerdefiniertes Lager	
26	`BearingsSeat` → Lagersockel	
27	`Column` → Stütze --- Im Unterschied zum Pfeiler, wird unter einer Stütze ein ***schlankes*** vertikales bzw. geneigtes tragendes Bauteil verstanden.	
28	`GroundAnchoringDevice` → Erdverankerung	
29	`Pier` → Pfeiler	
30	`GirderElement` → Brückendeck Element --- Die Klasse enthält weitere Unterelemente eines Brückendecks.	

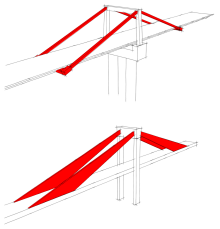
Fortsetzung auf der nächsten Seite ⇒

Tabelle 6: Beschreibung der Klassen. Fortsetzung

№	**Klasse / Beschreibung**	**Abbildung**
31	`GirderMainGirder` → Hauptträger / Längsbalken --- Der Klasse gehören tragende Längsträger, die in der Längsrichtung des Brückendeckquerschnitts verlaufen.	
32	`GirderSecondaryGirder` → Nebenträger / Hilfsträger --- Als Nebenträger kann ein Hilfsbalken definiert werden, der über keine Haupttragfunktion (sondern über eine Hilfstragfunktion) im Brückendeck verfügt. (z. B. Querbalken)	
33	`GirderDeck` → Obere Platte des Brückendeckquerschnitts	
34	`GirderDownPlate` → Untere Platte des Brückendeckquerschnitts	
35	`GirderDiaphragm` → Versteifungsdiaphragma (als plattenförmiges Element)	
36	`GirderBracing` → Aussteifung (als stabförmiges Element)	
37	`TensionElement` → Zugglieder --- Der Klasse gehören tragende Elemente, die hauptsächlich den Zugkräften ausgesetzt sind und hauptsächlich sich im Bereich des Überbaus befinden. *Der Klasse gehört `ConcreteStayTensionElement`, sowie ein Seiltragwerk, das aus `Cable` und `CableAnchoringDevice` besteht.*	

Fortsetzung auf der nächsten Seite ⇒

Tabelle 6: Beschreibung der Klassen. Fortsetzung

№	**Klasse / Beschreibung**	**Abbildung**
38	`ConcreteStayTensionElement` → Zugglieder aus Stahlbeton --- Die Klasse bezeichnet einen Zugglied, der analog einem Seiltragwerk Zugkräfte aufnimmt. Der Zugglied besteht aus Stahlbeton und kann in der Form von einem Balken bzw. einem dünnwandigen Element ausgeführt werden.	
39	`Cable` → Seil	
40	`CableAnchoringDevice` → Seilverankerung	
41	`ArchConstructionElement` → Element einer Bogenkonstruktion --- Der Klasse gehören Elemente, die in einer Bogenkonstruktion über eine Tragfunktion verfügen.	
42	`ArchElement` → Bogenelement --- Die Klasse definiert einen Bogen bzw. einen Teil des Bogens.	
43	`TowerElement` → Pylonelement --- Die Klasse definiert einen Pylon.	
44	`ReinforcingElement` → Bewehrungselement bzw. Spannglied	
45	`ReinforcingBar` → Bewehrungsstab	
46	`ReinforcingMesh` → Bewehrungsmatte	
47	`Tendon` → Spannglied	
48	`TendonAnchoringDevice` → Spanngliedverankerung	

Fortsetzung auf der nächsten Seite ⇒

Tabelle 6: Beschreibung der Klassen. Fortsetzung

№	**Klasse / Beschreibung**	**Abbildung**
	↓ *BridgeJoint* ↓	
49	`Joint` → Fuge	
51	`ExpansionJoint` → Dehnungs- / Bewegungsfuge	
52	`ConstructionJoint` → Arbeitsfuge	
53	`CouplingJoint` → Koppelfuge	

A.2 Beschreibung der ObjectProperties

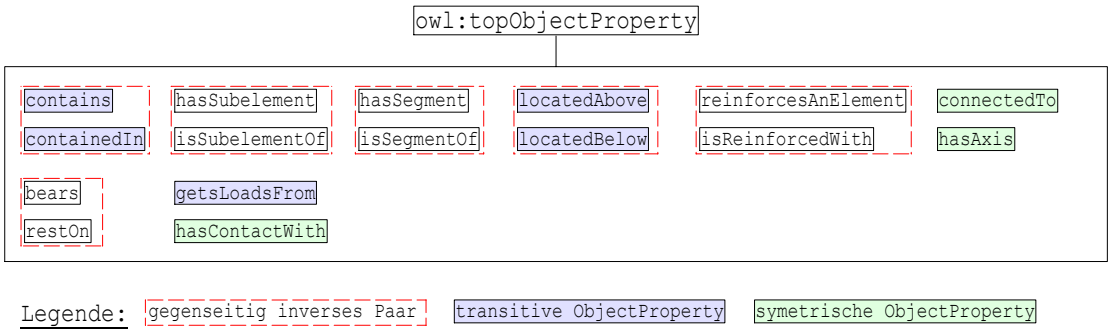


Abbildung 29: Übersicht ObjectProperties

Tabelle 7: Beschreibung der ObjectProperties

№	**ObjectProperty / Beschreibung**

1	**contains**
	→ enthält
	`:contains rdf:type owl:ObjectProperty , owl:TransitiveProperty ;`
	`owl:inverseOf :containedIn .`
	Relation **contains** gewährleistet die Zusammensetzung von physikalischen Elementen und räumlichen Zonen in eine andere übergeordnete räumliche Zone ;
	Zum Beispiel:
	Substructure — **contains** — **Pier**
	Substructure — **contains** — **AbutmentElement**
	Span — **contains** — **GirderSegment**

2	**containedIn**
	→ enthalten in einer räumlichen Zone
	`:containedIn rdf:type owl:ObjectProperty , owl:TransitiveProperty ;`
	`owl:inverseOf :contains ;`
	Relation **containedIn** ist eine inverse **contains** Relation. Sie gewährleistet die Zugehörigkeit eines physikalischen Elementes, einer Komponente bzw. einer räumlichen Zone zu einer anderen übergeordneten räumlichen Zone.

Fortsetzung auf der nächsten Seite ⇒

Tabelle 7: Beschreibung der Relationen. Fortsetzung

№	**Relation (ObjectProperty) / Beschreibung**

3	`hasSubElement`
	→ enthält ein Element
	`:enthält rdf:type owl:ObjectProperty ;`
	`owl:inverseOf :isSubElementOf ;`
	Die Relation `hasSubElement` gewährleistet die Aggregation eines physikalischen Elementes von einer anderen Oberelement.
	Zum Beispiel:
	`AbutmentElement` — `hasSubElement` — `AbutmentHeadWall`
	`AbutmentElement` — `hasSubElement` — `AbutmentWingWall`
	Hinweis: *Die Relation ist nicht transitiv, da die Aggregation in eine Komponente nur einstufig vorgesehen wird.*

4	`isSubElementOf`
	→ gehört zu einer Oberelement
	`:isSubElementOf rdf:type owl:ObjectProperty ;`
	`owl:inverseOf :hasSubElement ;`
	Relation `isSubElementOf` ist eine inverse `hasSubElement` Relation. Sie gewährleistet die Zugehörigkeit eines Elementes einer Komponente.
	Zum Beispiel:
	`GirderDeck` — `isSubElementOf` — `GirderElement`

5	`hasAxis`
	→ befindet sich in der Achse
	`:hasAxis rdf:type owl:ObjectProperty, owl:SymmetricProperty ;`
	`owl:propertyChainAxiom ( :isSubElementOf :hasAxis ) ;`
	Relation `hasAxis` ermöglicht die Anordnung eines Elementes zu einer Achse und umgekehrt.
	Zum Beispiel:
	`Column` — `hasAxis` — `Axis`
	`Axis` — `hasAxis` — `Column`

Fortsetzung auf der nächsten Seite ⇒

Tabelle 7: Beschreibung der Relationen. Fortsetzung

№	**Relation (ObjectProperty) / Beschreibung**

6	`connectedTo`
	→ angeschlossen zu . . .
	`:connectedTo rdf:type owl:ObjectProperty ;`
	`:connectedTo rdf:type owl:ObjectProperty , owl:SymmetricProperty ;`
	Die Relation `connectedTo` gewährleistet den Anschluss eines Elementes zu einem anderem, wobei die Lastübertragung durch die Verbindung vorgesehen wird.
	Zum Beispiel:
	`Cable` — `connectedTo` — `Tower`

7	`bears`
	→ trägt
	`:bears rdf:type owl:ObjectProperty ;`
	`rdfs:subPropertyOf :getsLoadsFrom , :hasContactWith , :locatedBelow ;`
	`owl:inverseOf :restOn .`
	Relation `bears` gibt an, dass ein Element ein anderes Element trägt, und als Folge einige Last aufnimmt.
	Zum Beispiel:
	`Pile` — `bears` — `Footing`

8	`restOn`
	→ liegt auf
	`:restOn rdf:type owl:ObjectProperty ;`
	`owl:inverseOf :bears ;`
	Die Relation `restOn` gibt an, dass ein Element A sich auf ein anderes Element B stützt und das Element A leitet die Lasten zu dem unteren Element B weiter.
	Zum Beispiel:
	`Footing` — `restOn` — `Pile`

Fortsetzung auf der nächsten Seite ⇒

Tabelle 7: Beschreibung der Relationen. Fortsetzung

№	**Relation (ObjectProperty) / Beschreibung**

9	`hasContactWith`
	→ hat einen Kontakt mit ...
	→ *automatisch geschlussfolgerte Relation! Manuelle Vergabe ist nicht vorgesehen!*
	`:hasContactWith rdf:type owl:ObjectProperty , owl:SymmetricProperty ;`
	Die Relation `hasContactWith` gibt an, dass zwischen zwei Elementen ein physikalischer Kontakt vorhanden ist.
	Zum Beispiel:
	`GirderElement` — `hasContactWith` — `PotBearings`

10	`locatedBelow`
	→ befindet sich unter ...
	→ *automatisch geschlussfolgerte Relation! Manuelle Vergabe ist nicht vorgesehen!*
	`:locatedBelow rdf:type owl:ObjectProperty , owl:TransitiveProperty ;`
	`owl:inverseOf :locatedAbove ;`
	Die Relation `locatedBelow` stellt dar, dass ein Element sich unter einem anderen befindet. Die Elemente sind nicht gezwungen einen physikalischen bzw. logischen Kontakt zu haben. Die Relation zeigt bloß das räumliche Verhalten.
	Zum Beispiel:
	`Pile` — `locatedBelow` — `GirderComponent`

11	`locatedAbove`
	→ befindet sich über ...
	→ *automatisch geschlussfolgerte Relation! Manuelle Vergabe ist nicht vorgesehen!*
	`:locatedAbove rdf:type owl:ObjectProperty , owl:TransitiveProperty ;`
	`owl:inverseOf :locatedBelow ;`
	Die Relation `locatedAbove` stellt dar, dass ein Element sich über einem anderen befindet. Die Elemente sind nicht gezwungen einen physikalischen bzw. logischen Kontakt zu haben. Die Relation zeigt bloß das räumliche Verhalten.
	Zum Beispiel:
	`GirderComponent` — `locatedAbove` — `Pile`

Fortsetzung auf der nächsten Seite ⇒

Tabelle 7: Beschreibung der Relationen. Fortsetzung

№	**Relation (ObjectProperty) / Beschreibung**

12	`getsLoadsFrom`
	→ nimmt Lasten von ... auf
	→ *automatisch geschlussfolgerte Relation! Manuelle Vergabe ist nicht vorgesehen!*
	`:getsLoadsFrom rdf:type owl:ObjectProperty ,`
	`owl:TransitiveProperty ;`
	Die Relation `getsLoadsFrom` zeigt von welchen oberhalb liegenden Elementen bzw. Komponenten wird ein Element belastet.
	Zum Beispiel:
	`Pile` — `getsLoadsFrom` : `Footing`, `TowerComponent`

13	`isReinforcedWith`
	→ ist bewehrt mit
	`:isReinforcedWith rdf:type owl:ObjectProperty ;`
	`owl:inverseOf :reinforcesAnElement ;`
	Die Relation `isReinforcedWith` gewährleistet die Bewehrung eines Bauteils.
	Zum Beispiel:
	`AbutmentElement` — `isReinforcedWith` — `ReinforcingBar`
	Hinweis: *Die Relation ist nicht transitiv, da die Aggregation in eine Komponente nur einstufig vorgesehen wird.*

14	`reinforcesAnElement`
	→ bewehrt ein Element
	`:reinforcesAnElement rdf:type owl:ObjectProperty ;`
	`owl:inverseOf :isReinforcedWith ;`
	Relation `reinforcesAnElement` ist eine inverse `isReinforcedWith` Relation.
	Zum Beispiel:
	`ReinforcingBar` — `reinforcesAnElement` — `AbutmentElement`

Fortsetzung auf der nächsten Seite ⇒

Tabelle 7: Beschreibung der Relationen. Fortsetzung

№	**Relation (ObjectProperty) / Beschreibung**

15	`hasSegment`
	→ hat ein Segment
	--- 	
	`:hasSegment rdf:type owl:ObjectProperty ;`
	`owl:inverseOf :isSubElementOf ;`
	--- 	
	Die Relation `hasSegment` gewährleistet die Aggregation einer `Segment`–Zone von der anderen größeren Zone.
	Zum Beispiel:
	`ArbitraryZone` — `hasSegment` — `PierZone`

16	`isSegmentOf`
	→ ist ein Segment von
	--- 	
	`:isSegmentOf rdf:type owl:ObjectProperty ;`
	`owl:inverseOf :hasSegment ;`
	--- 	
	Relation `isSegmentOf` ist eine inverse `hasSegment` Relation.

A.3 Beschreibung der DatatypeProperties

Tabelle 8: Beschreibung der DatatypeProperties

№	**DatatypeProperty / Beschreibung**
1	`struct_location` → Lage der Brücke Salzwasser; Salzhaftige_Luft; Festland; Benutzerdefiniert
2	`struct_bearingsConceptOfSuperstructure` → Tragprinzip des überbaus Balkenbrücke; Rahmenbrücke; Hängebrücke; Schrägseilbrücke; Bogenbrücke; Fachwerkbrücke; Benutzerdefiniert
3	`struct_bridgeFunctionType` → Brückentyp nach der Funktion Fussgänger; Eisenbahn; Autoverkehr; Kanal; Benutzerdefiniert
4	`struct_extremeEnvironmentalLoads` → Aussergewöhnliche Lasten Erdbeben; Tsunami; Keine;
5	`struct_loadCase` → Lastmodell
6	`struct_numberSpans` → Anzahl von Brückenfeldern
7	`struct_preservationOrder` → Denkmalschutz Ja; Nein; Nicht_Definiert
8	`struct_trafficType` → Typ des Verkehrs Fussgänger ; Schwerlast; Autobahn; Güter; Eisenbahn; Gemischt; Militär; Benutzerdefiniert
9	`struct_yearOfConstruction` → Baujahr der Brücke
10	`doc_appropriateRegulations` → Zugehörige Vorschriften

Fortsetzung auf der nächsten Seite ⇒

Tabelle 8: Beschreibung der DatatypeProperties. Fortsetzung

№	**DatatypeProperty / Beschreibung**
11	doc_dataCollectionCompleted → Datenerfassung ist abgeschlossen Abgeschlossen; Nicht_Abgeschlossen
12	doc_dataCollectionDate → Datumsangabe zur Datenerfassung
13	doc_documentationAvailability → Verfügbarkeit von Unterlagen (Statik, Pläne, Details usw.) Ja; Nein; Teilweise
14	doc_documentationRegister → Liste von verfügbaren Unterlagen
15	elem_ageOfElement → Alter des Bauteils
16	elem_bearingsCriticalTemperatureSummer → Kritische Temperatur für das Lager im Winter
17	elem_bearingsCriticalTemperatureWinter → Kritische Temperatur für das Lager im Sommer
18	elem_bearingsDegreeOfFreedomDisplacement → Freiheitsgrade des Lagers (Verschiebung) X_Fest; Y_Fest; Z_Fest; X_Y_Fest; Z_X_Fest; Z_Y_Fest; X_Y_Z_Fest;
19	elem_bearingsDegreeOfFreedomTorsion → Freiheitsgrade des Lagers (Torsion) Um_X_Fest; Um_Y_Fest; Um_Z_Fest; Um_X_Y_Fest; Um_X_Z_Fest; Um_Y_Z_Fest; Um_X_Y_Z_Fest
20	elem_bearingsReinforcedType → Typ des Lagers Bewehrt; Unbewehrt
21	elem_bridgeGirderTensionPresence → Vorhandensein der Vor- bzw. Nachspannung des Zuggliedes Vorspannung; Nachspannung

Fortsetzung auf der nächsten Seite ⇒

Tabelle 8: Beschreibung der DatatypeProperties. Fortsetzung

№	**DatatypeProperty / Beschreibung**
22	elem_bridgeGirderTensionValü → Wert der Vor- bzw. Nachspannung des Zuggliedes
23	elem_concreteCover → Betondeckung für das ganze Element
24	elem_concreteCoverAllSides → Betondeckung allseitig
25	elem_concreteCoverBottom → Betondeckung unten
26	elem_concreteCoverTop → Betondeckung oben
27	elem_elementDescription → Elementbezeichnung
28	elem_elementExistingOrNew → Element gehört zum Neubau oder Bestand Bestand; Neu
29	elem_expansionJointMovementRange → Bewegungsbereich der Fuge GROSS - Bewegungsfuge Für Den GROSSEN Bewegungsbereich (Mehr Als 10cm); MITTEL - Bewegungsfuge Für Den MITTLEREN Bewegungsbereich (5-10cm); KLEIN - Bewegungsfuge Für Den KLEINEN Bewegungsbereich (1-5cm)
30	elem_manufacturer → Hersteller
31	elem_memberType → Typ des Elements Zugelement; Druckelement; Element_Nichtdefiniert
32	elem_precastElement → Fertigbauteil Ja; Nein; Teilweise

Fortsetzung auf der nächsten Seite ⇒

Tabelle 8: Beschreibung der DatatypeProperties. Fortsetzung

№	**DatatypeProperty / Beschreibung**
33	elem_reinforcementLayerOrder → Reihenfolge der Bewehrungslage 1. 2. 3. 4. 5. Usw.
34	elem_reinforcementLayerPosition → Position der Bewehrungslage Horizontal; Vertikal; Schräg
35	elem_reinforcementPercentage → Bewehrungsgrad des Bauteils
36	elem_reinforcementPrestressingLevel → Vorspanngrad der Bewehrung
37	elem_reinforcementTask → Aufgabe der Bewehrung Statisch_Erforderlich; Konstruktiv
38	elem_tendonDirectionType → Typ des Spanglieds bezüglich der Ausrichtung im Brückendeck Längs; Quer
39	elem_tendonTensionType → Spannungstyp des Zugglieders Vorspannung; Nachspannung
40	elem_tensionForce → Spannkraft
41	elem_tensioningTechnology → Spannverfahren
42	elem_yearOfInstallation → Einbaujahr
43	mat_aggregateGrainSize → Korngrösseverteilung von Gesteinskörnungen für Beton Grob; Fein; Zsmmngstzt_Gesteinskörnung - (natürlich zusammengesetzte Gesteinskörnung 0/8); Korngemisch

Fortsetzung auf der nächsten Seite ⇒

Tabelle 8: Beschreibung der DatatypeProperties. Fortsetzung

№	**DatatypeProperty / Beschreibung**
44	mat_cementContent → Zementgehalt
45	mat_cementDesignation → Zementbezeichnung
46	mat_cementMainConstiuents → Hauptbestandteile vom Zement (Klassifizierung erfolgt gemaess DIN EN 197-1.) K - Portlandzementklinker S - Huettensand (granulierte Hochofenschlacke) P - Natuerliches Puzzolan Q - Natuerliches getempertes Puzzolan V - Kieselsaeurereiche Flugasche W - Kalkreiche Flugasche T - Gebrannter Schiefer LL - Kalkstein (darf einen Massenanteil von 0,20 % nicht ueberschreiten) L - Kalkstein (darf einen Massenanteil von 0,50 % nicht ueberschreiten) D - Silicastaub
47	mat_cementType → Zementart CEM_I - Portlandzemente; CEM_II - Portlandkompositzemente; CEM_III - Hochofenzemente; CEM_IV - Puzzolanzemente; CEM_V - Kompositzemente;
48	mat_coatDesignation → Beschichtungsbezeichnung
49	mat_concreteAdmixture → Position der Bewehrungslage Flugasche; Silikastaub; Hüttensand; Trass; Fasern
50	mat_concreteCompressionStrengthClass →Druckfestigkeitsklasse des Betons C8/10; C12/15; C16/20; C20/25; C25/30; C30/37; C35/45; C40/50; C45/55; C50/60; C55/67; C60/75; C70/85; C80/95; C90/105; C100/115
51	mat_concreteStrengthCube → Würfeldruckfestigkeit vom Beton
52	mat_concreteStrengthCylindrical → Zylinderdruckfestigkeit vom Beton

Fortsetzung auf der nächsten Seite ⇒

Tabelle 8: Beschreibung der DatatypeProperties. Fortsetzung

№	**DatatypeProperty / Beschreibung**
53	mat_expositionClassConcreteX0 → Expositionsklasse für Beton X0 - kein Korrosions- oder Angriffsrisiko (Klassifikation gemäss DIN 1045-2); X0
54	mat_expositionClassConcreteXA → Expositionsklasse für Betonkorrosion durch chemischen Angriff (Klassifikation gemäss DIN 1045-2); XA1; XA2; XA3
55	mat_expositionClassConcreteXF → Expositionsklasse für Betonkorrosion durch Frostangriff (Klassifikation gemäss DIN 1045-2); XA1; XA2; XA3
56	mat_expositionClassConcreteXM → Expositionsklasse für Betonkorrosion durch Verschleissbeanspruchung (Klassifikation gemäss DIN 1045-2); XA1; XA2; XA3
57	mat_expositionClassReinforcementXC → Expositionsklasse für Bewehrungskorrosion durch Karbonatisierung (Klassifikation gemäss DIN 1045-2); XA1; XA2; XA3
58	mat_expositionClassReinforcementXD → Expositionsklasse für Bewehrungskorrosion durch Chloride (ausser Meerwasser) (Klassifikation gemäss DIN 1045-2); XA1; XA2; XA3
59	mat_expositionClassReinforcementXS → Expositionsklasse für Bewehrungskorrosion durch Chloride aus Meerwasser (Klassifikation gemäss DIN 1045-2); XA1; XA2; XA3

Fortsetzung auf der nächsten Seite ⇒

Tabelle 8: Beschreibung der DatatypeProperties. Fortsetzung

№	**DatatypeProperty / Beschreibung**
60	`mat_moistureClassConcreteW` → Feuchtigkeitsklasse für Betonkorrosion WO; WF; WA; WS
61	`mat_monitoringClassConcrete` → überwachungklasse für Beton 1; 2; 3
62	`mat_reinforcementGrade` → Bewehrungsklasse B500A; B500B; B500C
63	`mat_reinforcementYieldStrength` → Streckgrenze des Bewehrungsstahls
64	`mat_tendonSteelBreakingStrength` → Bruchgrenze des Spanngliedsstahles
65	`mat_tendonSteelBreakingStrength` → Bruchgrenze des Spanngliedsstahles