# Ontologie-gestützte Optimierung des Entwurfs automobilelektronischer Systeme

**Dissertation**
der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
Dipl.-Inform. Jan Novacek
aus Heidelberg

Tübingen
2023

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

| | |
|---|---|
| Tag der mündlichen Qualifikation: | 16.10.2023 |
| Dekan: | Prof. Dr. Thilo Stehle |
| 1. Berichterstatter: | Prof. Dr. Oliver Bringmann |
| 2. Berichterstatter: | Prof. Dr. Anne Koziolek |

# Kurzfassung

Die zu beherrschende Komplexität bei der Entwicklung automobilelektronischer Systeme unterliegt einem stetigen Wachstum und ist nicht zuletzt aus diesem Grund mit mehreren ingenieurtechnischen Herausforderungen verbunden. Etablierte Ansätze wie die des Systems Engineering bieten Möglichkeiten, solch komplexe Systeme zu entwerfen und schließlich zu realisieren. Vordringliches Problem in diesem Zusammenhang ist jedoch, dass Engineering-Daten primär in über unterschiedliche Arbeitsplatzsysteme verstreuten Dokumenten abgelegt sind und dass diese nur unzureichend verwaltet werden. Einen Ausweg aus dieser Misere stellt die Abbildung dieser Daten auf Modelle dar. So gilt das Modell-basierte Systems Engineering derweil in der Automobil- und Luftfahrtindustrie als akzeptierter Weg, komplexe Systeme zu realisieren, auch wenn nicht alle Disziplinen dabei kontinuierlich gekoppelt sind. Nicht nur für diese Kopplung, sondern auch für das automatische Schlussfolgern benötigen Modelle zusätzliche, explizite Semantik. Automatisches Schlussfolgern ist beispielsweise für die Identifikation von Korrelationen zwischen Systems Engineering-Daten erforderlich. Für die Schaffung semantischer Interoperabilität eignen sich insbesondere Ontologien, die selbst auch Modelle sind.

In dieser Arbeit wird ein Ontologie-basierter Ansatz zur Optimierung des Entwurfsprozesses von automobilelektronischen Systemen vorgestellt. Wesentlicher Grundgedanke dabei ist es, Ontologien zu nutzen, um Entwurfsmethoden und -modelle zu konsolidieren und zu integrieren. Dazu beruht der Ansatz im Kern auf dem Vorschlag einer einheitlichen Basis zur Entwicklung und Ausführung von Anwendungen, unter konsequenter Nutzung etablierter Standards, um Modelle auf Ontologien abzubilden. Diese Basis wurde als Softwareplattform realisiert, welche unter anderem auf eine nahtlose Integration in existierende Arbeitsabläufe abzielt. Vorrangiger Aspekt des Lösungsansatzes ist die Berücksichtigung von Anforderungen, sowie spezieller Last- und Nutzungsprofilen in Form von sogenannten Mission Profiles und deren Integration in Entwicklungsprozesse. Ergänzt wird diese Arbeit zudem durch die Beschreibung und Einordnung dreier Anwendungen, welche auf der Plattform aufsetzen und zur Untersuchung und Bewertung in konkreten Fallbeispielen Gebrauch finden. Die damit entstandenen Entwurfsmethodiken adressieren jeweils spezielle Problemstellungen aus dem Umfeld der Entwicklung automobilelektronischer Systeme und demonstrieren zudem die Anwendbarkeit der vorgestellten Entwurfs- und Anwendungsplattform.

# Danksagung

Diese Arbeit wurde durch meine Teilnahme am Promotionskolleg Entwurf und Architektur eingebetteter Systeme (EAES) ermöglicht und entstand in Verbindung mit meiner Tätigkeit als wissenschaftlicher Mitarbeiter in der Abteilung Systementwurf in der Mikroelektronik (SiM) am FZI Forschungszentrum Informatik.

Mein besonderer Dank gilt Herrn Prof. Dr. Wolfgang Rosenstiel und Herrn Prof. Dr. Oliver Bringmann für die Betreuung und Unterstützung durch alle Abschnitte dieser Arbeit.

Weiter möchte ich meinen Kolleginnen und Kollegen am FZI, sowie aus dem Promotionskolleg EAES und den Studierenden danken, die an dieser Arbeit mitgewirkt haben. Herrn Dr. Alexander Viehl danke ich zutiefst für die vielen wertvollen inhaltlichen und fachlichen Gespräche, und die fürsorgliche Betreuung, ohne die diese Arbeit nicht möglich gewesen wäre. So gilt mein Dank ebenfalls Herrn Dr. Sebastian Reiter für seine Unterstützung, vor allem in der letzten Phase dieser Arbeit.

Zu guter Letzt danke ich meiner Familie inständig für die bedingungslose Unterstützung, für die große Geduld und den konsequenten, unerschütterlichen moralischen Beistand.

# Inhaltsverzeichnis

# 1 Optimierung des Entwurfs automobilelektronischer Systeme

Fahrerassistenzsysteme (FAS, im Englischen: Advanced Driver Assistance Systems, ADAS) und weitere Funktionen werden zunehmend in modernen Fahrzeugen verbaut [6]. Dies erhöht den Komfort und die Sicherheit beim Fahren. Vor dem Gesichtspunkt, dass 97 % aller Autounfälle auf menschliche Fehler zurückzuführen sind [7], konnte auch gezeigt werden, dass moderne FAS das Schaden- genauer gesagt Unfallgeschehen potenziell positiv beeinflussen können [8]. Neben FAS und dem hochautomatisierten Fahren zeichnen sich weitere globale Trends wie die Elektrifizierung, die gemeinsame Fahrzeugnutzung und die Vernetzungsfähigkeit ab [9]. Der Einzug der dazu erforderlichen Automobilelektronik ist mit einer stark wachsenden Komplexität der diese Funktionen realisierenden Systeme verbunden. Angesichts des hohen Wettbewerbs im Automobilmarkt herrscht zudem ein immenser Zeit- und Kostendruck in der Automobilindustrie. So sind Qualität, Funktionalität und Zuverlässigkeit von Automobilen entscheidende Faktoren globaler Wettbewerbsfähigkeit geworden [10; 11]. Die Gewährleistung der Qualität erfordert, angesichts der zugleich rauen Umgebungsbedingungen für die automobilelektronischen Systeme, zudem neue Herangehensweisen in der Entwicklung dieser Systeme. Diese müssen die vielfältigen Forderungen verschiedener etablierter Standards wie der ISO 26262 berücksichtigen. Aus diesem Spannungsfeld heraus, werden in dieser Dissertation wissensbasierte Lösungsmöglichkeiten vorgestellt, um den steigenden Anforderungen gerecht werden zu können und einigen Problemstellungen zu entgegnen, wobei am Entwurfsprozess angesetzt wird.

## Abschnitte

## 1.1 Motivation und Zielsetzung

Die domänenübergreifende Entwicklung komplexer Systeme, wie von Automobilelektronik, ist mit mehreren ingenieurtechnischen Herausforderungen verbunden. Ein vorherrschendes Problem ist, dass Systems Engineering (SE) Daten primär in Dokumenten abgelegt sind, welche über Arbeitsplatzsysteme verteilt sind und zudem unzureichend verwaltet werden [12]. Weiterhin fehlt in Entwurfsmodellen oft die für das automatische Schlussfolgern erforderliche explizite Semantik. Das automatische Schlussfolgern ist unter anderem notwendig, um Korrelationen zwischen SE-Daten identifizieren zu können.

Durch eine Darstellung von Systemen, deren Entwürfen sowie weiteren Entwurfsartefakten und sonstiger für die Entwicklung erforderlicher Informationen und Zusammenhänge in Modellen ergeben sich verschiedene Vorteile, mit denen dieser Problematik entgegnet werden kann. So wird die Nutzung von Modell-based Systems Engineering (MBSE) Methoden als ein effektiver Weg in der Automobil- und Luftfahrtindustrie akzeptiert, komplexe Systeme zu entwickeln [12; 13].

Nutzung von und Kommunikation zwischen heterogenen Systemen, welche sich durch die zuvor angeführte Verteilung der SE-Daten über die Entwicklungssysteme hinweg ergeben, kann durch den Einsatz von semantischen Technologien verbessert werden. Insbesondere Ontologien eignen sich, neben der Spezifikation expliziter Semantik, eine Konsolidierung und Integration von Entwurfsmethoden und -modellen unterschiedlicher Disziplinen zur Herstellung semantischer Interoperabilität. Dies kann beispielsweise durch deren Funktion als gemeinsames Vokabular bei der Integration heterogener Systeme geschehen, oder beim Zusammenführen unterschiedlicher Wissensquellen zu deren gemeinsamer Einbindung in Prozesse. Die Schaffung effektiver Wissensmanagementsysteme ist ein Schlüsselfaktor in der Prozessoptimierung des Engineerings [14]. Da jede Ontologie auch ein Modell darstellt, bietet sich deren Einsatz außerdem im Zusammenhang mit MBSE an.

Aus diesen Gründen wird in dieser Forschungsarbeit untersucht, wie sich Schlussfolgerungen aus Entwurfsmodellen ziehen lassen. Da hierfür eine explizite Semantik notwendig ist, geschieht dies unter Einsatz von Ontologien, die sich für deren Spezifikation eignen. Die zentrale Forschungsfrage lautet daher:

> *Lassen sich Schlussfolgerungen, unter Nutzung von Ontologien, auf der Basis von Entwurfsmodellen ziehen, zur Verbesserung des Entwurfsprozesses automobilelektronischer Systeme?*

Weiterhin spielen Umgebungs- und Lastprofile, sogenannte Mission Profiles (MPs) eine wichtige Rolle bei der Propagierung von Anforderungen entlang der Prozesskette der Entwicklung [15–17]. Angesichts dessen sind MPs auch Gegenstand der

Betrachtungen in dieser Arbeit und finden auch im Lösungsansatz ihre Berücksichtigung. Mit Bezug auf die zuvor aufgeworfene Forschungsfrage wurden zu deren Beantwortung folgende Ziele definiert:

1. Entwicklung exemplarischer Methoden sowie diese begleitenden Werkzeuge, welche Schlussfolgerungen auf der Basis von Entwurfsmodellen ziehen und zur Verbesserung des Entwurfsprozesses automobilelektronischer Systeme beitragen

2. Identifikation von Vor- und Nachteilen, die sich durch den Einsatz von Ontologien zum automatischen Schlussfolgern auf der Basis von Entwurfsmodellen im Umfeld des Automobilelektronikentwurfs ergeben

## 1.2 Beitrag

Wesentlicher Beitrag ist der Vorschlag einer einheitlichen Basis für Anwendungen, die MPs berücksichtigen. Diese Berücksichtigung ermöglicht Mission Profile Aware Design (MPAD), was im Grundlagen-Kapitel in Abschnitt 2.2.2.1 näher erläutert wird. Neben der *semantischen Mission Profile Aware Design Plattform* werden zudem drei, die Plattform nutzende Anwendungen vorgestellt. Insgesamt setzt sich der Beitrag dieser Arbeit wie folgt zusammen:

1. Einheitliche Grundlage für die Entwicklung und Ausführung von MPAD Anwendungen,

2. Erweiterbarkeit des vorgeschlagenen Systems um weitere, neue Ansätze, die zur Verbesserung des Entwurfsprozesses automobilelektronischer Systeme beitragen,

3. Abstraktion über MPs und Anforderungen, sowie Wissensbasen, Entwicklungswerkzeuge und Entwurfssysteme,

4. Methode zur Unterstützung bei der Analyse und Sicherstellung der Robustheit von automobilelektronischen Systemen

5. Methode zur Verwaltung und Korrelation von Entwurfsparametern und Anforderungen zur Untersuchung der Auswirkungen von Anforderungsänderungen

6. Methode zur Transformation von Anforderungen unter Berücksichtigung der Anwendbarkeit von Fehlermodellen

Abbildung 1.1 illustriert den Zusammenhang zwischen der Plattform, peripheren Komponenten beziehungsweise Systemen, sowie den Anwendungen. Zu den peripheren Komponenten beziehungsweise Systemen zählen die MPs, eine Wissensbasis,

Anforderungskataloge sowie Entwurfswerkzeuge und -systeme. Die Plattform abstrahiert über die Peripherie, sodass den auf der Plattform aufsetzenden Anwendungen eine einheitliche Schnittstelle zur Verfügung steht. MPs können dabei sowohl als Eingabe genutzt werden, als auch ein Ausgabe-Ergebnis darstellen. Dasselbe gilt auch für Anforderungen. Aus der Wissensbasis werden Fakten abgerufen, welche von den Anwendungen, die auf der Plattform aufsetzen, benötigt werden. Ebenso können Ergebnisse von Ausführungen der Anwendungen in der Wissensbasis abgelegt werden. Die Plattform ermöglicht weiterhin eine Anbindung externer Entwurfswerkzeuge und -systeme, welche von den Anwendungen genutzt und diesen auch Ergebnisse von Ausführungen bereitstellen können.

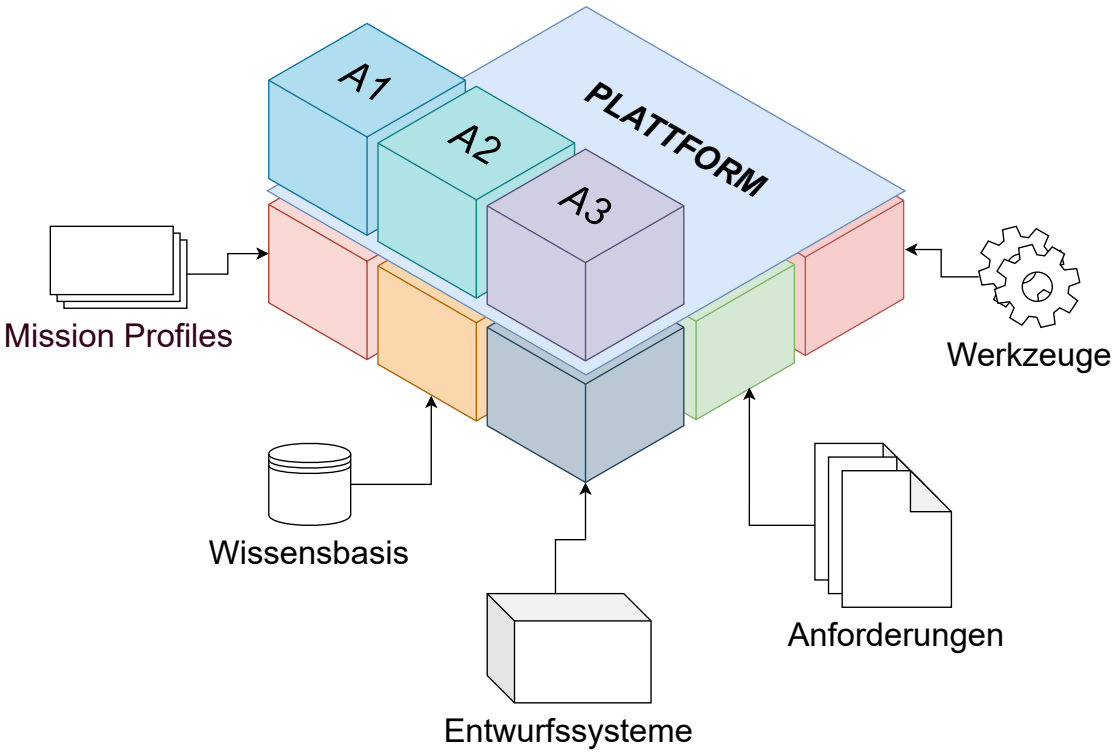



**Abbildung 1.1** – Illustration des Zusammenhangs zwischen der Semantischen Mission Profile Aware Design Plattform, peripheren Komponenten beziehungsweise Systemen, sowie Anwendungen

## 1.3 Gliederung der Arbeit

Diese Dissertation ist in insgesamt zehn Kapitel unterteilt. Einem einleitenden Kapitel, welches einen allgemeinen Überblick über diese Arbeit gibt, folgen relevante

Grundlagen in Kapitel 2. Die Grundlagen sind dabei in die Bereiche „Semantic Web Technologies“ (SWT) und „Design for Automotive“ (DfA) aufgeteilt und führen jeweils in Themen ein, die mit der vorliegenden Arbeit im Zusammenhang stehen. Den Grundlagen schließt sich Kapitel 3 an, in welchem der Stand der Technik diskutiert wird. Dieser beinhaltet einen Überblick über die generelle Anwendung von Ontologien im Engineering, sowie eine Aufstellung relevanter Arbeiten – sowohl in Bezug auf verwandte Dissertationen, als auch auf weiteren Arbeiten. Nach diesen Voraussetzungen folgt Kapitel 4 mit einer Beschreibung, der im Rahmen des gewählten Lösungsansatzes entwickelten Plattform sowie dem übergeordneten methodischen Rahmenwerk in Hinsicht auf die Entwicklung mit der Plattform. Daraufhin werden in den Kapiteln 5, 6 und 7 jeweils weitere, eigenständige Arbeiten vorgestellt, welche die in Kapitel 4 vorgestellte Plattform nutzen und somit als deren Anwendungen betrachtet werden können. In Kapitel 8 werden einige Details zur Implementierung der Plattform beschrieben, gefolgt von Kapitel 9, in welchem die Evaluation behandelt wird, zusammen mit Fallbeispielen, die insgesamt im selben Kapitel zur Bewertung des Lösungsansatzes herangezogen werden und diskutiert werden. Die Arbeit schließt letztlich mit einer Zusammenfassung und einem Ausblick in Kapitel 10 ab. Weitere feingranulare Gliederungen werden in dieser Arbeit jeweils zu Beginn eines Kapitels, genauer gesagt eines übergeordneten Abschnitts beschrieben.

# 2 Grundlagen

In diesem Kapitel werden für das Verständnis dieser Arbeit relevante Grundlagen angeführt. Dies geschieht durch kurzgehaltene Einführungen in die entsprechenden Themenbereiche sowie zusätzlichen Verweisen auf weiterführende Literatur. Das Kapitel ist in zwei Grundlagenabschnitte unterteilt: Im ersten Abschnitt 2.1 werden zunächst relevante Grundlagen von Semantic Web Technologies (SWT) angeführt. Diese umfassen eine allgemeine Einführung in SWT in Abschnitt 2.1, gefolgt von einer Beschreibung der Semantic Web Architektur in Abschnitt 2.1.1. Weitere relevante Grundlagen zu SWT umfassen in dieser Arbeit eingesetzte Datenformate und -Modelle für den Informationsaustausch im Semantic Web, in Abschnitt 2.1.2, darauf aufbauend eingesetzte Sprachen zur Wissensrepräsentation und -abfrage in Abschnitt 2.1.3, eine grundlegende Einführung in die automatische Inferenz neuen Wissens in Abschnitt 2.1.4 und schließlich einführende Informationen zum Linked Data Konzept in Abschnitt 2.1.5. Die Beschreibungen der Datenformate oder -modelle, sowie die der Sprachen zur Wissensrepräsentation/-abfrage bauen jeweils aufeinander auf.

Im zweiten Grundlagenabschnitt 2.2 werden verschiedene relevante Grundlagen des Entwurfs für Automobile angeführt. Zu diesen zählen sowohl relevante Entwurfsmethoden in den Abschnitten 2.2.2.1 und 2.2.1, sowie ein Überblick über die Verwendung von Modellen in der Entwicklung in Abschnitt 2.2.4 und der Analyse der Auswirkungen von Änderungen in Abschnitt 2.2.5.

Während die Informationen zu SWT sowohl für die in Kapitel 4 vorgestellte Plattform, als auch für die in den Kapiteln 5, 6 und 7 beschriebenen Anwendungen relevant sind, beziehen sich die Abschnitte 2.2.1 und 2.2.5 auf die in den Kapiteln 5 und 6 beschriebenen Anwendungen. Schließlich bezieht sich der Unterabschnitt 2.2.4.1, in welchem Grundlagen der Modelltransformation angeführt werden, auf die Anwendung, welche in Kapitel 7 beschrieben wird.

## Abschnitte

## 2.1 Semantic Web Technologien

Das World Wide Web (WWW) entstand 1990 am Kernforschungszentrum CERN. Es ermöglicht einen einfachen Informationszugriff ohne Expertenwissen und die Informationsrecherche durch Suchmaschinen. Die Informationen im WWW werden über Dokumente dargestellt, die sogenannte *Hyperlinks* enthalten können. Dadurch wird eine Verknüpfung von Inhalten erreicht, was das Auffinden relevanter, weiterführender Informationen ermöglicht.

Das WWW hat allerdings gewisse Grenzen. Ein wesentliches Problem ist das Auffinden der richtigen Informationen, sowie die Sicherstellung der Korrektheit der Informationen im WWW. Weiterhin sind die Informationsextraktion und die Personalisierung im WWW schwierig. Grundsätzlich ist das WWW für die Benutzung durch Menschen entworfen. Die Inhalte der Dokumente im WWW sind für Maschinen nicht oder nur aufwendig zu verarbeiten. So ist etwa die Auszeichnungssprache Hypertext Markup Language (HTML) primär für die Beschreibung, der *Darstellung* und *Verknüpfung* von Informationen ausgelegt. Ein Hyperlink in HTML verweist zwar auf einen Inhalt im WWW, enthält jedoch keine Informationen zur Bedeutung der Verknüpfung. Auch abgesehen von den Hyperlinks beschreibt HTML nicht, was die dargestellten Informationen *bedeuten*. Das heißt HTML beschreibt nicht die *Semantik* der Inhalte und Verknüpfungen. Unter *Semantik* wird in diesem Zusammenhang die Bedeutung von Wörtern, sowie deren Beziehungen untereinander verstanden.

Der Lösungsansatz des *Semantic Web* sieht die formelle Beschreibung der Inhalte und Verknüpfungen vor. Inhalte sollen demnach mit *Metadaten* versehen werden, welche diese beschreiben, deren Auffinden vereinfachen und die weitere Verarbeitung durch Maschinen ermöglichen. In diesem Rahmen kommen verschiedene Technologien zum Einsatz, die unter dem Begriff Semantic Web Technologies (SWT) zusammengefasst werden. Ausgewählte, relevante SWT und deren Zusammenhänge sollen in diesem Grundlagenabschnitt erläutert werden.

### 2.1.1 Semantic Web Architektur

Es gibt verschiedene Vorschläge für die Architektur des Semantic Web [18–20]. Abbildung 2.1 zeigt eine populäre Darstellung des Schichtenmodells der SWT. Nach der *Semantic Web Roadmap* [21] gibt es vier Stufen, die zur Verwirklichung des Semantic Web zu erreichen sind. Die erste Stufe stellen teilweise strukturierte oder vollständig unstrukturierte Textdokumente und Datenbankeinträge dar, die Daten enthalten. Auf dieser Stufe sind die Daten bisher nicht in einer Form, die es ermöglichen würde, diese zwischen heterogenen Systemen auszutauschen. An-

ders auf der zweiten Stufe, nach der Daten mittels der in Abschnitt 2.1.2.1 näher beschriebenen Extensible Markup Language (XML) formatiert werden sollen. Die dritte Stufe sieht vor, dass Daten nach dem in Abschnitt 2.1.2.2 beschriebenen Resource Description Framework (RDF) strukturiert und verknüpft werden. Letztlich sollen in der vierten Stufe, die in Abschnitt 2.1.3.3 beschriebenen Web Ontology Language (OWL) Ontologien zum Einsatz kommen, um anhand von vorhandenem Wissen über automatisches Schlussfolgern, wie in Abschnitt 2.1.4 beschrieben, neues Wissen abzuleiten.

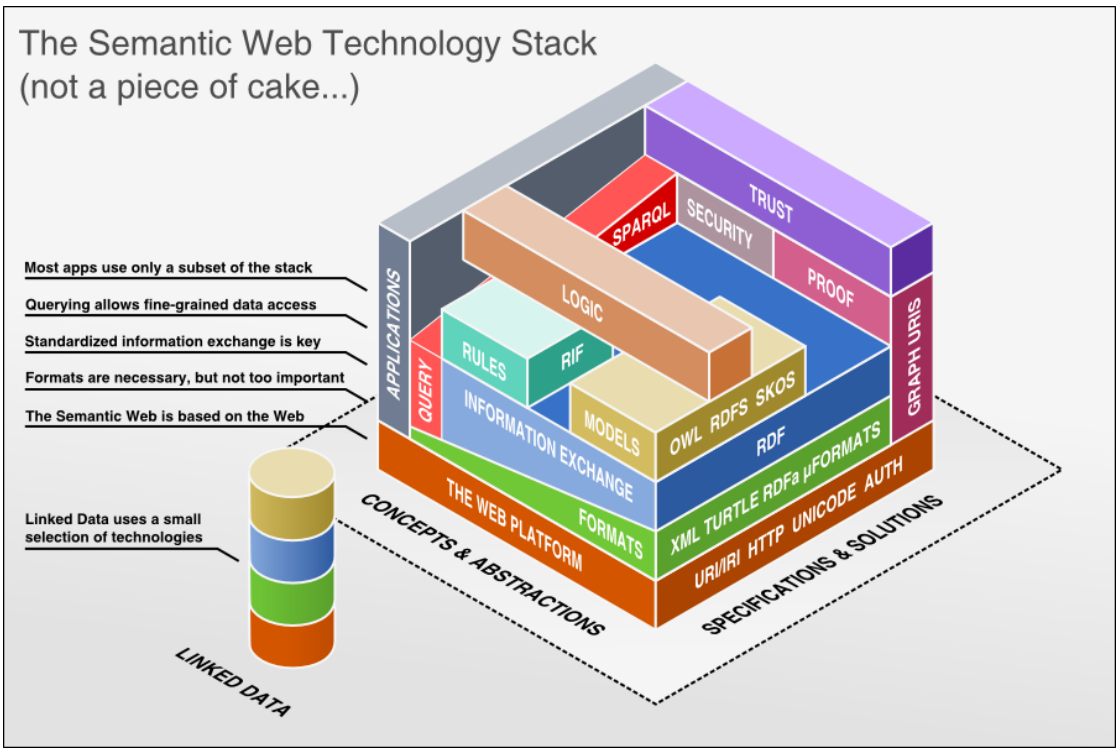



**Abbildung 2.1** – Populäre Darstellung des SWT Stapels von Benjamin Nowack, CC BY 3.0 Lizenz, `https://creativecommons.org/licenses/by/3.0/`

### 2.1.2 Datenformate und -modelle für den Informationsaustausch

Grundlegend für den interoperablen Daten- und Informationsaustausch zwischen mit dem Internet verbundenen Systemen sind standardisierte Formate und Modelle für Daten und Informationen. Abbildung 2.2 zeigt als Übersicht einen Ausschnitt wichtiger Standardisierungen in diesem Zusammenhang, im Zeitraum zwischen 1996 und 2016. In diesem Abschnitt werden drei für diese Arbeit relevante Formate beziehungsweise Modelle kurz vorgestellt. Dazu zählt XML in Abschnitt 2.1.2.1, sowie RDF und RDFS in den Abschnitten 2.1.2.2 und 2.1.2.3.

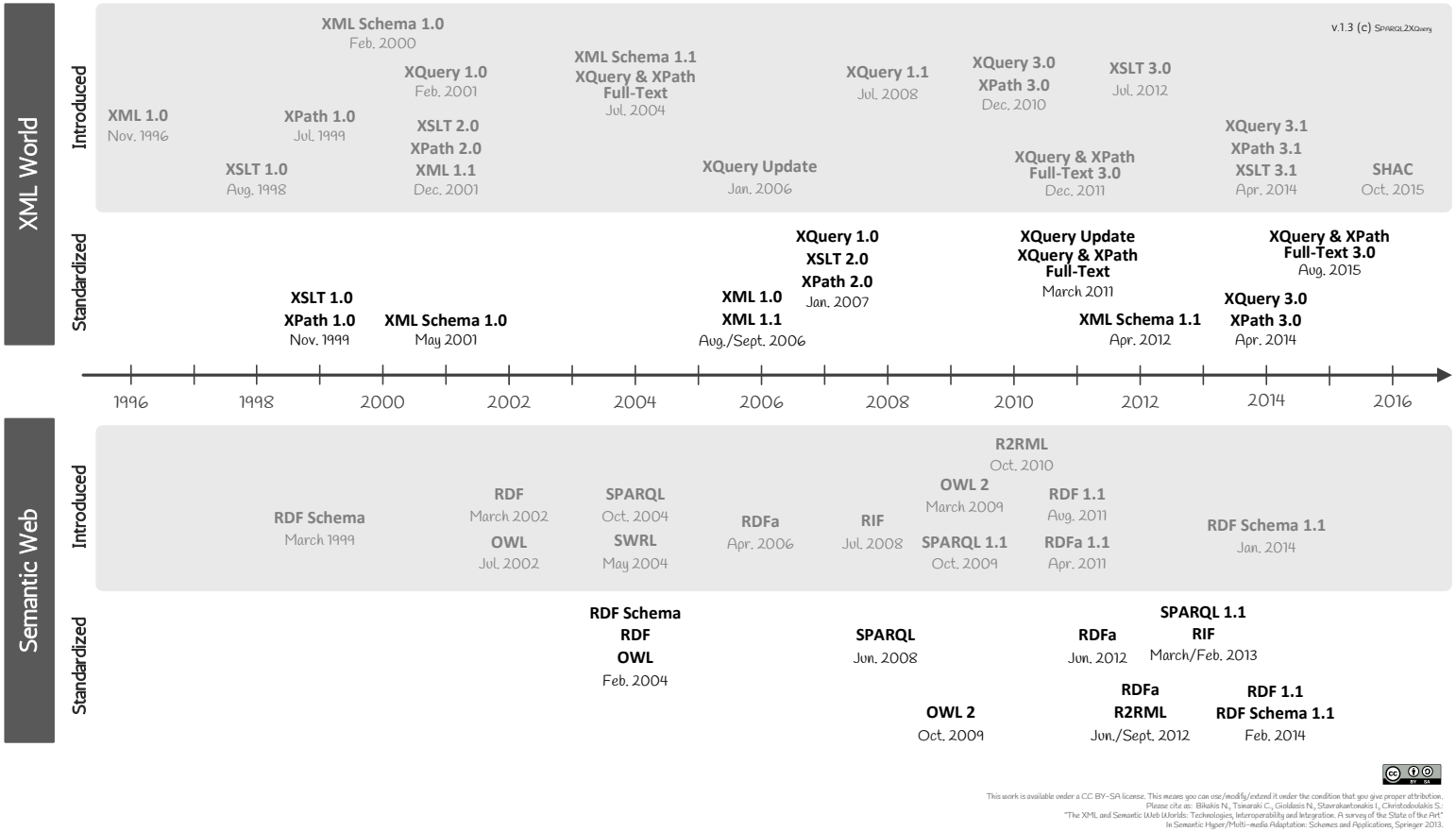



**Abbildung 2.2** – Eine Zeitachse, die wichtige XML und Semantic Web Standardisierungen darstellt, aus [22], CC BY-SA Lizenz, `https://creativecommons.org/licenses/by-sa/4.0/`

#### 2.1.2.1 Extensible Markup Language

Die Extensible Markup Language (XML) [23] ist eine Auszeichnungssprache zur Strukturierung von Daten, welche auf der im ISO 8879:1986[1] Standard definierten Sprache Standard Generalized Markup Language (SGML) basiert. Das World Wide Web Consortium (W3C), welches das Datenformat standardisierte, empfiehlt dessen Einsatz zum Austausch von Daten über das Internet und auch in anderen systemübergreifenden Bereichen. XML formatiert Daten Text-basiert, unterstützt aber auch binäre Serialisierungen. Die Text-basierten Serialisierungen von XML sind sowohl für Menschen als auch für Maschinen lesbar.

XML im Semantic Web

Für das Semantic Web ist XML insofern von Bedeutung, als es die syntaktische Grundlage für den Datenaustausch zwischen heterogenen Systemen bildet. Es ist unter anderem möglich, eine OWL Ontologie in XML zu serialisieren. Durch das einheitliche Vokabular kann Unabhängigkeit zwischen Anwendungen und den Daten erreicht werden, wodurch der Austausch von Daten zwischen unterschiedlichen Anwendungen ermöglicht wird.

#### 2.1.2.2 Resource Description Framework

Konzept

Das Resource Description Framework (RDF) [24–26] ist eine Methode zur Strukturierung von Daten und stellt somit ein *Datenmodell* dar. Es ermöglicht es, *Aussagen* über beliebige sogenannte *Ressourcen* im Web zu spezifizieren und war zunächst für den Transport von *Metadaten* gedacht [27]. Inzwischen stellt RDF die Basis für den Informationsaustausch im Semantic Web dar. Eine Ressource wird über einen Uniform Resource Identifier (URI) identifiziert. Somit können alle *identifizierbaren* Objekte erfasst werden, wenngleich diese nicht *abrufbar* sind. Für eine Übersicht über die Sprachkonstrukte von RDF sei auf die Tabellen 2.1, 2.2 und 2.3 verwiesen. Eine Aussage hat in RDF die Form: *Subjekt – Prädikat - Objekt* und bildet somit ein *Tripel* beziehungsweise *3-Tupel*. So lässt sich etwa die Aussage

„`https://example.org/index.html` `ist in der Sprache` `Englisch`“

wie folgt in die Bestandteile des Tripels zerlegen:

| | |
|---|---|
| `Subjekt` | `https://example.org/index.html` |
| `Prädikat` | `ist in der Sprache` |
| `Objekt` | `Englisch` |

Während ein Subjekt immer eine Ressource ist, kann ein Objekt entweder ein *Literal* oder eine beliebige andere Ressource sein. Ein Prädikat ist eine Spezialisierung einer Ressource. Tripel bilden einen *RDF Graphen*, siehe Abbildung 2.3.

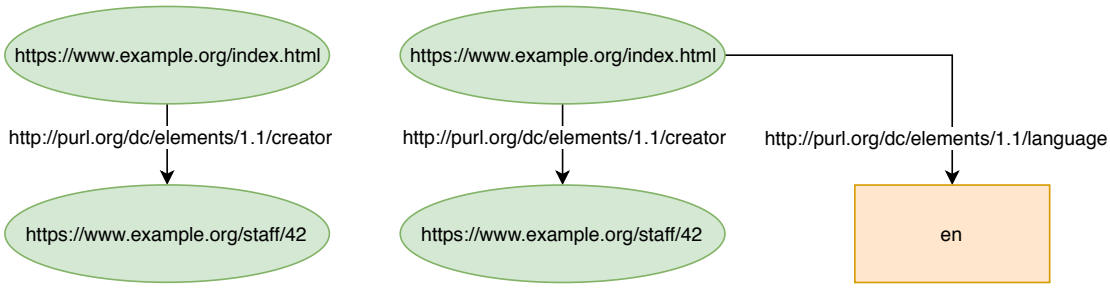
[1] ISO 8879:1986 Information processing – Text and office systems – Standard Generalized Markup Language (SGML)

Kurzschreibweise

Wie bereits erwähnt, werden Ressourcen in RDF mit URIs identifiziert. Es gibt die Möglichkeit den voll qualifizierten Namen – der so genannte *QName* – durch eine Kurzform auszudrücken, welche ein *Präfix* verwendet. Das Präfix, gefolgt von einem Doppelpunkt und anschließend dem *lokalen Namen*, bildet den voll qualifizierten Namen. Das Präfix wird dabei in der Regel mit einem Namensraum assoziiert. Wenn also etwa das Präfix `ex` mit dem Namensraum `https://example.com/` assoziiert wird, dann bildet `ex:index.html` die Kurzform für `https://example.com/index.html`. Demnach können die beiden in Abbildung 2.3 (b) enthaltenen Aussagen in der Tripel Kurzschreibweise wie folgt darstellt werden:

```
ex:index.html  dc:creator   exstaff:42
ex:index.html  dc:language  "en"
```

Hierbei stehen die beiden Präfixe `exstaff` und `dc` für die beiden Namensräume `https://example.org/staff/` respektive `http://purl.org/dc/elements/1.1/`. Letzterer Namensraum bezieht sich auf das standardisierte Vokabular zur Spezifikation von Metadaten *Dublin Core Metadata* [28].




(a) Einfache Aussage (b) Zwei Aussagen

**Abbildung 2.3** – Beispiele für RDF Graphen. Grüne Ellipsen stellen *Ressourcen* und das orangene Rechteck ein *Literal* dar

| Vokabel | Superklasse | Beschreibung |
|---|---|---|
| rdf:HTML | rdfs:Literal | RDF Literal mit HTML Inhalt |
| rdf:langString | rdfs:Literal | Datentyp für die Sprachangabe bei Literalen |
| rdf:PlainLiteral | rdfs:Literal | Generische Literale in OWL 2 und RIF |
| rdf:XMLLiteral | rdfs:Literal | Datentyp für XML Literale |

**Tabelle 2.1** – Datentyp-Vokabeln von RDF

| Vokabel | Superklasse | Beschreibung der Instanzen |
|---|---|---|
| rdf:Property | rdfs:Resource | RDF Eigenschaften (Properties) |
| rdf:Statement | rdfs:Resource | RDF Aussagen (Statements) |
| rdf:Bag | rdfs:Container | Ungeordnete Kontainer |
| rdf:Seq | rdfs:Container | Geordnete Kontainer |
| rdf:Alt | rdfs:Container | Kontainer, welche Alternativen enthalten |
| rdf:List | – | Listen |

**Tabelle 2.2** – Klassen-Vokabeln von RDF

| Vokabel | Domäne | Bereich | Beschreibung |
|---|---|---|---|
| rdf:type | rdfs:Resource | rdfs:Class | Subjekt ist Instanz einer Klasse |
| rdf:subject | rdf:Statement | rdfs:Resource | Subjekt einer RDF Aussage |
| rdf:predicate | rdf:Statement | rdfs:Resource | Prädikat einer RDF Aussage |
| rdf:object | rdf:Statement | rdfs:Resource | Objekt einer RDF Aussage |
| rdf:value | rdfs:Resource | rdfs:Resource | Darstellung strukturierter Werte |
| rdf:first | rdf:List | rdfs:Resource | Erstes Listenelement |
| rdf:rest | rdf:List | rdf:List | Liste ab dem zweiten Listenelement |

**Tabelle 2.3** – Prädikat-Vokabeln von RDF

#### 2.1.2.3 Resource Description Framework Schema

Das Resource Description Framework Schema (RDFS) [29] erweitert RDF um ein Vokabular, welches verwendet werden kann, um eine konkrete Domäne zu modellieren. Es bildet eine Sprache, mit der einfache Ontologien ausgedrückt werden können. Dies wird durch die Möglichkeit der Formulierung von Klassen und Instanzen erreicht, sowie durch die Bildung von *Taxonomien* und durch die so genannten *Properties* zur Angabe von Beziehungen zwischen Ressourcen.

Einschränkungen

Mittels der beiden Properties `rdfs:domain` und `rdfs:range` können der *Definitions-* beziehungsweise der *Wertebereich* von Properties auf Klassen angegeben werden. Auch in Bezug auf Literale kann mittels `rdfs:range` ein bestimmter Datentyp gefordert werden. Für eine Übersicht über RDFS-Sprachkonstrukte sei auf die Tabellen 2.4 und 2.5 verwiesen.

| Vokabel | Superklasse | Instanz von | Beschreibung der Instanzen |
|---|---|---|---|
| rdfs:Resource | - | rdfs:Class | Entitäten in RDF |
| rdfs:Class | rdfs:Resource | rdfs:Class | Klassen in RDF |
| rdfs:Literal | rdfs:Resource | rdfs:Class | Literale Werte (Strings/Integers) |
| rdfs:Datatype | rdfs:Class | rdfs:Class | Unterklassen von rdfs:Literal |
| rdf:langString | rdfs:Literal | rdfs:Datatype | Sprach-Tag Literale |
| rdf:HTML | rdfs:Literal | rdfs:Datatype | HTML Literale |
| rdf:Property | rdfs:Resource | rdfs:Class | RDF Properties |

**Tabelle 2.4** – Klassenvokabeln von RDFS

| Vokabel | Domäne | Bereich | Beschreibung |
|---|---|---|---|
| rdfs:range | rdf:Property | rdfs:Class | Wertebereicheinschränkung |
| rdfs:domain | rdf:Property | rdfs:Class | Definitionsbereicheinschr. |
| rdf:type | rdfs:Resource | rdfs:Class | Instanz einer Klasse |
| rdfs:subClassOf | rdfs:Class | rdfs:Class | Transitive Unterklasse |
| rdfs:subPropertyOf | rdf:Property | rdf:Property | Transitives Untermerkmal |
| rdfs:label | rdfs:Resource | rdfs:Literal | Bezeichnung einer Ressource |
| rdfs:comment | rdfs:Resource | rdfs:Literal | Beschreibung einer Ressource |

**Tabelle 2.5** – Prädikat-Vokabeln von RDFS

### 2.1.3 Sprachen zur Wissensrepräsentation und -abfrage

Während zuvor XML, RDF und RDFS beschrieben wurden, werden in diesem Abschnitt auf diesen aufsetzende Sprachen zur Darstellung und Abfrage von Wissen beschrieben. Zunächst folgt zum besseren Verständnis eine kurze syntaktische Einführung in die darunterliegenden Konzepte der Prädikatenlogik erster Stufe und Beschreibungslogiken. Während RDF und RDFS bereits genutzt werden können, um Assertionaler Formalismus (ABox) respektive Terminologischer Formalismus (TBox) auszudrücken und damit Ontologien zu formulieren, stellt die in Abschnitt 2.1.3.3 beschriebene Ontologiesprache OWL weitere nützliche Sprachkonstrukte bereit. Weiter wird in Abschnitt 2.1.3.4 die OWL ergänzende Regelsprache Semantic Web Rule Language (SWRL) beschrieben. Schließlich folgt eine Beschreibung der RDF-Abfragesprache SPARQL in Abschnitt 2.1.3.5.

#### 2.1.3.1 Prädikatenlogik erster Stufe

Dieser Abschnitt gibt Definitionen nach [30] wieder. Für weitere Informationen zur Prädikatenlogik erster Stufe sei der Leser auf die Literatur verwiesen [31–33].

Symbole

Die logischen Zeichen der Prädikatenlogik erster Stufe umfassen die Symbole der Aussagenlogik $\neg, \wedge, \vee, \rightarrow, \leftrightarrow, (,)$ sowie zusätzlich:

$\forall$ Allquantor $\quad \mathsf{v_i}$ Individuenvariablen
$\exists$ Existenzquantor $\quad \doteq$ Objektsprachliches Gleichheitssymbol
, Komma

Die Menge dieser Zusatzsymbole wird im Folgenden mit dem Symbol $\mathcal{Z}$ bezeichnet.

Signatur

Die Signatur wird weiterhin durch ein Tripel gebildet $\Sigma = (\mathsf{F_\Sigma}, \mathsf{P_\Sigma}, \alpha_\Sigma)$, wobei:

$\mathsf{F_\Sigma}, \mathsf{P_\Sigma}$ endliche oder abzählbar unendliche Mengen

$\mathsf{F_\Sigma}, \mathsf{P_\Sigma}$ und $\mathcal{Z}$ paarweise disjunkt

$\alpha_\Sigma \colon \mathsf{F_\Sigma} \cup \mathsf{P_\Sigma} \rightarrow \mathbb{N}$

Terme

Überdies sei $\mathsf{Term_\Sigma}$, die Menge der *Terme über* $\Sigma$, induktiv definiert:

(1) $\mathsf{Var} \subseteq \mathsf{Term_\Sigma}$

(2) Mit $\mathsf{f} \in \mathsf{F_\Sigma}$, $\alpha(\mathsf{f}) = \mathsf{n}$ und $\mathsf{t_1}, ..., \mathsf{t_n} \in \mathsf{Term_\Sigma}$
ist auch $\mathsf{f}(\mathsf{t_1}, ..., \mathsf{t_n}) \in \mathsf{Term_\Sigma}$

Atomare Formeln

Nun sei die Menge der *atomaren Formeln* $\mathsf{At_\Sigma}$ definiert als:

$$\mathsf{At_\Sigma} := \{\mathsf{s} \doteq \mathsf{t} | \mathsf{s}, \mathsf{t} \in \mathsf{Term_\Sigma}\} \cup \{\mathsf{p}(\mathsf{t_1}, ..., \mathsf{t_n}) | \mathsf{p} \in \mathsf{P_\Sigma}, \mathsf{t_i} \in \mathsf{Term_\Sigma}\}$$

Formel

Dann ist $\mathsf{For_\Sigma}$, die Menge der *Formeln über* $\Sigma$, induktiv definiert durch:

(1) $\{1, 0\} \cup \mathsf{At_\Sigma} \subseteq \mathsf{For_\Sigma}$

(2) Mit $\mathsf{x} \in \mathsf{Var}$ und $\mathsf{A}, \mathsf{B} \in \mathsf{For_\Sigma}$ sind ebenfalls in $\mathsf{For_\Sigma}$:
$\neg\mathsf{A}, (\mathsf{A} \wedge \mathsf{B}), (\mathsf{A} \vee \mathsf{B}), (\mathsf{A} \rightarrow \mathsf{B}), (\mathsf{A} \leftrightarrow \mathsf{B}), \forall\mathsf{x}\mathsf{A}, \exists\mathsf{x}\mathsf{A}$

Beispiel

Prädikatenlogik erster Stufe ermöglicht bereits viele Fakten formal auszudrücken, wie beispielsweise:

$\forall\mathsf{x}(\mathsf{Katze}(\mathsf{x}) \rightarrow \mathsf{flauschig}(\mathsf{x}))$
$\mathsf{Katze}(\mathsf{Sally})$
$\exists\mathsf{x}\exists\mathsf{y}(\mathsf{Katze}(\mathsf{x}) \wedge \mathsf{Marshmallow}(\mathsf{y}) \rightarrow \mathsf{isst}(\mathsf{x}, \mathsf{y}))$

Diese beiden Formalisierungen drücken aus, dass (1) *alle Katzen flauschig sind*, beziehungsweise dass (2) *Sally eine Katze ist*, beziehungsweise dass (3) es *eine Katze gibt, die einen Marshmallow isst.*

### 2.1.3.2 Beschreibungslogiken

Erste Beschreibungslogiken (englisch: Description Logics, DLs) entstanden bereits in den frühen 1980ern und gehen auf *Semantische Netze* [34] und *KL-ONE* [35] zurück. Sie sind entscheidbare Fragmente von Prädikatenlogik erster Stufe und daher mit formeller Semantik ausgestattet. Die formelle Semantik ermöglicht Mehrdeutigkeiten auszuschließen, sowie die Deduktion neuer Informationen. Abbildung 2.4 zeigt ein Beispiel für ein semantisches Netzwerk.

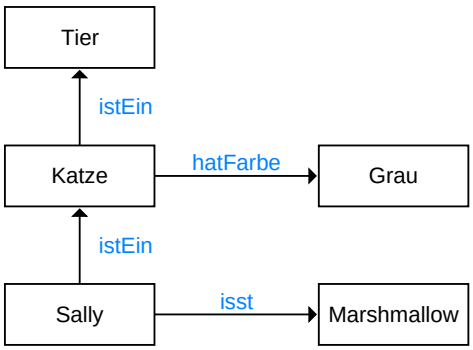



**Abbildung 2.4** – Ein Beispiel für ein semantisches Netzwerk

Signatur

Im Folgenden werden die grundlegenden Entitäten von DLs kurz beschrieben:

Konzepte

Korrespondiert zum Konzept von *Klassen*. Äquivalent zu *unären Prädikaten* der Prädikatenlogik erster Stufe. Bsp.: *Katze*, *Tier*, *Mensch*.
So gibt beispielsweise der Ausdruck

$$\mathsf{Katze}(\mathsf{Sally})$$

an, dass das Individuum Sally eine Instanz der Klasse Katze ist. Spezielle Konzepte sind $\top$ und $\bot$, die für das *universelle Konzept* beziehungsweise das *leere Konzept* stehen. Während erstere *alle* Instanzen enthält, enthält letzteres *keine* Instanzen.

Rollen

Korrenspondiert zum Konzept von *Attributen*. Äquivalent zu *binären Prädikaten* der Prädikatenlogik erster Stufe. Bsp.: *hatFarbe*, *isst*, *istEin*. Ein Beispiel für einen Rollenausdruck wäre:

$$\mathsf{isst}(\mathsf{Sally}, \mathsf{Marshmallow})$$

was ausdrückt, dass Sally einen Marshmallow isst.

| | |
|---|---|
| Individuen | Korrespondiert zum Konzept von *Instanzen*. Äquivalent zu *Konstanten* der Prädikatenlogik erster Stufe. Bsp.: Sally, Lucas, Barbara. Wichtig ist hierbei, dass nicht angenommen wird, dass Individuen-Namen eindeutig sind, also keine so genannte *unique name assumption* bei welcher gilt, dass zwei unterschiedliche Individuen-Namen tatsächlich das selbe Individuum benennen. |

Operatoren

DLs verfügen über verschiedene Operatoren. Die folgenden Operatoren beziehen sich auf die DL $\mathcal{AL}$ [36] mit *komplexen Konzepten* $\mathsf{C}, \mathsf{D}$ und *atomarem Konzept* $\mathsf{A}$, sowie *Rollen* $\mathsf{R}$:

| | |
|---|---|
| $\mathsf{C} \sqcap \mathsf{D}$ | Schnitt aus zwei Konzepten. Alle Instanzen dieses komplexen Konzepts müssen sowohl Instanzen von $\mathsf{C}$ wie auch von $\mathsf{D}$ sein. |
| $\neg\mathsf{A}$ | Negation. Alle Instanzen dieses komplexen Konzepts sind keine Instanzen von $\mathsf{A}$. |
| $\forall\mathsf{R}.\mathsf{C}$ | Einschränkung auf einen Wert. Für die Rolle $\mathsf{R}$ werden mögliche Instanzen auf die atomare Klasse $\mathsf{C}$ eingeschränkt. |
| $\exists\mathsf{R}.\top$ | beschränkte existentielle Einschränkung. Für die Rolle $\mathsf{R}$ existiert mindestens eine Instanz. |

Bestandteile

TBox, ABox und ggf. Rollen Axiome (RBox) bilden die wesentlichen Bestandteile, aus denen sich ein *Wissensverarbeitungssystem* basierend auf Beschreibungslogiken, zusammensetzt. Dabei umfasst die TBox alle Klassenbeschreibungen beziehungsweise -Axiome, die ABox alle Aussagen in Bezug auf Individuen und schließlich die RBox alle Rollenbeschreibungen beziehungsweise -Axiome. Entsprechende Sprachkonstrukte von TBox, ABox und RBox werden im Abschnitt 2.1.3.2.2 angeführt. Es gibt nicht *die eine* Beschreibungslogik, stattdessen ist die Ausdrucksmächtigkeit im Verhältnis zur Komplexität des automatischen Schlussfolgerns zu berücksichtigen. Tabelle 2.6 zeigt eine Übersicht über DLs und deren Sprachkonstrukte. Für weiterführende Literatur zu DL sei auf [37] verwiesen.

| DLs | Konstruktoren | | Axiome | | |
|---|---|---|---|---|---|
| | Konzepte | Rollen | TBox | ABox | RBox |
| $\mathcal{AL}$ | $\top$, A, C1 $\sqcap$ C1, $\exists$R.C | R | A $\equiv$ C<br>$C_1 \sqsubseteq C_2$ | a$\exists$C<br>R(a, b) | |
| $\mathcal{ALC}$ | -\|\|-, $\neg$C | -\|\|- | -\|\|- | -\|\|- | |
| $\mathcal{S}$ | -\|\|- | -\|\|- | -\|\|- | -\|\|- | tra R |
| $+\mathcal{I}$ | | $R^-$ | | | |
| $+\mathcal{H}$ | | | | | $R_1 \sqsubseteq R_2$ |
| $+\mathcal{F}$ | | | | | fun R |
| $+\mathcal{N}$ | $\exists^{\geqslant n} S$ | | | | |
| $+\mathcal{Q}$ | $\exists^{\geqslant n} S.C$ | | | | |
| $+\mathcal{O}$ | $\{i\}$ | | | | |
| $+\mathcal{R}$ | $\exists R.Self$ | $\neg$R<br>U | | | R $\circ$ S$\dot{\sqsubseteq}$R<br>S $\circ$ R$\dot{\sqsubseteq}$R<br>ref R<br>irr R<br>asy R<br>Disj(R, S) |

**Tabelle 2.6** – Übersicht über DL Sprachkonstrukte, adaptiert aus [38]

**2.1.3.2.1 Die Beschreibungslogik $\mathcal{ALC}$** Die Beschreibungslogik $\mathcal{ALC}$ (Abkürzung für *Attributive Language with Complement*) erweitert die zuvor vorgestellte DL $\mathcal{AL}$ um die Negation komplexer Konzepte [36]. Sie bildet die Basis für weitere, komplexere Sprachen und eröffnet bereits die Möglichkeit, einen signifikanten Teil von OWL DL zu modellieren [39]. Siehe auch Tabelle 2.6 für eine Übersicht über die Sprachkonstrukte und die Unterschiede zu $\mathcal{AL}$. Für die Grammatik von $\mathcal{ALC}$ ergibt sich somit, mit den *komplexen Konzepten* C, D, dem *atomaren Konzept* A und der *Rolle* R:

$$\mathsf{C}, \mathsf{D} := \mathsf{A} | \top | \bot | \neg \mathsf{C} | \mathsf{C} \sqcap \mathsf{D} | \mathsf{C} \sqcup \mathsf{D} | \forall \mathsf{R}.\mathsf{C} | \exists \mathsf{R}.\mathsf{C}$$

**2.1.3.2.2 Die Beschreibungslogik $\mathcal{SROIQ}$** Während die zuvor vorgestellten Beschreibungslogiken noch einigen Einschränkungen unterlagen, im Vergleich zum Sprachumfang von OWL, so bildet die Beschreibungslogik $\mathcal{SROIQ}$ [40] die Basis für OWL 1.1, weshalb sie hier ebenfalls kurz angeführt wird. Tabelle 2.7 bietet eine Übersicht über die Sprachkonstrukte von $\mathcal{SROIQ}$. Für einen Vergleich zu anderen DLs sei außerdem auf Tabelle 2.6 verwiesen.

| TBox | | ABox | | RBox | |
|---|---|---|---|---|---|
| Konzept | Syntax | Konzept | Syntax | Konzept | Syntax |
| Inklusion | $\mathsf{C} \sqsubseteq \mathsf{D}$ | Klassen-zugehörigkeit | $\mathsf{C}(\mathsf{a})$ | Rolleninklusion | $\mathsf{R}_1 \sqsubseteq \mathsf{R}_2$ |
| Äquivalenz | $\mathsf{C} \equiv \mathsf{D}$ | Rollen-zugehörigkeit | $\mathsf{R}(\mathsf{a}, \mathsf{b})$ | komplexe Rolleninklusion | $\mathsf{R}_1 \circ ... \circ \mathsf{R}_{\sqsubseteq}\mathsf{R}$ |
| | | negierte Rollen-zugehörigkeit | $\neg\mathsf{S}(\mathsf{a}, \mathsf{b})$ | Transitivität | $\mathsf{tra}\ \mathsf{R}$ |
| | | Gleichheit | $\mathsf{a} \approx \mathsf{b}$ | Symmetrie | $\mathsf{sym}\ \mathsf{R}$ |
| | | Ungleichheit | $\mathsf{a} \not\approx \mathsf{b}$ | Reflexivität | $\mathsf{ref}\ \mathsf{R}$ |
| | | | | Irreflexivität | $\mathsf{irr}\ \mathsf{S}$ |
| | | | | Disjunktheit | $\mathsf{Disj}(\mathsf{R}, \mathsf{S})$ |

**Tabelle 2.7** – Überblick über die Sprachkonstrukte von $\mathcal{SROIQ}$ nach TBox, ABox und RBox mit $\mathsf{C}, \mathsf{D}$ *komplexen Konzepten*, $\mathsf{R_i}, \mathsf{S}, \mathsf{T}$ *Rollen* und $\mathsf{a}, \mathsf{b}$ Individuen

#### 2.1.3.3 Web Ontology Language

OWL & DLs

Die standardisierte Ontologie-Sprache Web Ontology Language (OWL)[2] [41] stellt eine wichtige Anwendung von DLs dar. Es lassen sich die Sprachkonstrukte von OWL direkt in äquivalente DL Syntax übersetzen. Für eine Übersicht, wie OWL Konstruktoren auf DL Syntax abgebildet werden, sei auf die Tabellen 2.8 und 2.9 verwiesen. Dabei entsprechen OWL *Classes* in DL *Konzepten* und OWL *Properties* in DL den *Rollen*. OWL Properties können dabei auch entsprechende Eigenschaften wie *Transitivität*, *Symmetrie*, etc. annehmen, die im Englischen *property characteristics* genannt werden.

Profiles

Es gibt verschiedene Varianten von OWL, welche *Profiles* genannt werden und die sich im Sprachumfang und damit der Ausdrucksmächtigkeit, sowie der damit verbundenen Komplexität und Entscheidbarkeit unterscheiden. So entspricht beispielsweise die Untersprache OWL *Lite* der DL $\mathcal{SHIF}_{(\mathcal{D})}$ und OWL *DL* der DL $\mathcal{SHOIN}_{(\mathcal{D})}$. Hierbei steht $(\mathcal{D})$ für die Verwendung von *Datentypen*. OWL *Lite* fügt im Vergleich zur DL $\mathcal{ALC}$ noch *Datentypen*, *funktionale Attribute*, *transitive Rollen* und *Rollenhierarchien* hinzu und hat eine Komplexität von *ExpTime* im schlimmsten Fall. OWL *DL* umfasst im Vergleich zur DL $\mathcal{ALC}$ außerdem noch *Nominals*, d.h. Klassen, welche durch eine Menge von Instanzen beschrieben werden zu Klassenbeschreibungen und *beliebige Kardinalitätsbeschränkungen*. Die Komplexität von OWL *DL* ist *NExpTime* im schlimmsten Fall. OWL *Full*, welches sowohl auf OWL *DL*,

[2]Siehe auch `https://www.w3.org/TR/owl-overview/`

| Konstruktor | DL Syntax |
|---|---|
| owl:Thing | $\top$ |
| owl:Nothing | $\bot$ |
| intersectionOf | $C_1 \sqcap ... \sqcap C_n$ |
| unionOf | $C_1 \sqcup ... \sqcup C_n$ |
| complementOf | $\neg C$ |
| oneOf | $\{a_1, ..., a_m\}$ |
| allValuesFrom | $\forall R.C$ |
| someValuesFrom | $\exists R.C$ |
| hasValue | $\exists R.\{a\}$ |
| minCardinality | $\geqslant nR$ |
| maxCadinality | $\leqslant nR$ |
| inverseOf | $R^-$ |

**Tabelle 2.8** – Übersicht über OWL Konstruktoren, adaptiert aus [42]

| Konstruktor | DL Syntax |
|---|---|
| subClassOf | $C_1 \sqsubseteq C_2$ |
| equivalentClass | $C_1 \equiv C_2$ |
| subPropertyOf | $R_1 \sqsubseteq R_2$ |
| equivalentProperty | $R_1 \equiv R_2$ |
| disjointWith | $C_1 \sqcap C_2 \equiv \bot$ beziehungsweise $C_1 \sqcap \neg C_2$ |
| sameAs | $\{a_1\} \equiv \{a_2\}$ |
| differentFrom | $\{a_1\} \equiv \neg\{a_2\}$ |
| TransitiveProperty | definiert eine transitive Rolle |
| FunctionalProperty | definiert eine funktionale Rolle |
| InverseFunctionalProperty | definiert eine inverse funktionale Rolle |
| SymmetricProperty | definiert eine symmetrische Rolle |

**Tabelle 2.9** – Übersicht über OWL Konstruktoren für Axiome, adaptiert aus [42]

als auch auf RDFS basiert, ist jedoch keine DL mehr, sondern eine unentscheidbare Untermenge von Prädikatenlogik erster Stufe [38]. In dieser Arbeit wird ausschließlich OWL 2 eingesetzt. OWL 2 entspricht der zuvor vorgestellten DL $\mathcal{SROIQ}_{(\mathcal{D})}$. In OWL 2 gibt es ebenfalls unterschiedliche Sprachvarianten beziehungsweise Profiles: OWL 2 *EL*, OWL 2 *QL*, and OWL 2 *RL*. Für weitere Informationen zu den Unterschieden zwischen den OWL Profiles sei auf die Literatur verwiesen [39; 43].

Datentypen

OWL 2 unterscheidet weiterhin zwischen sogenannten *object properties* und *data properties*, in Abhängigkeit von der Art des zweiten Arguments, des DL *Prädikats* beziehungsweise des OWL *property*. Handelt es sich beim zweiten Argument nicht um ein *Individuum*, sondern um ein *Literal*, so handelt es sich um eine *data property*, welche einen eindeutig definierten Datentyp hat. Die Menge der Datentypen in OWL ist dabei erweiterbar und viele XML Schema Datentypen sind ebenfalls erlaubt. OWL erlaubt in Verbindung mit den Datentypen die Angabe von sogenannten *beschränkenden Facetten*, die ihren Ursprung aus der XML Schema Sprache haben. Dieses Konzept geht über das von DLs hinaus, weshalb es keine Standardnotation zu dessen Darstellung gibt. Ein Beispiel für den Einsatz von beschränkenden Facetten lässt sich jedoch in funktionaler OWL Syntax darstellen, siehe Quelltext 2.1.

```
EquivalentClasses(
    :SeniorCat
    ObjectIntersectionOf(
        :Cat
        DataSomeValuesFrom(
            :hasAge
            DatatypeRestriction(
                xsd:integer
                xsd:minExclusive
                "10"^^xsd:integer))))
```

**Quelltext 2.1** – Beispiel zur Verwendung von beschränkenden Facetten in OWL in funktionaler OWL Syntax

**2.1.3.3.1 Syntaxen** Für die syntaktische Darstellung von OWL existieren verschiedene Formen, die je nach Anwendungszweck zu wählen sind, siehe Abbildung 2.5. Beispielsweise eignet sich die funktionale Syntax von OWL für eine kompakte, gut menschenlesbare Darstellung, während die RDF/XML Syntax dagegen geeigneter ist, um die dargestellte Ontologie zwischen unterschiedlichen Applikationen auszutauschen. Grundsätzlich können die unterschiedlichen Serialisierungen von OWL Ontologien aber auch ineinander überführt werden.

```
<owl:Class rdf:about="http://my.ontology.com#Mother">
    <owl:equivalentClass>
        <owl:Class>
            <owl:intersectionOf rdf:parseType="Collection">
                <rdf:Description rdf:about="http://my.ontology.com#Female"/>
                <rdf:Description rdf:about="http://my.ontology.com#Parent"/>
            </owl:intersectionOf>
        </owl:Class>
    </owl:equivalentClass>
</owl:Class>
```

(a) RDF/XML Syntax

```
<EquivalentClasses>
    <Class IRI="#Mother"/>
    <ObjectIntersectionOf>
        <Class IRI="#Female"/>
        <Class IRI="#Parent"/>
    </ObjectIntersectionOf>
</EquivalentClasses>
```

(b) OWL 2 XML Syntax

```
:Mother rdf:type owl:Class ;
        owl:equivalentClass [ owl:intersectionOf ( :Female
                                                   :Parent
                                                 ) ;
                              rdf:type owl:Class
                            ] .
```

(c) Turtle Syntax

```
Class: <http://my.ontology.com#Mother>

    EquivalentTo:
        <http://my.ontology.com#Female>
         and <http://my.ontology.com#Parent>
```

(d) Manchester Syntax

```
EquivalentClasses(:Mother ObjectIntersectionOf(:Female :Parent))
```

(e) OWL funktionale Syntax

**Abbildung 2.5** – Verschiedene OWL Serialisierungen des DL Ausdrucks $\mathsf{Mother} \equiv \mathsf{Female} \sqcap \mathsf{Parent}$, wobei Mother, Female und Parent DL *Konzepte* beziehungsweise OWL *Klassen* sind

#### 2.1.3.4 Semantic Web Rule Language

Mit der SWRL[3] [44] lassen sich Konzepte und Zusammenhänge ausdrücken, die mit OWL alleine nicht ausgedrückt werden können. Gründe für den Einsatz von SWRL sind nach [45]:

- Deklarative Problembeschreibung
- Separieren des Wissens von der Programmierung
- Teilbarkeit des Wissens
- Bessere Wartbarkeit des Wissens

[3] siehe auch `https://www.w3.org/Submission/SWRL/`

Im Folgenden werden SWRL Regeln kurz definiert. SWRL Regeln basieren auf *Horn-Klauseln*, siehe Regel 2.1:

$$\underbrace{\mathsf{A}_1 \wedge \mathsf{A}_2 \wedge ... \wedge \mathsf{A}_n}_{Rumpf} \rightarrow \underbrace{H}_{Kopf} \tag{2.1}$$

wobei jedes $\mathsf{A}_i$ und $\mathsf{H}$ eine *atomare Formel* $\mathsf{P}(\mathsf{t}_1, ..., \mathsf{t}_m)$ der Prädikatenlogik erster Stufe ist, siehe Abschnitt 2.1.3.1. $\mathsf{P}$ ist hierbei ein $\mathsf{m}$-*äres* Prädikat, $\mathsf{t}_j$ *Terme*, also eine *Variable* oder ein *Ausdruck* $\mathsf{f}(\mathsf{s}_1, ..., \mathsf{s}_k)$ mit $\mathsf{f}$ $\mathsf{k}$-*äres Funktionssymbol* und *Termen* $\mathsf{s}_1, ..., \mathsf{s}_k$. Davon abgesehen sind keine Quantoren oder Negationen zulässig.

Teilweise lassen sich OWL Axiome in SWRL Regeln übersetzen und umgekehrt. Möchte man beispielsweise ausdrücken, dass alle Katzen, die älter als 10 Jahre alt sind, Seniorenkatzen sind, könnte man Axiom 2.2 verwenden:

$$\mathsf{SeniorCat} \equiv \mathsf{Cat} \sqcap \exists \mathsf{hasAge}.(\mathsf{xsd} : \mathsf{integer} \geqslant 10) \tag{2.2}$$

Es gilt zu beachten, dass dies keine Standard DL Notation ist, da hier *beschränkende Facetten* eingesetzt werden, siehe Abschnitt 2.1.3.3. Das Axiom lässt sich nun mittels SWRL als Regel formulieren, wie in Regel 2.3:

$$\mathsf{Cat}(\mathsf{x}) \wedge \mathsf{hasAge}(\mathsf{x}, \mathsf{y}) \wedge \mathsf{swrlb} : \mathsf{greaterThan}(\mathsf{y}, 10) \rightarrow \mathsf{SeniorCat}(\mathsf{y}) \tag{2.3}$$

Hierbei ist $\mathsf{swrlb} : \mathsf{greaterThan}$ ein vordefiniertes Prädikat aus dem Sprachumfang von SWRL, den so genannten *Built-Ins*.

#### 2.1.3.5 SPARQL Protocol and RDF Query Language

Die Sprache SPARQL Protocol and RDF Query Language (SPARQL)[4] [46] ist eine Abfrage- und Manipulationssprache für RDF basierte Wissensbanken. Wie das rekursive Akronym andeutet, hat SPARQL Ähnlichkeit mit der Sprache Structured Query Language (SQL). SQL ist jedoch für den Einsatz mit relationalen Datenbanken vorgesehen und berücksichtigt nicht das RDF Datenmodell – ganz im Unterschied zu SPARQL. Neben SPARQL existieren noch weitere Abfragesprachen für RDF, siehe [47].

Im Folgenden wird SPARQL anhand eines praktischen Beispiels dargestellt. An dieser Stelle sollen die Beispiel-Tripel aus Abschnitt 2.1.2.2 nochmals aufgegriffen werden. Der RDF Graph wird hier allerdings in *Turtle Syntax*[5] [48] dargestellt und ist um einige zusätzliche Tripel erweitert, siehe Quelltext 2.2. Unter anderem wird in den zusätzlichen Tripeln das FOAF Präfix und dessen zugeordneter Namensraum verwendet, welche auf die Friend of a Friend (FOAF) Ontologie [49] zurück

[4]Siehe `https://www.w3.org/TR/sparql11-overview/`
[5]Siehe `https://www.w3.org/TR/turtle/`

gehen – eine der ersten Anwendungen von SWT. FOAF wird hier eingesetzt, um den Namen der Autoren der Dokumente zu spezifizieren (letzte zwei Tripel). Außerdem wurden drei weitere Dokumente `index-de.html`, `doc.html` und `doc-de.html` mit zugehörigen Informationen zum jeweiligen Autor und der Sprache des Dokuments hinzugefügt:

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix ex: <https://example.com/> .
@prefix exstaff: <https://example.com/staff/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

ex:doc-de.html dc:creator exstaff:41 ;
        dc:language "de" .
ex:doc.html dc:creator exstaff:42 ;
        dc:language "en" .
ex:index-de.html dc:creator exstaff:41 ;
        dc:language "de" .
ex:index.html dc:creator exstaff:42 ;
        dc:language "en" .
ex:example.html dc:creator exstaff:41 ;
        dc:language "de" .
ex:example.html dc:creator exstaff:42 ;
        dc:language "en" .
exstaff:41 foaf:name "Elfriede Wierzbinka" .
exstaff:42 foaf:name "Anton Wierzbinka" .
```

**Quelltext 2.2** – Beispiel RDF Graph in Turtle Syntax

Angenommen, diese Tripel sind in einer RDF Graph-Datenbank, einem so genannten *Triplestore* oder auch *Subjekt-Prädikat-Objekt Datenbank* abgelegt. Unterstützt dieser Triplestore SPARQL Anfragen, so spricht man von einem *SPARQL Endpoint*. Eine mögliche Anfrage an diesen SPARQL Endpoint könnte wie in Quelltext 2.3 aussehen:

```
PREFIX dc:      <http://purl.org/dc/elements/1.1/>
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
SELECT ?creatorName (COUNT(?document) as ?germanDocuments)
WHERE {
  ?document dc:creator  ?creator .
  ?document dc:language "de" .
  ?creator  foaf:name   ?creatorName .
}
GROUP BY ?creatorName
```

**Quelltext 2.3** – Beispiel SPARQL Anfrage

Diese SPARQL Anfrage besteht aus vier Bestandteilen. Der erste Block, in welchem das Schlüsselwort `PREFIX` eingesetzt wird, dient dazu Namensräume zu definieren, siehe auch Kurzschreibweise von RDF Tripeln in Abschnitt 2.1.2.2. Die

nächste Zeile, in welcher das `SELECT` Schlüsselwort verwendet wird, bestimmt welche Variablen ausgegeben werden und wie diese zu formatieren sind. Im Beispiel sollen demnach die beiden Variablen `creatorName` und `germanDocuments` ausgegeben werden, wobei `germanDocuments` die Anzahl aller Elemente in der `document` Variablen darstellt. Im nächsten Block, welcher mit dem `WHERE` Schlüsselwort beginnt, wird die eigentliche Anfrage definiert. Hervorzuheben ist die Tripel-Schreibweise in Turtle-Syntax, siehe Abschnitt 2.1.2.2, sowie Abbildung 2.5. SPARQL ermöglicht es also, *Graph-Muster* zu definieren, wobei Variablen mit einem führenden Fragezeichen gekennzeichnet werden. Die letzte Zeile mit dem Schlüsselwort `GROUP BY` ist ein *Modifikator*, welcher in diesem Fall die Ergebnisse nach dem `creatorName` aufschlüsselt.
Diese Anfrage gibt insgesamt als Ergebnis eine Tabelle zurück, welche die Anzahl aller deutschsprachigen Dokumente nach Autoren aufschlüsselt, in Bezug auf das Beispiel also wie in Tabelle 2.10 dargestellt:

| creatorName | germanDocuments |
|---|---|
| ”Elfriede Wierzbinka” | ”3” |

**Tabelle 2.10** – Ergebnis der SPARQL-Abfrage aus dem Beispiel

### 2.1.4 Automatische Inferenz neuen Wissens

Wissensbasierte und Expertensyteme

In einem *wissensbasierten System* erfolgt eine Trennung zwischen dem Domänenwissen (*Wissensbasis*) und der Wissensverarbeitung [50]. Systeme, welche darüber hinaus Handlungsempfehlungen aus einer *Wissensbank* ableiten können, bezeichnet man als Expertensystem (ES). Nach Hayes-Roth besteht ein ES aus einer *Wissensbank* und einer *Inferenzmaschine* [51]. ES unterscheiden sich von dem allgemeineren Konzept der wissensbasierten Systeme durch die Herkunft des vorhandenen Wissens. Stammt dieses von Experten, spricht man von einem ES [52]. Im Unterschied zu klassischen Programmieransätzen lassen sich mit einem wissensbasierten System folgende Aspekte realisieren [50]:

- Trennung zwischen Problembeschreibung und Problemlösung
- Explizites Wissen über den Anwendungsbereich

Aus dem zweiten Aspekt folgt weiterhin, dass das Wissen teilbar und einfacher zu warten wird. Bewertungskriterien für Expertensystemprojekte sind ebenfalls in [52] zu finden.

Inferenzaufgaben

Nach Baader et al. können Inferenzaufgaben nach Aufgaben für TBox und ABox unterschieden werden [37]. In Hinsicht auf OWL sind folgende Anfragen typisch, nach Hitzler et al. in [39]:

*Klassenäquivalenz* Zwei Klassen $\mathsf{C}, \mathsf{D}$ sind genau dann äquivalent, wenn $\mathsf{C} \sqsubseteq \mathsf{D}$ und $\mathsf{D} \sqsubseteq \mathsf{C}$ logische Konsequenz der Wissensbasis ist.

*Unterklassenbeziehung* $\mathsf{C}$ ist dann Unterklasse von $\mathsf{D}$, wenn $\mathsf{C} \sqsubseteq \mathsf{D}$ aus der Wissensbasis folgt.

*Klassendisjunktheit* Zwei Klassen $\mathsf{C}, \mathsf{D}$ sind genau dann disjunkt, wenn $\mathsf{C} \sqcap \mathsf{D} \equiv \bot$ aus der Wissensbasis folgt. Anders ausgedrückt: Wenn der Fakt folgt, dass es keine Individuen geben kann, die sowohl $\mathsf{C}$ als auch $\mathsf{D}$ angehören.

*Globale Konsistenz* Eine Ontologie ist *global konsistent*, wenn sie widerspruchsfrei ist.

*Klassenkonsistenz* Eine Klasse $\mathsf{C}$ ist genau dann konsistent, wenn $\mathsf{C} \not\equiv \bot$ aus der Wissensbasis folgt.

*Instanzprüfung* Ein Individuum $\mathsf{a}$ gehört zur Klasse $\mathsf{C}$, wenn $\mathsf{C}(\mathsf{a})$ logische Konsequenz der Wissensbasis ist.

*Klasseninstanzen* Auffinden aller Individuen einer Klasse geschieht durch Instanzprüfung aller bekannten Individuen.

Für weitere Informationen zu Inferenzmaschinen (englisch: *Reasoner*) und -Mechanismen sei der Leser auf die Literatur verwiesen [53–55].

### 2.1.5 Linked Data

Das Linked Data (LD) Konzept greift die zuvor vorgestellten Sprachen und Formate auf, um Daten zu verknüpfen und so unter anderem besser auffindbar zu machen [56; 57]. Im Jahr 2006 veröffentlichte Berners-Lee eine Menge von Regeln, die bei der Veröffentlichung von Daten im Web zu beachten sind [56]:

1. Nutzung von URIs als Namen für Dinge
2. Nutzung von Hypertext Transfer Protocol (HTTP) URIs, damit diese Namen nachgeschlagen werden können
3. Schlägt jemand eine URI nach, sollten nützliche Informationen bereitgestellt werden, unter Verwendung von Standards wie RDF und SPARQL
4. Inklusion weiterer Links zu anderen URIs, damit weitere Dinge entdeckt werden können

Diese Regeln sind als die *LD Prinzipien* bekannt. Ein wichtiges Prinzip von LD ist, dass Inhalte im Web nicht einfach nur verknüpft werden, sondern dass RDF genutzt wird, um typisierte Aussagen zu machen, die beliebige Inhalte verbinden. Ebendiese Typangabe von Links ermöglicht die Spezifikation von semantischen Relationen. In Verbindung mit LD gibt es auch die Bestrebung, einige Daten für jeden frei verfügbar zu machen, ähnlich dem Open Source Gedanken. Sind die LD Daten *offen*, so spricht man auch spezieller von Linked Open Data (LOD). Weitere Informationen zu LD finden sich in der Literatur [58].

## 2.2 Design for Automotive

In diesem Abschnitt werden relevante Design for Automotive (DfA) Grundlagen beschrieben. Zunächst werden Methoden zur Entwicklung von automobilelektronischen Systemen sowohl unter Berücksichtigung von funktionalen als auch Umgebungslasten, sowie zur Sicherstellung von deren Robustheit, ebenfalls unter Berücksichtigung der Applikation, in den Abschnitten 2.2.1 und 2.2.2 angeführt. Diesen folgt eine kurze Beschreibung eines Austauschformats für Anforderungen in Abschnitt 2.2.3. Den Beschreibungen der Entwicklungsmethoden und dem Austauschformat für Anforderungen folgen Grundlagen zu Modellen in der Entwicklung im Abschnitt 2.2.4. Schließlich werden im Abschnitt 2.2.5 grundlegende, für diese Arbeit bedeutsame Informationen zur Analyse der Auswirkungen von Änderungen angeführt.

### 2.2.1 Robustness Validation

Als Robustness Validation (RV) wird der Prozess bezeichnet, den Nachweis zu erbringen, dass ein Produkt dessen beabsichtigte Funktion(en) mit ausreichender Marge der Robustheit unter einem definierten MP für die spezifizierte Lebensdauer erfüllt [10]. Als *Robustheit* eines Produkts bezeichnet man in diesem Zusammenhang dessen Ausprägung der Eigenschaft, unter den im MP spezifizierten, unterschiedlichen Bedingungen, vorgesehene Funktionen zuverlässig erbringen zu können. Durch diesen Bezug zu MPs, stellt die RV eine Methode zur Qualifizierung dar, welche tatsächliche Bedingungen berücksichtigt, denen das zu untersuchende Produkt in dessen Lebenszyklus ausgesetzt ist. Weiterhin werden in der RV Fehlermechanismen durch jeweils geeignete spezifische Maßnahmen adressiert. Siehe Abbildung 2.6 für eine Übersicht über den Ablauf der RV.

Die RV ist nicht nur auf die Automobilbranche beschränkt [59]. Sie steht im Unterschied zum bisherigen, etablierten Standard *Q100* [60] des Automotive Electronics Council (AEC), welcher für die Qualifikation von Automobilelektronik eingesetzt wird: Der Standard AEC Q100 definiert Stresstests durch Angaben zu Stichproben- und Chargengrößen, Akzeptanzkriterien, Methodiken zur Durchführung der Tests, sowie zu den Bedingungen, unter welchen die jeweiligen Stresstests durchzuführen

sind und für welche Komponenten die Tests durchzuführen sind. Stress, dem die zu testenden Komponenten nach diesem Qualifikationsverfahren ausgesetzt werden, ist dabei ohne eine Berücksichtigung der tatsächlichen Bedingungen, denen die Komponente in ihrem Lebenszyklus ausgesetzt ist, allgemein definiert. Weiter zielt in den Stresstests definierter Stress auf die gemeinsame Provokation einiger bekannter Fehlermechanismen ab – der Standard enthält keine Definitionen von spezifischen Stresstests für einzelne Fehlermechanismen. Insgesamt hat daher die RV im Vergleich das Potenzial, genauere und verlässlichere Aussagen zur Robustheit von automobilelektronischen Systemen zu treffen.

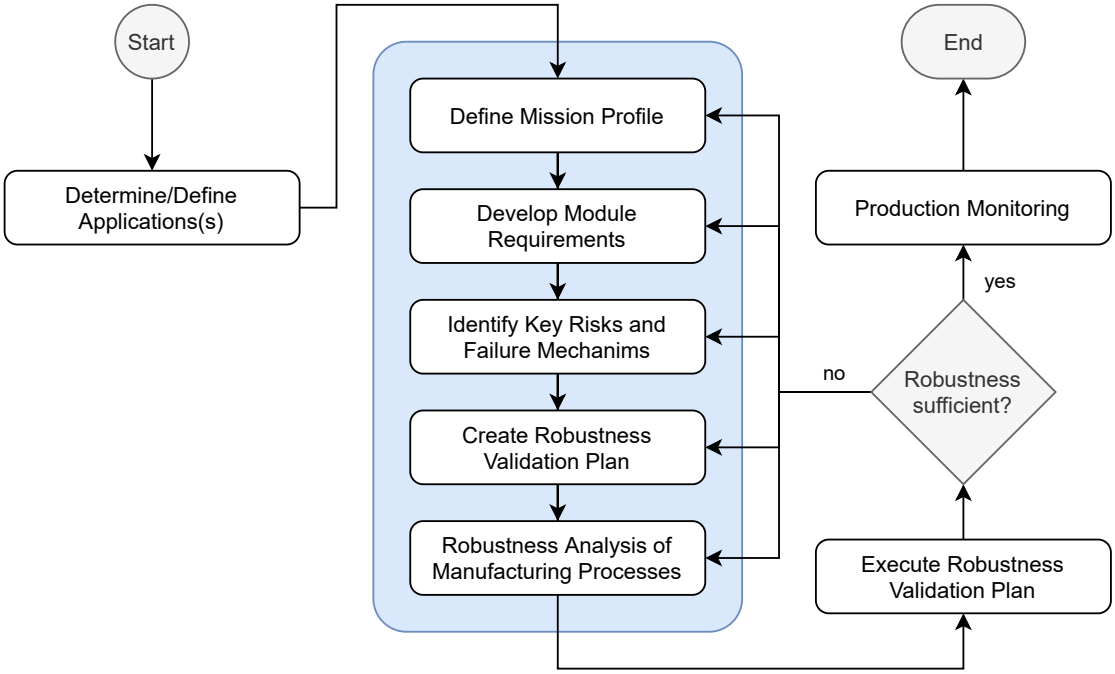



**Abbildung 2.6** – Darstellung des Ablaufs der Robustness Validation, adaptiert aus [10]

### 2.2.2 Mission Profiles

Definition

Mission Profiles (MPs) „definieren den applikationsspezifischen Kontext einer bestimmten Komponente“ (aus dem Englischen) [17]. Dieser applikationsspezifische Kontext kann durch die Spezifikation verschiedener Informationsquellen ausgedrückt werden, siehe Abbildung 2.7. MPs sind vereinfachte Darstellungen sämtlicher relevanter Bedingungen, welchen eine Komponente über ihren gesamten Lebenszyklus ausgesetzt ist [10]. Durch die Spezifikation aller Betriebsbedingungen ermöglichen MPs somit die frühzeitige und präzise Identifikation von die Robustheit einer Komponente einschränkenden Faktoren [16].

MP Format

Es sind nicht nur Vollständigkeit und Präzision von Anforderungen wichtig für den Erfolg von Entwicklungsprozessen, sondern auch deren effizienter Transport und Austausch. Bislang existiert noch kein Standard für den Austausch und die Verarbeitung von MPs. Im Projekt ResCar 2.0[6] wurde ein erster Entwurf für ein Format zur Beschreibung von MPs entwickelt, welches in [17] näher beschrieben wird. Der Entwurf wurde außerdem im Projekt autoSWIFT[7] weiterentwickelt. Zum Zeitpunkt dieser Arbeit soll zudem ein weiteres Projekt ins Leben gerufen werden, welches die Entwicklung und Etablierung eines Standards für MPs zum Ziel hat. Die aktuelle Version des Formatentwurfs, mit dem Namen „Mission Profile Format“ (MPFO)[8], sowie dessen ausführliche Dokumentation ist mittlerweile online abrufbar. Diese ersten Versionen des Mission Profile Format (MPFO) basieren auf XML und werden durch XML Schema Definition (XSD) Dateien definiert.

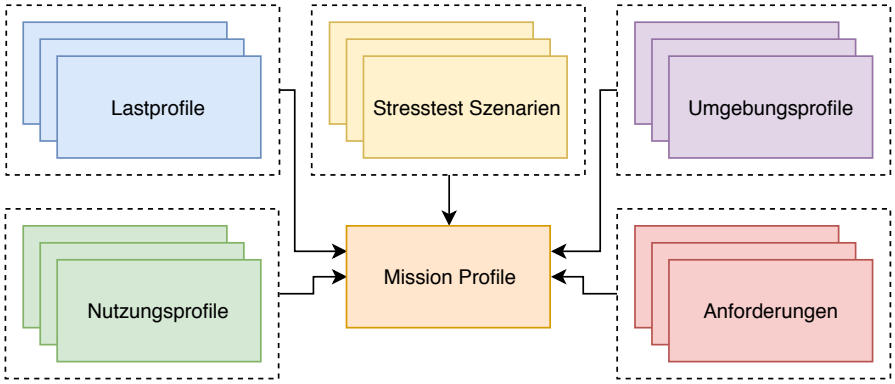



**Abbildung 2.7** – Komposition der Informationsquellen eines MPs

### 2.2.2.1 Mission Profile Aware Design

Motivation

Das Konzept des Mission Profile Aware Design (MPAD) wurde zunächst von Jerke und Kahng in [16] vorgestellt. Danach ist MPAD ein allgemeiner Entwurfsansatz, der die konsequente Berücksichtigung von MPs in den Vordergrund stellt. Dem Konzept nach lassen sich Teile des in Abschnitt 2.2.1 beschriebenen Ablaufs der RV – ein vor allem in der Automobilindustrie eingesetzter und bislang manueller Produkt-Qualifikationsprozess – automatisieren, was auch in dieser Arbeit gezeigt werden konnte, siehe Kapitel 5. Die Automatisierung wird durch die Formalisierung der Generierung, Transformation, Propagierung und Nutzung von MPs, auf den jeweiligen Ebenen in der Lieferkette zur Implementierung und Validierung von Komponenten ermöglicht. Weiterhin formalisiert MPAD den Kommunikationsablauf

[6] Siehe `https://www.edacentrum.de/rescar/`
[7] Siehe `https://www.edacentrum.de/autoswift/`
[8] Siehe `http://www.mpfo.org/`

in der Lieferkette. Insgesamt bewirkt es eine Verringerung von Unsicherheiten in der Entwicklung und hilft bei der Entwicklung von Komponenten, welche sich auf noch unausgereifte Technologien oder auf Technologien mit geringem Spielraum für Parameterveränderungen, stützen. Der grundlegende Ablauf im MPAD wird in Abbildung 2.8 dargestellt. Ausgangspunkt ist die Ableitung eines initialen Modul-MP auf Original Equipment Manufacturer (OEM) Ebene über die Erhebung von Testdaten durch Messungen von umgebungsbedingten Stressoren wie der Temperatur.

Ablauf

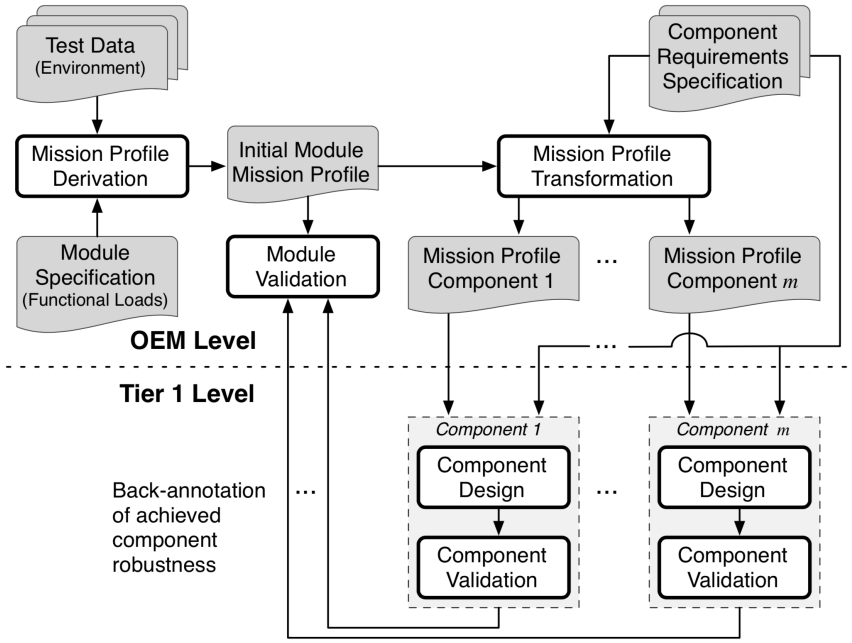



**Abbildung 2.8** – Allgemeiner Ablauf des MPAD, aus [16]

Anschließend werden dem MP funktionale Lasten wie elektrische Spannung hinzugefügt. Das initiale Modul MP wird dann durch Transformation auf einzelne komponentenspezifische MPs heruntergebrochen, welche dann zur Implementierung, Verifikation und Validierung der jeweiligen Komponenten genutzt werden. Zuletzt werden die Ergebnisse der Validierungen der einzelnen Komponenten an das nächsthöhere Modul weitergegeben. Der Prozess aus Transformation, Entwurf und Validierung endet, wenn alle Subkomponenten validiert wurden.

### 2.2.3 Austauschformat für Anforderungen

Das Requirements Interchange Format (ReqIF oder auch RIF[9]) [61] ist ein Austauschformat für Anforderungen, das von der Object Management Group (OMG) standardisiert wurde. Der Standard umfasst ein durch XSD ausgedrücktes Metamodell für Modelle von Anforderungskatalogen. Diesem Metamodell entsprechende XML-Dateien können zwischen Werkzeugen für die Anforderungsverwaltung ausgetauscht werden. Außerdem definiert der Standard den Ablauf des Austauschs, der Status von Anforderungen zwischen dem Anforderungsgeber und dem Empfänger der Anforderungen. Das Dateiformat ermöglicht neben den Anforderungen auch Metadaten der Anforderungen beziehungsweise der Anforderungskataloge sowie weitere Dateien zu transportieren. Jedes Element, das in einem Anforderungskatalog im ReqIF-Format enthalten ist, wird durch einen Globally Unique Identifier (GUID) eindeutig identifiziert, was ebenenübergreifende Relationen zwischen den Elementen ermöglicht.

### 2.2.4 Modelle in der Entwicklung

Modellbegriff

In dieser Arbeit wird der allgemeine Modellbegriff nach der Definition von Stachowiak verwendet [62]. Demnach wird hier ein *Modell* als vereinfachte Repräsentation von etwas verstanden. Ein Modell existiert weiterhin zu einem bestimmten Zweck und bezieht sich auf eine bestimmte Zeit, in der das Modell seine Funktion erfüllt. Nach Stachowiak lassen sich Modelle in grafische, technische und semantische Modelle einteilen. In dieser Arbeit vorkommende Modelle sind respektive symbolisch-grafischer, physikotechnischer und extern-semantischer Art.

Einsatz von Modellen

Modelle spielen eine Schlüsselrolle in vielen wissenschaftlichen Kontexten und in den Ingenieurswissenschaften insbesondere durch deren Möglichkeit zu entwickelnde Systeme zu beschreiben [63; 64]. Neben diesem primären Einsatzzweck lassen sich Modelle auch verwenden, um die Umgebungen von zu entwickelnden Systemen zu beschreiben, was für diese Arbeit in Hinsicht auf MPs von Bedeutung ist: Ein MP wird hier als Modell der funktionalen und Umgebungslasten verstanden, siehe Abschnitt 2.2.2. Weiter stellt ebenso jede Ontologie auch ein Modell dar [65].

Modelle in Entwicklungsmethoden

Der Begriff Model-Driven Engineering (MDE) wird in der Literatur vorrangig in Bezug auf die Entwicklung von Softwaresystemen verwendet [63; 66–69]. Prominentestes Beispiel für einen MDE Ansatz ist der Ansatz der OMG mit dem Namen Model-Driven Archicture (MDA) [70]. MDA gilt als der bekannteste MDE Ansatz [71] und ist in Hinsicht auf Transformationen von Bedeutung, worauf in Abschnitt 2.2.4.1 eingegangen wird. In dieser Arbeit werden Entwicklungsmethoden dann

[9] In dieser Dissertation wird ausschließlich ReqIF als Abkürzung für das Requirements Interchange Format verwendet, da der ebenfalls verwendete Begriff Robustness Indicator Figure auch mit RIF abgekürzt wird.

als *modellgestützt* bezeichnet, wenn Modelle bei der Entwicklung von Hard- oder Software-Systemen eingesetzt werden. Es gibt diverse verwandte Namen für modellgestützte Ansätze: Model-Driven Development (MDD) [72], Model-Driven Software Development (MDSD) [63] oder auch Model-Driven Software Engineering (MDSE) [63]. Diese Bezeichnungen werden in der Literatur teilweise synonym verwendet, wobei es jedoch Unterschiede zwischen ihnen gibt und auch Abgrenzungen der Begriffe existieren [68; 73; 74].

Produktivitätssteigerung

MDE kann die Produktivität in der Entwicklung steigern [63; 75] und wurde erfolgreich in verschiedenen Industriebranchen, wie auch in der Automobilbranche, eingesetzt [76; 77]. Ältere Untersuchungen berichteten zunächst sowohl von einer möglichen Produktivitätssteigerung durch MDE, als auch von einer Abnahme der Produktivität, was auf unausgereifte Werkzeugen zur praktischen Anwendung von MDE zurückgeführt wurde [78]. Jüngeren Studien zufolge wurden Produktivitätssteigerungen von 20-30 % in Unternehmen durch den Einsatz von MDE berichtet, wobei ein Großteil der erfolgreichen Anwendungen von MDE unter dem Einsatz von Domain-Specific Modelling (DSM) geschah [79]. In Bezug auf MDD oder MDA wurde dies durch eine Untersuchung bestätigt, welche von Produktivitätssteigerungen bis zu 40 % und unter Einsatz von DSM 500 %-1000 % berichtet [80]. MDE unterstützt auch das Software-Produktlinien-Engineering [81]. Was die praktische Arbeit mit Modellen betrifft, existiert eine Vielzahl von Modellierungssprachen und Werkzeugen. So gaben in einer Untersuchung zum Stand der MDE Praxis aus dem Jahr 2014, Entwickler mehr als 40 Modellierungssprachen und 100 Werkzeuge an, die sie regelmäßig verwenden [79].

Entwicklung eingebetteter Systeme

Auch in Hinsicht auf die Entwicklung von Software eingebetteter Systeme war bereits im Jahre 2009 ein Trend zu modellgestützten Entwicklungsansätzen absehbar [82]. Für die Entwicklung eingebetteter Systeme ist modellgestützte Entwicklung nicht nur von ökonomischer Bedeutung – es werden darüber hinaus weitere Vorteile, sowohl für die Industrie, als auch die Wissenschaft erwartet [83]. Speziell in Bezug auf automobilelektronische Systeme ist die Entwicklungsmethode MBSE eine wichtige modellgestützte Entwicklungsmethode [12]. MBSE Prozesse und Anforderungen sind im Standard ISO/IEC/IEEE 15288 *Systems and software engineering — System life cycle processes* beschrieben. Komplementär werden Praktiken und Prozessaktivitäten des MBSE im *Systems Engineering Handbook* [84] beschrieben. Die gebräuchliche Definition des International Council on Systems Engineering (INCOSE)[10] von MBSE [85] lautet, aus dem Englischen:

> „MBSE ist die formalisierte Anwendung von Modellierung zur Unterstützung von Systemanforderungsanalyse, -Entwurf, -Analyse, -Verifikation und -Validierung, beginnend in der konzeptuellen Entwurfsphase und fortgesetzt durch die Entwicklung und späteren Lebenszyklusphasen."

[10] Siehe `https://www.incose.org/`

Die im Jahre 2007 definierte Vision *MBSE 2020* des INCOSE ist eine integrierte Modellverwaltung von für die Entwicklung relevanten Modellen, in einem verteilten Modell-Verzeichnis, welches sicheren und zuverlässigen Datenaustausch ermöglicht und auf dem *Publisher-Subscriber* Entwurfsmuster basiert. Ein Übergang zu MBSE ist durch Substitution von SE-Dokumenten durch Modelle gekennzeichnet. Die in Abschnitt 4 beschriebene Plattform kann als Mittel zur teilweisen Ermöglichung dieses Übergangs betrachtet werden, da sie Komponenten für die Abbildung von MP- und Anforderungs-Dokumenten auf ontologische Modelle enthält.

#### 2.2.4.1 Modelltransformation

Klassifikationen

Für modellgestützte Entwicklungsansätze wie MDE oder MBSE sind Modelltransformationen von großer Bedeutung [86]. Nach der Literatur lassen sich Modelltransformationen in drei Hauptkategorien einteilen: Model-to-Model (M2M)[11], Model-to-Text (M2T) und Text-to-Model (T2M) [87–89]. Teilweise wird auch von Model-to-Code Transformationen gesprochen, die aber als Spezialfall von M2M Transformationen aufgefasst werden können [87]. In dieser Arbeit werden ausschließlich M2M Transformationen behandelt. Neben den genannten Hauptkategorien existieren viele weitere, feingranulare Klassifikationen von Transformationen [87; 88; 90–93]. Häufig ist eine weitere Unterteilung in relational beziehungsweise deklarative, imperative beziehungsweise operationale, graph-basierte und hybride Ansätze. Eine etwas jüngere Arbeit klassifiziert Modelltransformationen über deren Absichten [94]. Für einen weiterführenden Überblick über Klassifikationen von Modelltransformationen sei der Leser auf Abschnitt 7 in letzterer Arbeit verwiesen.

Anforderungen an Modelltransformationsansätze, speziell in Bezug auf, OMG Meta Object Facility (MOF) Query/View/Transformation (QVT) wurden in [95] ermittelt. Mittlerweile existiert eine Vielzahl an Transformationssprachen- und Werkzeugen, wie eine aktuelle Übersicht über 60 Transformationswerkzeuge zeigt [89]. Es gibt Autoren, die argumentieren, dass für den Fall von einfachen Transformationen der Einsatz spezieller Transformationssprachen und -werkzeuge sehr aufwendig ist und dass dies oft auch mittels einfacher, imperativer Programmiersprachen realisiert werden kann [96].

### 2.2.5 Change Impact Analysis

Änderungsverwaltung

Im Jahr 1979 formulierte Lehman Gesetze der Software-Evolution [97]. Das erste dieser Gesetze ist das *Gesetz ständiger Veränderung* (aus dem Englischen: *Law of continuing change*). Software unterliegt nicht nur während der Entwicklungsphase

[11] Es wird auch die Abkürzung Model-to-Model Transformation (MMT) für diese Art von Transformationen verwendet, um diese vom Begriff *Machine-to-Machine* abzugrenzen

ständigen Änderungen. Diese ergeben sich zum Beispiel aus geänderten Anforderungen, was insbesondere bei iterativen Entwicklungsmethoden im Rahmen von Verfeinerungen üblich ist. Es gibt viele weitere Gründe für Änderungsanfragen. Um zu verhindern, dass ein System seine Nützlichkeit verliert, sind Anpassungen vorzunehmen, was ein Grund für die Notwendigkeit der Änderungsverwaltung ist. Für die Änderungsverwaltung ist wiederum die Change Impact Analysis (CIA) von Bedeutung, da sie die Auswirkungen beziehungsweise den Einfluss von Änderungen untersucht. Änderungsverwaltung und die damit verbundene CIA ist in der Entwicklung sicherheitskritischer Systeme von Bedeutung und wird von Standards wie der ISO 26262 [98] in der Automobilbranche und weiteren in anderen Domänen vorgeschrieben [99].

Definitionen

Eine gebräuchliche Definition von CIA stammt von Bohner und Arnold [100]:

> „Identifying the potential consequences of a change, or estimating what needs to be modified to accomplish a change."

Nach dieser Definition soll CIA Konsequenzen einer Änderung ermitteln und abschätzen, welche Modifikationen notwendig sind. Eine weitere, weniger gebräuchliche Definition von Pfleeger und Atlee [101] lautet:

> „The evaluation of the many risks associated with the change, including estimates of the effects on resources, effort, and schedule."

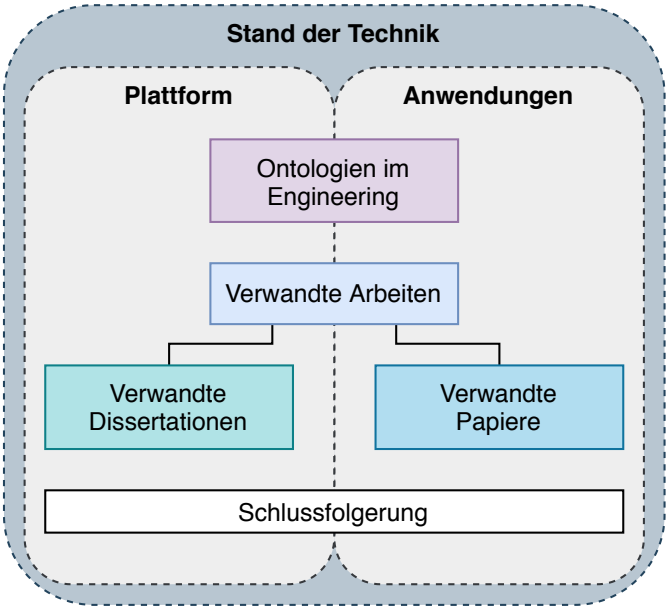
Beiden Definitionen gemein ist, dass Auswirkungen von Änderungen bei der CIA Gegenstand der Untersuchungen sind. Die CIA ist historisch aus dem Software-Engineering gewachsen. Daher bezieht sich die Literatur unter Verwendung des Begriffs CIA zum großen Teil auf die Analyse des Einflusses von Änderungen an Software. Aus diesem Grund liegt der Fokus existierender Arbeiten zur CIA auch meist auf Quelltext (65 % nach Lehnert in [102]). Es ist allerdings anzumerken, dass CIA Techniken sich auch in der Hardwareentwicklung einsetzen lassen, da verschiedene Abstraktionsgrade möglich sind und daher nicht nur Programmiersprachen, sondern auch Hardwarebeschreibungssprachen betrachtet werden können. In dieser Arbeit wird der Begriff CIA weit gefasst verwendet und schließt auch die Analyse der Auswirkungen von Änderungen auf ganze Systeme bestehend aus Hard- und Software mit ein.

Klassifikationen und Anwendungen

Es existieren zwei bekannte Taxonomien für Software CIA [103; 104]. Nach Li et al. wird zwischen vier grundlegenden Perspektiven der Software CIA unterschieden: traditionelle Abhängigkeitsanalyse, Software Repository Mining, Kopplungsmessung und Sammlung von Ausführungsinformationen [105]. Üblicherweise erfolgt einerseits eine Analyse möglicher Auswirkungen einer Änderungsanfrage und für den Fall, dass die Anfrage akzeptiert wird, erfolgt andererseits eine Analyse der Modifikationen, welche die Änderung erforderlich macht [106].

# 3 Stand der Technik

In diesem Kapitel wird der Stand der Technik zum Zeitpunkt dieser Arbeit dargestellt. Das Kapitel ist in zwei Abschnitte unterteilt. Im ersten Abschnitt 3.1 wird allgemein über die Anwendung von Ontologien im Engineering berichtet, während im zweiten Abschnitt 3.2 verwandte Arbeiten behandelt werden. Dieser zweite Abschnitt ist zudem in zwei Unterabschnitte unterteilt, wobei sich Abschnitt 3.2.1 auf verwandte Dissertationen und Abschnitt 3.2.2 auf weitere verwandte Papiere in Bezug auf die Applikationen der Plattform, welche in den Kapiteln 5, 6 und 7 beschrieben werden, beziehen.




## Abschnitte

## 3.1 Anwendung von Ontologien im Engineering

Motivationen

Übliche Motivationen für den Einsatz von Ontologien im Engineering, sind adaptiert nach El Kadiri et al. [107]:

- Das Problem semantischer Interoperabilität in Umgebungen, in denen Systeme, Werkzeuge und Datenquellen keine gemeinsame Erkennung der Datentypen und -Beziehungen haben,
- Bedarf einer gemeinsamen und vertrauenswürdigen Quelle für Wissen des Engineerings, welches sowohl von Menschen, als auch von Maschinen genutzt werden kann
- Aufwertung von Wissen des Engineerings durch Inferenzmechanismen
- Visualisierung von verknüpften Informationen und Daten, welche in Ontologien abgelegt sind
- Separierung von Domänenwissen von proprietären Systemen durch Ontologien erlaubt schnelle Modifikationen von Applikationen, ohne Änderung des Quelltextes, wenn sich Geschäftsbedingungen ändern.

Software Engineering

Happel und Seedorf präsentierten einen Überblick und ein Rahmenwerk zur Klassifikation von Anwendungen von Ontologien im Software-Engineering [108]. Demnach gibt es mindestens folgende Einsatzmöglichkeiten für Ontologien im Software-Engineering:

- Analyse und Entwurf
  - Requirements Engineering
  - Komponenten-Wiederverwendung
- Implementierung
  - Integration mit Software-Modellierungssprachen
  - Ontologie als Domänen Objekt Modell
  - Quelltext Dokumentation
- Bereitstellung und Laufzeit
  - Semantische Middleware
  - Geschäftsregeln
  - Semantic Web Dienste
- Wartung
  - Projekt Unterstützung
  - Aktualisierungen
  - Testen

Nach dieser Arbeit lassen sich weiterhin vier Nutzungskategorien von Ontologien im Software-Engineering aufstellen, siehe auch Abbildung 3.1: Klassifikation

- **Ontology-driven Development (ODD)** Nutzung von Ontologien zur Entwicklungszeit, welche die Problemdomäne beschreiben
- **Ontology-enabled Development (OED)** Ebenfalls Nutzung von Ontologien zur Entwicklungszeit, jedoch zur Unterstützung von Entwicklungsaufgaben
- **Ontology-based Architectures (OBA)** Nutzung von Ontologien zur Laufzeit. Die Ontologie stellt einen zentralen Teil der Anwendungslogik dar
- **Ontology-enabled Architectures (OEA)** Nutzung von Ontologien zur Bereitstellung von Infrastruktur zur Laufzeit

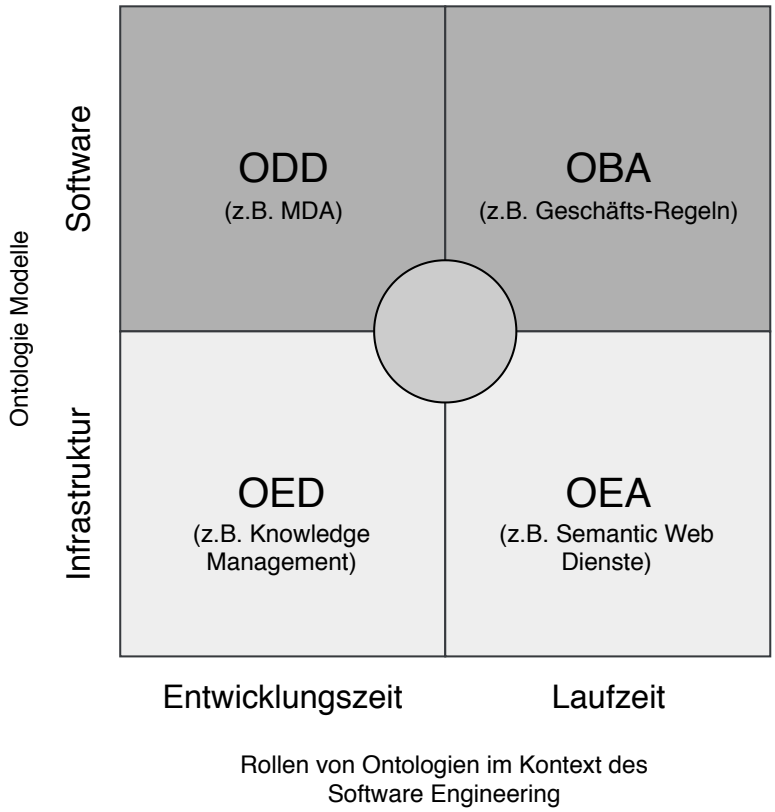



**Abbildung 3.1** – Nutzungskategorien für Ontologien im Software-Engineering, adaptiert aus [108]

Ein Beispiel für einen ODD Ansatz ist die Arbeit von Knublauch [109]. Diese Arbeit präsentiert eine mögliche Softwarearchitektur und Entwicklungsmethode für

Semantic Web Dienste und Agenten. Dies geschieht unter Einsatz des populären Ontologie-Editors und -Anwendungs-Rahmenwerks Protégé[1]. Als Vertreter aus OED ist beispielsweise das System *Dhruv* von Ankolekar anzuführen [110]. Dhruv ist ein System, das SWT nutzt, um Problemlösungsprozesse in Open-Source-Software Gemeinschaften zu unterstützen. Im Zusammenhang mit ODD stellten Bossche et al. den *Ontology-driven Architecture for Software-Engineering (ODASE)* Ansatz für Ontologie-getriebenes Software-Engineering vor [111]. Diese Arbeit beschreibt eine Herangehensweise und die *Hedwig Plattform*, welche aus Werkzeugen und Bibliotheken besteht, um OWL Ontologien in Applikationen zu integrieren.

Allgemein lassen sich, neben der zuvor vorgestellten Klassifizierung, nach Jasper et al. Ontologie-basierte Anwendungen in vier Kategorien einteilen [112]. Nach dieser Klassifikation, welche die Anwendungen nach Szenarien klassifiziert, lässt sich die in Kapitel 4 vorgestellte Plattform selbst in die Kategorie *Common Access to Information* einordnen, genauer *Data Access via Shared Ontology*. Die in den Kapiteln 5, 6 und 7 beschriebenen Anwendungen fallen ebenfalls in diese Kategorie.

Requirements Engineering

Es existiert eine Vielzahl diverser Anwendungen von Ontologien im Engineering. Zur Übersicht über den Einsatz von Ontologien alleine im Requirements Engineering beispielsweise sei der Leser auf [113; 114] verwiesen. Wesentlich sind darin Bestrebungen, Ontologien zu nutzen, um Anforderungen einerseits und Domänenwissen der Anwendung andererseits, formell darzustellen. Siegemund et al. präsentierten dazu in [115] ein Metamodell für Ontologie-gestütztes, Ziel-orientiertes Requirements Engineering.

Weitere Gebiete

Ait-Ameur et al. stellten die auf Engineering-Daten spezialisierte *OntoDB* Datenbank vor [116] und stellten fünf verschiedene Anforderungen an Ontologien im Engineering auf. In Bezug auf Computer-Aided Production Engineering (CAPE) stellten Morbach et al. die wiederverwendbare Ontologie *OntoCAPE* vor [117]. Van Ruijven präsentierte eine Ontologie für SE in [118], welche den Standard *ISO 15926* nutzt, um Prozesse aus dem Standard *ISO 15288* zu modellieren. Eine weitere Untersuchung zur Entwicklung von SE Ontologien ist in der Arbeit [119] von Sarder et al. zu finden. Im MBSE und der Raumfahrtdomäne nutzten Henning et al. Ontologien, um ein Systemmodell für die Integration unterschiedlicher Modellierungsparadigmen und Modell-Semantiken zu etablieren [120; 121]. Ernadote präsentierte in [122] einen Ansatz, um MBSE durch die Kombination von Standard Metamodellen mit Ontologien zu unterstützen und dadurch das Verstehen und die Erstellung von Modellen zu vereinfachen. Dieser Ansatz verbirgt die Komplexität von Basis-Metamodellen und lässt Benutzer stattdessen mit Ontologie-basierten Vokabularien arbeiten. Diese auf Kategorientheorie basierenden Vokabularien unterstützen das Lesen und Erstellen von Modelldaten.

[1] Siehe `https://protege.stanford.edu/`

## 3.2 Aufstellung relevanter Arbeiten

Dieser Abschnitt umfasst zwei Unterabschnitte, die sich auf relevante Dissertationen, beziehungsweise auf relevante Papiere in Bezug auf Plattform-Applikationen beziehen. Gründe für diese Separation liegen einerseits in der Tatsache, dass Dissertationen üblicherweise von größerem Umfang sind als einzelne Papiere und andererseits darin, dass die Konzepte der in Kapitel 4 beschriebenen semantischen MPAD Plattform und der Applikationen, welche in den Kapiteln 5, 6 und 7 beschrieben werden, jeweils unterschiedlich sind.

### 3.2.1 Dissertationen

Übersichtstabelle

Eine Übersicht über relevante Dissertationen ist in Tabelle 3.1 zu finden. Die Arbeiten sind nach acht verschiedenen Kriterien eingeordnet. Zugehörige Ausprägungen zum Kriterium *Grundsatz* beschreiben, welche grundlegenden Ansätze den entsprechenden Arbeiten zugrunde liegen. Das Kriterium *Domäne* bezieht sich auf die Anwendungsdomäne des jeweiligen Lösungsansatzes. Weiter bezieht sich das Kriterium *Ontologie-Nutzung* auf die zuvor in Abschnitt 3.1 angeführten Kategorien der Nutzung im Software-Engineering, sofern entsprechende Ansätze Ontologiebasiert sind. Unter dem Kriterium *Realisierung* wird jeweils die Umsetzung des Lösungsansatzes angeführt. Weiterhin stehen Einträge unter dem Kriterium *Anforderungen* für Unterstützungsmöglichkeiten des jeweiligen Ansatzes in Hinsicht auf das Requirements Engineering. Zudem stehen Einträge unter dem Kriterium *Modelltransformation* für die jeweils unterstützten Arten von Modelltransformationen, siehe dazu auch Abschnitt 2.2.4.1. Mit dem Kriterium der *Änderungen* werden Möglichkeiten aufgeführt, welche ein jeweiliger Ansatz zur Unterstützung im Umgang mit Änderungen an Entwurfsartefakten oder auch Anforderungen, aufweist. Zuletzt werden unter dem Kriterium *Konsistenzprüfung* jeweils Gegenstände angeführt, zu denen entsprechende Ansätze Maßnahmen zur Sicherstellung der Konsistenz bereitstellen.

| Quelle | Grund-satz | Domäne | Ontologie-Nutzung | Realisierung | Anforder-ungen | Modell-transfor-mationen | Änderungen | Konsistenz-prüfung |
|---|---|---|---|---|---|---|---|---|
| [123] | wissens-basiert | MB | OBA | Rahmenwerk | Spezifikation | M2M | Propagierung von Änderungen an Entwurfsmodellen | Entwurfs-modelle, Anforderungen |
| [124] | wissens-basiert | MT | OBA | Rahmenwerk | Rück-verfolgbarkeit | - | Evaluierung von Änderungen an Entwurfsmodellen | Entwurfs-modelle, Anforderungen |
| [125] | modell-basiert | A | - | Werkzeug | Rück-verfolgbarkeit | M2T | - | Entwurfs-modelle, Anforderungen |
| [126] | modell-basiert | A | - | DSL, IDE | Konsistenz, Spezifikation | M2M, M2T | - | Anforderungen |

**Tabelle 3.1** – Übersicht relevanter Dissertationen. Für Werte des Kriteriums der *Domäne* gelten die Abkürzungen: Automotive (A), Maschinenbau (MB) und Medizintechnik (MT)

Tudorache

Wesentlicher Beitrag der Arbeit [123] von Tudorache ist ein Ontologie-basiertes Rahmenwerk für die Entwicklung von Entwurfsmodellen, deren Konsistenz damit automatisch überprüft werden kann. Darüber hinaus ermöglicht dieses Rahmenwerk eine automatische Propagierung von Änderungen zwischen Entwurfsmodellen und unterstützt somit die Änderungsverwaltung. Die genannte Arbeit von Tudorache zielte auf Optimierung von Produktentwicklungsprozessen im Maschinenbau ab, was in enger Verbindung zur Anwendungsdomäne der in dem hier vorliegenden Dokument dargestellten Arbeit steht. Während die Zielsetzung verwandt ist, unterscheiden sich sowohl die Kontexte als auch die Lösungsansätze beider Arbeiten allerdings in mehreren Punkten. So schließt der Entwurf automobilelektronischer Systeme noch weitere Disziplinen wie das Electrical/Electronic-, Software- und Safety-Engineering, neben dem Maschinenbau, mit ein. Dieser Unterschied äußert sich beispielsweise in der Diversität der Arten von behandelten Entwurfsartefakten und Entwicklungsmethoden, wie auch deren Komplexität als einzelnes oder im Zusammenhang mit weiteren Systemen. Die Arbeit beschreibt mehrere Ontologien, die als Teil des Lösungsansatzes entwickelt wurden. Eine dieser Ontologien ist die *requirements* Ontologie, welche verwendet wird, um Anforderungen formell darzustellen und zur weiteren Verarbeitung dem System zur Verfügung zu stellen. Dies setzt einen Grad an Formalisierung der Anforderungen voraus, welcher in der Praxis bis heute nicht oder nur selten vorkommt [127–131]. Die in Kapitel 4 vorgestellte Plattform berücksichtigt im Unterschied dazu höhere Abstraktionsgrade von Anforderungen: Es wird von Anforderungsbeschreibungen in natürlicher Sprache ausgegangen, welche in einem standardisierten Austauschformat übergeben werden können.

Hagedorn

In seiner Arbeit [124] präsentierte Hagedorn einen Ontologie-basierten Ansatz zur Adressierung von Herausforderungen bei der Entwicklung von Medizintechnik (MT)-Geräten. Zu diesen Herausforderungen zählen oft unüberschaubare Entwicklungsumgebungen, zu denen Entwickler nur beschränkt Zugang haben und mit denen sie nicht vertraut sind. Weiterhin sind die Benutzer der zu entwickelnden Geräte für deren Entwickler nur schwer zugänglich und höchst divers in vielerlei Hinsicht. Zudem weist der MT-Markt eine Struktur auf, welche oft Interessengruppen gegeneinander ausspielt. Der wissensbasierte Ansatz von Hagedorn entgegnet der Problemstellung durch die Bereitstellung eines Wissens-Rahmenwerks, bestehend aus domänenspezifischen Ontologien, die die Ideenfindung beim Design von MT-Geräten, die Erfassung von Wissen und die Entwurfsverifikation unterstützen. Der Ansatz von Hagedorn unterscheidet sich in der Anwendungsdomäne von dieser Arbeit. In Bezug auf die Lösungsansätze gibt es sowohl Gemeinsamkeiten als auch Unterschiede. So kommen in beiden Arbeiten Ontologien zum Einsatz, um Domänenwissen zu formalisieren und den jeweiligen Problemstellungen zu entgegnen. Weitere Gemeinsamkeiten sind die Berücksichtigungen von Anforderungen mit der Möglichkeit der Anforderungs-Rückverfolgbarkeit. Ein bedeutender Unterschied

zwischen den beiden Arbeiten ist jedoch, dass sich die Herangehensweise von Hagedorn auf die Entwicklung und den Einsatz von Ontologien konzentriert, welche mit dem Ontologie-Editor und -Anwendungsframework *Protégé*[2] [132] erstellt und auch über diesen genutzt wurden. Der Unterschied dazu liegt in der hier vorgestellten Arbeit, auf der konzeptionellen Ebene. Es wurde auf der Basis des Konzepts eine Software-Plattform entwickelt, die Ontologien mit Domänenwissen verarbeitet und erstellt.

Kugele

Einen neuartigen Ansatz für die durchgängige modellbasierte Entwicklung automobilelektronischer Systeme stellte Kugele in [126] vor. Der vorgeschlagene *COLA Automotive Ansatz* folgt der Idee eines *Systemcompilers* und sieht die Verwendung einer Domain-specific Language (DSL) mit dem Namen *COLA* zur Beschreibung von zu entwickelnden Systemen vor. *COLA* bildet das Fundament des Ansatzes und soll in Verbindung mit der *COLA-Integrated Development Environment (IDE)* genutzt werden. COLA-IDE bietet zusätzliche Funktionen, wie die Generierung von Anforderungsspezifikationen aus logischen Architekturbeschreibungen, Modellanalysen und -transformationen, Ausleitung simulierbarer Lastenhefte und Anforderungs-Rückverfolgbarkeit. Sämtliche Entwicklungsartefakte werden zudem zentralisiert in einer Datenbank abgelegt, wodurch Lücken bei der Werkzeugintegration vermieden werden sollen. Während die Problemdomäne der Arbeit von Kugele mit der hier vorgestellten Arbeit übereinstimmt, unterscheidet sich jedoch der modellbasierte Grundsatz vom wissensbasierten Grundsatz dieser Arbeit. In Konsequenz sind die Herangehensweise und resultierende Lösungsansätze unterschiedlich. Der COLA Automotive Ansatz bietet eine durchgängige Entwurfsmethodik und begleitende Werkzeuge. Dagegen stellt die in Abschnitt 4 vorgestellte Plattform eine einheitliche Basis zur Entwicklung von Anwendungen bereit, die in der Entwicklung automobilelektronischer Systeme eingesetzt werden können. In Hinsicht auf die Integration in existierende Prozesse und die Integration von weiteren Entwicklungswerkzeugen bietet der COLA Automotive Ansatz die zentrale Datenbank zur Abbildung des Produktdatenmodells an. Auf diese Weise bereitgestellte Daten können jedoch im Vergleich zum Ontologie-basierten Ansatz dieser Arbeit nicht so einfach mit anderen Systemen ausgetauscht oder von diesen verarbeitet werden. Dies liegt darin begründet, dass in der hier vorliegenden Arbeit ausschließlich standardisierte Sprachen und -formate zur Abbildung von beispielsweise System- oder Anforderungsmodellen zum Einsatz kommen, die den Austausch zwischen heterogenen Systemen nicht nur von Daten, sondern auch deren Semantik ermöglichen.

Reke

Die Arbeit von Reke in [125] stellte einen Ansatz für die Entwicklung automobiler Steuerungssysteme für Kleine und mittlere Unternehmen (KMU) vor. Dieser Ansatz definiert einen Entwicklungsprozess, basierend auf Schnittstellen und Methoden von Prozessen aus großen Unternehmen. In diesem Zusammenhang wird weiterhin

[2] Siehe `https://protege.stanford.edu/`

vorgestellt, dass sich die modellbasierte Entwicklung eignet, selektiv auf für KMU ungeeignete Prozesselemente zu verzichten und solche einzuschließen, die sich eignen. Die Arbeit von Reke stellt darüber hinaus auch ein Werkzeug zur modellbasierten Softwareentwicklung vor. Auch bei dieser Arbeit stimmt die Domäne mit der hier vorgestellten Arbeit überein, wobei die Arbeit von Reke etwas spezifischer auf Steuerungssysteme im Automobil bezogen ist. Eine Gemeinsamkeit ist zudem die Berücksichtigung von Anforderungen im Entwurf mit der Unterstützung für Anforderungs-Rückverfolgbarkeit. In der Herangehensweise unterscheiden sich die beiden Arbeiten allerdings grundsätzlich. Während diese Arbeit einem wissensbasierten Grundsatz folgt, ist die Arbeit von Reke modellbasiert. Wie auch bei der zuvor angeführten Arbeit von Kugele kann bei der Arbeit von Reke von einer durchgängigen Entwicklungsmethodik gesprochen werden, die Entwicklungsprozesse und -werkzeuge definiert. Zwar wird die Unabhängigkeit vom Modellierungswerkzeug berücksichtigt, wodurch ein gewisser Grad an Wiederverwendbar und -teilbarkeit der Modelle gewährleistet wird, jedoch können die Systemmodelle im Vergleich zum Ontologie-basierten Ansatz dieser Arbeit ebenfalls nicht so einfach unter heterogenen Anwendungen beziehungsweise Systemen ausgetauscht werden.

### 3.2.2 Arbeiten in Relation zu Plattform-Applikationen

Reasoning-gestützte Robustenss Validation

In der Robotik-Domäne nutzten Zander und Awad DL-Ontologien, um Eigenschaften von Komponenten auszudrücken, Eigenschaften über zusammenhängende Komponenten zu propagieren und Eigenschaften zu aggregieren, welche die Berechnung von komplexen Eigenschaften erlauben [133]. Die Autoren formulierten entsprechende Axiome in DL-Syntax, wie in Abschnitt 2.1.3.2 vorgestellt. Deren Realisierung in einer Ontologie wurde mittels RDF, RDFS und OWL Ausdrücken (siehe Abschnitte 2.1.2.2, sowie 2.1.2.3 und 2.1.3.3) dargestellt. Die Propagierung von Eigenschaften über die Verwendung von OWL *property chains*, wie in Abschnitt 5.5 beschrieben, ist von dieser Arbeit inspiriert. Jedoch unterscheiden sich sowohl die Domäne, als auch die Anwendung von der in Kapitel 5 vorgestellten Arbeit.

In der Raumfahrtdomäne nutzten Hennig et al. OWL-Ontologien, um ein gemeinsames Systemmodell für die Integration von verschiedenen Modellierungsparadigmen und unterschiedlicher Semantik von Modellen zu etablieren [120; 121]. Dieser Ansatz basiert auf der Beschreibung von Systemmodellen und konzeptuellen Datenmodellen durch Ontologie-Beschreibungssprachen. Im Vergleich zur in Kapitel 5 vorgestellten Arbeit unterscheiden sich die Anwendungsdomäne, der Fokus beider Ansätze, die Anwendung selbst und auch, dass Henning et al. keine MPs berücksichtigen.

Im Bereich des Maschinenbaus schlugen Zhang und Luo einen Ontologie-basierten Ansatz für die Materialauswahl vor [134]. Dieser Ansatz nutzt eine Wissensbasis, welche aus einer Maschinenbau-Ontologie inklusive gekennzeichneten Materialin-

stanzen, sowie SWRL Regeln (siehe Abschnitt 2.1.3.4) besteht. Wenngleich ebenso ein Auswahlverfahren stattfindet, so unterscheiden sich doch die Domäne und die eigentliche Anwendung von der in Kapitel 5 vorgestellten Arbeit. Außerdem werden auch keine MPs berücksichtigt.

Ontologie-gestützte Change Impact Analysis

Wie in Abschnitt 2.2.5 bereits erwähnt, liegt der Fokus von CIA-Ansätzen zum großen Teil auf Quelltexten und anderen Ausdrücken wie natürlicher Sprache. Dagegen schlugen Borg et al. einen CIA-Ansatz vor, mit einem spezifischen Fokus auf Artefakte, die keine Quelltexte sind [99]. Dieser Ansatz stützt sich auf die Analyse von Problem-Berichten. In Bezug auf Hardware-Entwürfe stellten Ring et al. ein CIA-Rahmenwerk vor, welches sich auf die Änderungsverwaltung konzentriert [135]. Dies geschieht durch die Detektion von Beziehungen zwischen Spezifikationen unterschiedlicher Abstraktionsgrade. Wenngleich dieser Ansatz, wie auch die in Kapitel 6 vorgestellte Arbeit, auf Hardware-Entwürfe anwendbar ist, berücksichtigt er jedoch keine Anforderungen oder MPs. Bisher liegt der Fokus bei CIA-Ansätzen zumeist auf funktionalen Änderungen bei Entwurfsartefakten. Deswegen wird vermutlich der Einfluss auf Systemebene unterschätzt, bleibt in der Praxis unbemerkt und führt schließlich zu unvollständigen Analysen [106]. Goknil et al. stellten einen CIA-Ansatz vor, welcher ebenfalls wie die in Kapitel 6 vorgestellte Arbeit Anforderungen in Verbindungen mit Änderungen an Entwurfsartefakten bringt [136]. Dieser Ansatz unterscheidet sich jedoch im Grundsatz von der in Kapitel 6 vorgestellten Arbeit, da er nicht wissensbasiert ist, sich nur auf Software Architekturen bezieht und zudem keine MPs berücksichtigt. Weitere Informationen zu Software CIA-Ansätzen sind in der detaillierten Übersicht von Lehnert zu finden [102].

Ontologie-basierte Transformation von Anforderungen

Eine frühe Arbeit zu Ontologie-basierter Modelltransformationen ist der Ansatz von Roser und Bauer [137]. Dieser Ansatz stellt eine *Zwischenmodell-Transformationssprache* in den Vordergrund, in welcher durch einen Generator anhand von Metamodell-, Ontologie-Definitionen und einer *Transformationskonfiguration*, Transformationen spezifiziert werden. Dabei besteht die Transformationskonfiguration aus *Metamodell-Ontologie-Bindungen*, *Ontologie-Metamodell-Bindungen* sowie entsprechenden *semantischen Transformationen*. Die semantischen Transformationen beschreiben, welche Elemente der Quell-Ontologie auf welche Elemente der Ziel-Ontologie abgebildet werden. Die in der Zwischenmodell-Transformationssprache ausgedrückten Transformationen können dann in konkrete spezifische Transformationssprachen wie QVT [138] oder Atlas Transformation Language (ATL) [139] überführt werden.

Nach einer Arbeit von Wouters und Gervais war zum Zeitpunkt der Veröffentlichung im Jahr 2012 die einzige Möglichkeit OWL 2 Ontologien zu transformieren, durch Übersetzungen der Ontologien in MOF-basierte Modelle gegeben [140]. Dieses Vorgehen geht auch auf vorherige Arbeiten wie der von Geihs et al. zurück [141]. Die Idee bei diesem Vorgehen ist es, vorhandenes Wissen und Arbeiten zum Gebiet der Modelltransformationen zu nutzen. Ontologien sollen dabei als Modelle betrach-

tet und in entsprechende Modellsprachen übersetzt werden, die von existierenden Modelltransformationswerkzeugen verarbeitet werden können. Es gab zu diesem Zeitpunkt bereits jedoch auch andere Ansätze, die nicht MOF-basiert sind, wie der Ansatz von Šváb-Zamazal und Svátek [142; 143].

Der in Kapitel 6 vorgestellte Ansatz zur Ontologie-basierten Transformation von Anforderungen ist ebenfalls nicht MOF-basiert, ermöglicht aber grundsätzlich, um einen solchen Ansatz erweitert zu werden. Der Ansatz sieht die Erweiterbarkeit um weitere Transformationsansätze vor und fokussiert nicht die Erforschung eines neuen Transformationsansatzes für Ontologien, sondern die Integration unterschiedlicher Daten und Prozesse in heterogenen Systemen unter Berücksichtigung verschiedener Spezifika der Anwendungsdomäne. Um die Komplexität der exemplarischen Transformationen im Rahmen der Untersuchungen im Zusammenhang mit dieser Arbeit möglichst gering zu halten, wurde auf den MOF-basierten Ansatz verzichtet. Stattdessen wurden simple Transformationen implementiert. Dieses Vorgehen ist mit dem von Akehurst et al. in [96] vergleichbar, siehe auch Abschnitt 2.2.4.1.

## 3.3 Schlussfolgerung

In Bezug auf die in Abschnitt 1.1 beschriebene Motivation und Zielsetzung werden zunächst Kriterien definiert, welche für einen qualitativen Vergleich des Stands der Technik herangezogen werden. Diese Kriterien setzen sich wie folgt zusammen:

***MPAD*** MPs spielen eine wichtige Rolle im Entwurf automobilelektronischer Systeme, weshalb der Lösungsansatz MPs unbedingt berücksichtigen sollte.

***Anforderungen*** Da MPs auch für die Propagierung von Anforderungen im Entwurf automobilelektronischer Systeme eingesetzt werden, liegt es nahe auch Anforderungen im Lösungsansatz zu berücksichtigen.

***Wissensbasiert*** Dieses Kriterium bezieht sich auf die Beantwortung der Frage, ob sich Schlussfolgerungen, unter Nutzung von Ontologien, auf der Basis von Entwurfsmodellen ziehen lassen. Es beschreibt, ob der jeweilige Ansatz sich auf Ontologien für die Spezifikation der dafür erforderlichen Semantik stützt.

***Standardisierte Schnittstellen*** Nicht nur in Hinsicht auf die Eigenschaft von Ontologien, eine Konsolidierung und Integration von Entwurfsmethoden- und Modellen zu ermöglichen, sondern auch auf die Tatsache, dass eine einsetzbare Lösung in bestehende Infrastruktur und Systeme integriert werden muss, ist der Einsatz von standardisierten Schnittstellen unabdingbar.

***Änderungsverwaltung*** Die Unterstützung der Änderungsverwaltung ergibt sich aus dem Einsatz von MBSE beziehungsweise heute üblichen iterativen und

inkrementellen Entwicklungsmethoden, bei denen mit jedem Inkrement Änderungen an Entwurfsartefakten im Vergleich zur vorherigen Version übernommen werden. Die Änderungsverwaltung sollte auch die Änderungen von Anforderungen berücksichtigen, welche im Laufe der Entwicklung erforderlich sein können.

Aus der Betrachtung des Stands der Technik geht hervor, dass keiner der betrachteten Ansätze das Konzept des MPAD berücksichtigt. Dies trifft auch weitgehend auf die Berücksichtigung von Anforderungen zu, die, sofern sie doch berücksichtigt wurden, jedoch nicht in natürlichsprachlicher Form, sondern als Modelle vorausgesetzt wurden. Von den 13 betrachteten Ansätzen sind außerdem nur drei wissensbasiert beziehungsweise setzen Ontologien zur Lösung der Problemstellung ein. In Hinsicht auf eine einfache Integration der entstandenen Lösung in existierende Entwurfsprozesse setzen nur fünf Ansätze standardisierte Schnittstellen ein. Auch in Bezug auf die Unterstützung der Änderungsverwaltung werden nur in sechs Ansätzen Mittel dazu bereitgestellt. Insgesamt unterstützt keiner der betrachteten Ansätze alle unbedingt geforderten Kriterien. Das Ergebnis des Vergleichs mit dem Stand der Technik ist in Tabelle 3.2 dargestellt.

| | M | A | W | S | Ä |
|---|---|---|---|---|---|
| Tudorache [123] | ✗ | ✓ | ✓ | ✓ | ✓ |
| Hagedorn [124] | ✗ | ✓ | ✓ | ✓ | ✗ |
| Kugele [126] | ✗ | ✓ | ✗ | ✗ | ✓ |
| Reke [125] | ✗ | ✓ | ✗ | ✗ | ✓ |
| Hennig et al. [120; 121] | ✗ | ✓ | ✓ | ✓ | ✓ |
| Zhang u. Luo [134] | ✗ | ✗ | ✗ | ✓ | ✗ |
| Borg et al. [99] | ✗ | ✓ | ✗ | ✓ | ✗ |
| Ring et al. [135] | ✗ | ✗ | ✗ | ✗ | ✓ |
| Goknil et al. [136] | ✗ | ✓ | ✗ | ✗ | ✓ |
| Roser u. Bauer [137] | ✗ | ✗ | ✗ | ✗ | ✗ |
| Wouters u. Gervais [140] | ✗ | ✗ | ✗ | ✗ | ✗ |
| Šváb-Zamazal u. Svátek [142; 143] | ✗ | ✗ | ✗ | ✗ | ✗ |
| Akehurst et al. [96] | ✗ | ✗ | ✗ | ✗ | ✗ |
| Novacek | ✓ | ✓ | ✓ | ✓ | ✓ |

**Tabelle 3.2** – Vergleich mit relevanten Arbeiten in Bezug auf die Kriterien: **M**ission Profile Aware Design, **A**nforderungen, **W**issensbasiert, **S**tandardisierte Schnittstellen, **Ä**nderungsverwaltung

# 4 Semantische Mission Profile Aware Design Plattform

Dieses Kapitel stellt den Lösungsansatz vor, der im Rahmen dieser Arbeit entwickelt wurde. In Hinsicht auf die in Abschnitt 1.1 beschriebene Problematik, dass Entwurfsmodellen oft die für das automatische Schlussfolgern benötigte explizite Semantik fehlt, werden Ontologien als Bestandteil des Lösungsansatzes eingesetzt. Diese werden verwendet, um die Integration von relevanten Daten beziehungsweise Modellen aus heterogenen Systemen sowie das automatische Schlussfolgern zu ermöglichen. Die Kernidee ist, eine einheitliche Basis zu schaffen, auf welcher den Entwurfsprozess unterstützende Anwendungen entwickelt und ausgeführt werden können. Dies geschieht unter konsequenter Berücksichtigung von MPs. Ein Beispiel für eine solche Anwendung ist die in Kapitel 5 näher beschriebene Methode, die einen Teil der Analyse und Sicherstellung der Robustheit von automobilelektronischen Systemen durch Automatisierung unterstützt.

Zu Beginn wird in Abschnitt 4.1 das Konzept zum Ontologie-gestützten Mission Profile Aware Design (MPAD) eingeführt. Darauf beruhend wird dann in Abschnitt 4.2 die Zielsetzung wiedergegeben und anschließend in den Abschnitten 4.3 und 4.4 davon ausgehend Anforderungen abgeleitet. Danach folgt in Abschnitt 4.5 eine Zusammenfassung der übergreifenden Entwicklungsmethodik.

## Abschnitte

## 4.1 Konzept

Das zugrundeliegende Konzept des in Abschnitt 2.2.2.1 vorgestellten MPAD wurde zunächst um den Aspekt des Einsatzes von Ontologien erweitert, was in Abbildung 4.1 dargestellt wird. Dabei findet im Wesentlichen eine zusätzliche, frühe Abbildung des initialen MP auf eine Mission Profile Ontologie (MPO) statt. Der weitere Ablauf ist dann analog zu sehen, wobei die Ableitung von Komponenten-MPs beziehungsweise MPOen durch Transformation dann auf Grundlage der Modul-MPO stattfindet. Das heißt im Ontologie-gestützten MPAD werden weitere MPs unmittelbar nach der Ableitung des initialen MP durch MPOen substituiert.

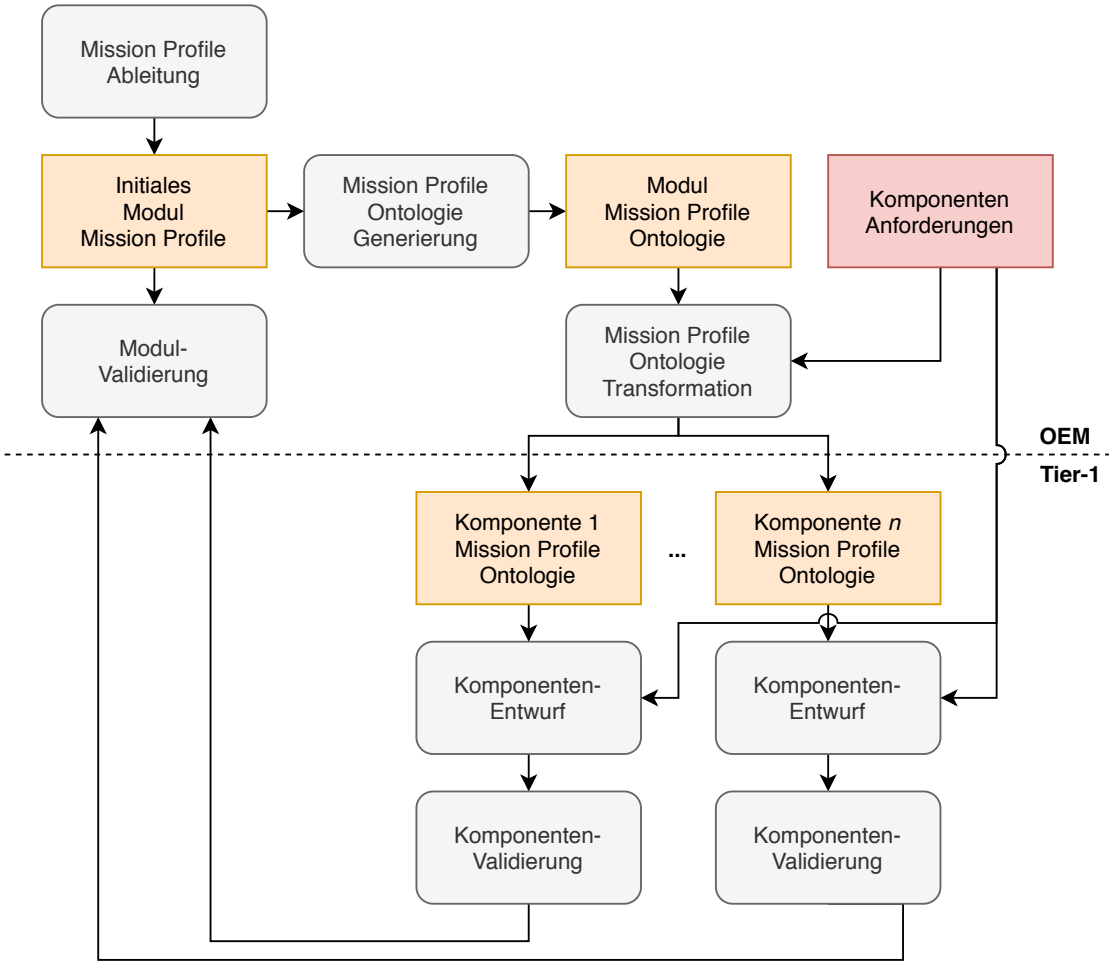



**Abbildung 4.1** – Ablauf des Ontologie-gestützten MPAD

## 4.2 Ziele

Beim Entwurf und der Entwicklung der Plattform zur Realisierung eines Ontologie-gestützten MPAD wurden die folgenden Ziele verfolgt:

(1) Möglichkeit der Integration heterogener Daten beziehungsweise Modelle

Die Plattform soll es ermöglichen, relevante Daten beziehungsweise Modelle aus unterschiedlichen Quellen beziehen und verarbeiten zu können. Korrelierende Bestandteile der Daten beziehungsweise Modelle sollen explizit verknüpft werden können. Die Verknüpfung soll eine vereinheitlichende und simultane Verarbeitung von Daten aus unterschiedlichen Quellen ermöglichen. In Hinsicht auf das Ontologie-gestützte MPAD soll die Integration von MPs und Anforderungen ermöglicht werden.

(2) Automatisches Schlussfolgern

Die Plattform soll Mittel bereitstellen, die das automatische Schlussfolgern auf der Basis integrierter Daten beziehungsweise Modelle ermöglichen. Das automatische Schlussfolgern soll mittels Ontologien umgesetzt werden. Dabei soll eine standardisierte Sprache eingesetzt werden, um die Interoperabilität zu gewährleisten.

(3) Unterstützung von Änderungsverwaltung

In heute üblichen Entwicklungsmethoden wie dem inkrementellen Vorgehensmodell ist eine schrittweise Verfeinerung von Anforderungen vorgesehen. Zudem können aber auch bei anderen Entwicklungsmethoden Änderungen an Anforderungen ebenfalls nach der Anforderungsanalyse notwendig werden. Aus diesem Grund sollte die zu entwickelnde Lösung Mechanismen zur Unterstützung der Änderungsverwaltung bereitstellen.

(4) Ermöglichung der Einbindung in existierende Arbeitsabläufe

Um gebrauchstauglich zu sein, muss sichergestellt werden, dass die entwickelte Lösung auch in existierende Arbeitsabläufe integriert werden kann. So sind üblicherweise in Unternehmen auf Mitarbeitercomputersystemen systemweite Installationen von Fremdsoftware aufgrund von Sicherheitsvorkehrungen nicht möglich. Des Weiteren sind etablierte oder in Entwicklung befindliche Standards für Datenformate und Kommunikationsprotokolle zu berücksichtigen, um eine möglichst direkte Verwendung der Systemlösung zu ermöglichen.

## 4.3 Anforderungen an die Plattform zur Realisierung eines Ontologie-gestützten MPAD

Anhand der im vorherigen Abschnitt 4.2 festgelegten Ziele wurden Anforderungen an die Plattform selbst abgeleitet, welche im Folgenden, in funktionale und nichtfunktionale unterteilt, beschrieben werden:

### 4.3.1 Funktionale Anforderungen an die Plattform zur Realisierung eines Ontologie-gestützten MPAD

P1 Integration heterogener Daten beziehungsweise Modelle über Ontologien

Es sollen Ontologien zur Verarbeitung von Entwurfsmodellen genutzt werden, weshalb die Plattform entsprechende Abbildungsmechanismen von Entwurfsmodellen auf Ontologien bereitstellen soll. In Hinsicht auf MPAD sollen Abbildungsmechanismen von MPs im MPFO- und Anforderungen im ReqIF-Format bereitgestellt werden. Diese Anforderung bezieht sich auf Ziel (1).

P2 Verknüpfung von MP-Daten mit Anforderungen

Das System muss über einen Mechanismus zur Spezifikation von feingranularen Verknüpfungen zwischen MP Daten nach dem MPFO und Anforderungen im ReqIF verfügen. Derart spezifizierte Verknüpfungen sollen die gemeinsame Verarbeitung beider Datenarten ermöglichen. Dies stellt ebenfalls einen Beitrag zur Erreichung von Ziel (1) dar.

P3 Automatisches Schlussfolgern mittels Ontologien

Um automatische Schlussfolgerungen auf der Basis von Entwurfsmodellen durchführen zu können, muss das System entsprechende Mittel dafür bereitstellen. Diese Anforderung ist ein Beitrag zur Erreichung von Ziel (2).

P4 Identifikation von Unterschieden zwischen Eingabe-Modellen

Zur Erreichung von Ziel (3), muss das zu entwickelnde System in der Lage sein, Unterschiede zwischen verschiedenen Versionen von Eingabemodellen feststellen zu können und definierbare Aktionen auf Basis der identifizierten Unterschiede auszuführen. Dies gilt insbesondere für MPs- und ReqIF-Dokumente.

P5 Bereitstellung von resultierenden Ontologien

Um Daten nach deren Verarbeitung abrufen und weitergeben zu können, muss das System diese den Clients bereitstellen. Dabei muss ein sicherer Transport zum Schutz von Intellectual Property (IP) gewährleistet sein. Dies trägt zur Erreichung von Ziel (4) bei.

### 4.3.2 Nicht-funktionale Anforderungen an die Plattform zur Realisierung eines Ontologie-gestützten MPAD

P6 Erweiterbarkeit

In Abhängigkeit von den Anforderungen zur Lösung einer MPAD Aufgabe muss die Plattform um weitere Funktionalität ergänzt werden. Dies trifft auch auf die Realisierung von Änderungsverwaltung mittels der Plattform für einen konkreten Anwendungsfall zu, wobei Bezug auf Ziel (3) hergestellt wird. Genauso können zusätzliche Funktionen oder Eigenschaften bei der Systemintegration erforderlich sein, womit Ziel (4) adressiert wird. Aus diesen Gründen muss die Plattform-Software erweiterbar sein.

P7 Integrierbarkeit

Als Beitrag zur Erreichung von Ziel (4) verlangt diese Anforderung primär eine Ausgestaltung der extern zugänglichen Schnittstellen so, dass diese standardkonforme Kommunikationstechniken verwenden. In Hinsicht auf MPAD sollen MPs im MPFO und Anforderungen im ReqIF Format verarbeitet werden können.

## 4.4 Anforderungen an Anwendungen für die Plattform zur Realisierung eines Ontologie-gestützten MPAD

Zuvor wurden Anforderungen an die Plattform selbst angeführt. Anwendungen, welche wiederum auf der Plattform basieren, sollten weiterhin die hier beschriebenen Anforderungen erfüllen.

### 4.4.1 Funktionale Anforderungen an Anwendungen für die Plattform zur Realisierung eines Ontologie-gestützten MPAD

A1 Unterstützung von MPAD

Wie in Abschnitt 2.2.2.1 beschrieben, spielen MPs eine wichtige Rolle im Entwurfsprozess automobilelektronischer Systeme. Aus diesem Grund sollten Plattform-Anwendungen MPAD unterstützen. In erster Linie bedeutet dies eine konsequente Berücksichtigung von MPs innerhalb der Anwendungen.

A2 Möglichkeit zur Integration heterogener Datenquellen

Auf der Plattform basierende Anwendungen sollten eine Integration heterogener Datenquellen ermöglichen. Diese Anforderung ergibt sich aus der Tatsache, dass bereits eine Berücksichtigung von MPs gefordert wurde und eine sinnvolle Plattform Anwendung höchstwahrscheinlich weitere Datenquellen benötigt, um deren Ziel zu erreichen.

### 4.4.2 Nicht-funktionale Anforderungen an Anwendungen für die Plattform zur Realisierung eines Ontologie-gestützten MPAD

A3 Bereitstellung von Wissen in standardisierter Form

In Hinsicht auf Ziel (4) der Plattform, sowie Anforderung P7, ist es erforderlich, dass Wissen, welches aus einer Verarbeitung durch Plattform-Anwendungen resultiert, in einem standardisierten Austauschformat bereitgestellt wird. Dies ermöglicht dann eine Integrierbarkeit des entsprechenden Ansatzes in existierende Arbeitsabläufe.

## 4.5 Methodik für die Entwicklung mit der Plattform zur Realisierung eines Ontologie-gestützten MPAD

In Hinsicht auf die Entwicklung von Anwendungen, welche auf der Plattform basieren beziehungsweise diese nutzen, wird eine in vier Phasen unterteilte Entwurfsmethode angewendet. Siehe Abbildung 4.2 für einen Überblick über diese Methodik. Grundsätzlich umfassen die Phasen der anwendeten Entwurfsmethode zunächst die *Wissensidentifikation*, *Abbildung des Wissens auf Ontologien*, *Definition von Inferenzmechanismen* sowie der *Definition der Anwendungslogik* unterteilt, welche in den folgenden Abschnitten näher beschrieben werden.

Grundsätzlich gilt, dass die vier Phasen zunächst nacheinander bearbeitet werden. Im Laufe einer iterativen, inkrementellen Entwicklung ist dann jedoch die Reihenfolge der Bearbeitung nicht festgelegt. Im weiteren Verlauf sind dann auch parallele Bearbeitungen möglich. Wird beispielsweise erst bei der Definition von Inferenzmechanismen offensichtlich, dass weitere Wissensquellen benötigt werden, dann müssen die beiden vorherigen Phasen der Wissensidentifikation und Abbildung erneut durchlaufen werden.

### 4.5.1 Wissensidentifikation

Die Wissensidentifikation ist der initiale Schritt bei der Entwicklung einer Anwendung, welche auf der semantischen MPAD-Plattform basiert oder diese nutzt. Sie hat es zum Ziel, sämtliche Wissensquellen und damit verbundene Datenquellen zu identifizieren, welche für die Bearbeitung der Anwendungsaufgabe erforderlich sind.

Neben den beiden Wissensquellen der MPs und Anforderungskatalogen im ReqIF Format gilt es, relevantes *Domänenwissen* zu identifizieren. Unter Domänenwissen fällt explizites Wissen, beispielsweise aus Dokumenten oder Datenbanken sowie auch implizites Wissen.

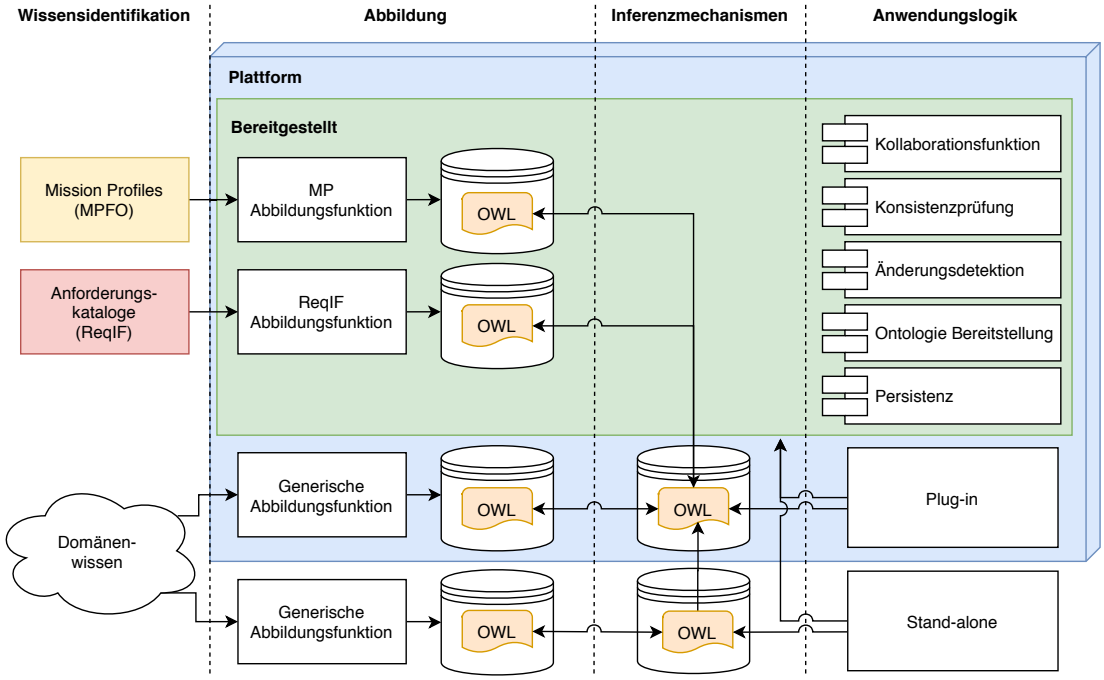



**Abbildung 4.2** – Entwurfsmethode der Entwicklung der semantischen Mission Profile Aware Design Plattform und darauf basierenden Anwendungen

### 4.5.2 Abbildung des Wissens auf Ontologien

In der zweiten Phase soll das identifizierte Wissen auf entsprechende Ontologien abgebildet werden. Dabei ist zwischen drei Fällen zu unterscheiden: (1) Abbildung von MPs oder Anforderungskatalogen im ReqIF-Format auf Ontologien. (2) Abbildung von Domänenwissen, wobei die Abbildungsfunktion ein Plattformbestandteil ist und (3) Abbildung von Domänenwissen, wobei die Abbildungsfunktion kein Plattformbestandteil ist.

Sowohl für die Abbildung von MPs als auch für die Abbildung von Anforderungskatalogen im ReqIF-Format auf OWL-Ontologien stellt die Plattform entsprechende Abbildungsfunktionen bereit. Die Abbildung von Domänenwissen muss dagegen jedoch stets für eine Anwendung implementiert werden. Wird die entsprechende Abbildungsfunktion jedoch als Bestandteil der Plattform implementiert, besteht später die Möglichkeit, diese auch bei anderen beziehungsweise neuen Anwendungen sehr einfach wiederzuverwenden. Diese Möglichkeit besteht allerdings auch dann, wenn die Abbildungsfunktion nicht als Bestandteil der Plattform implementiert wird. Dadurch wird ein Grad an Flexibilität erreicht, welcher es erlaubt, Abbildungsfunktionen der Plattform mit externen Abbildungsfunktionen zu kombinieren. Die Ablage und Verwaltung aus den Abbildungen resultierender Ontologien kann ebenfalls als Bestandteil der Plattform oder unabhängig von dieser realisiert werden,

wobei eine Kombination nicht ausgeschlossen ist. Diese Flexibilität wird durch eine Entkopplung zwischen der Anwendungslogik und dem in den Ontologien abgebildeten Wissen in Verbindung mit der Nutzung des darunterliegenden RDF erreicht: Über die URIs von Ressourcen können externe Inhalte referenziert werden, auch wenn diese nicht abrufbar sind. Dadurch ist es bei (3) möglich, Inhalte aus Ontologien aus einem Ansatz wie in (2) oder (1) zu nutzen und auch umgekehrt.

### 4.5.3 Definition von Inferenzmechanismen

Die dritte Phase der Entwurfsmethodik ist durch die Definition von Inferenzmechanismen bestimmt. Dies geschieht primär durch Spezifikation entsprechender OWL Axiome und sekundär durch die Spezifikation komplementärer SWRL Regeln in Ontologien. Durch das den OWL/SWRL Ontologien zugrundeliegende RDF können Inhalte übergreifend referenziert und verwendet werden. Dadurch ist eine nahtlose Integration von MP Ontologien, ReqIF Ontologien und den Ontologien möglich, in welchen Inferenzmechanismen definiert werden. Dies ist selbst dann möglich, wenn die Ontologien, in welchen Inferenzmechanismen definiert werden nicht als Bestandteil der Plattform realisiert werden.

Während eine Referenzierung externer Inhalte stets möglich ist, ist zu deren Verarbeitung die Abrufbarkeit dieser Voraussetzung. Dafür stellt die Plattform zwei Komponenten bereit: Einen Webserver, über den Ontologien verfügbar gemacht werden können, sowie eine Komponente die es ermöglicht Ontologien auf beliebigen RDF-Graph Datenbanken mittels SPARQL (siehe Abschnitt 2.1.3.5) bereitzustellen.

### 4.5.4 Definition der Anwendungslogik

In der vierten Phase wird die gesamte Anwendungslogik definiert. Dies kann durch (1) die Implementierung eines Plug-ins für die Plattform, oder (2) durch die Implementierung einer eigenständig lauffähigen Anwendung, geschehen. Es ist wichtig zu beachteten, dass die Anwendungslogik stets definiert werden muss und dass die Plattform lediglich Komponenten bereitstellt, die dabei nützlich sein können. Im Fall (2) nutzt die eigenständig lauffähige Applikation externe Funktionen der Plattform und kann sowohl mit eigenständig verwalteten Inferenzmechanismen interagieren, wie auch mit solchen, die von der Plattform verwaltet werden.

Flexibilität bei der Entwicklung der Anwendung wird dadurch erreicht, dass die Anwendungslogik sowohl als Plug-in der Plattform implementiert werden kann, wie auch als eigenständige Applikation. Während ersterer Ansatz bei der Implementierung die Programmiersprache auf Java oder Scala festlegt, kann bei letzterem Ansatz eine beliebige Sprache eingesetzt werden.

# 5 Reasoning-gestützte Robustness Validation

Die in diesem Abschnitt beschriebene MPAD Anwendung bezieht sich auf eine Methode zur Gewährleistung der Robustheit von Electrical/Electronic (E/E) Systemen, die im Handbook of Robustness Validation [10] vorgestellt wurde. Diese Arbeit wurde bereits in [1; 2] vom Autor dieser Dissertation beschrieben.

Um Robustheit von E/E-Systemen zu gewährleisten, schlägt das „Handbook for Robustness Validation“ eine Methode vor, bei der ein Schritt auch die Erstellung eines Plans umfasst, welcher spezifiziert, welche Analysen mit der zu untersuchenden Komponente durchgeführt werden müssen – der so genannte *Robustness Validation Plan*. Abbildung 2.6 auf Seite 28 stellt den Ablauf der RV dar, wobei die Erstellung des RV Plans unter Schritt fünf zu finden ist. Die Auswahl der Analysen, die im Rahmen der RV durchzuführen sind, hängt von den Eigenschaften der betrachteten Komponente, sowie deren Umgebung im späteren Einsatz ab. Dieses Kapitel beschreibt eine Methode zur Unterstützung bei der Durchführung der RV unter Einsatz von Ontologien und Inferenzmaschinen. Die Methode basiert auf der in Kapitel 4 vorgestellten Plattform und ist als ein Beispiel für eine MPAD Anwendung zu sehen. Für entsprechende Fallbeispiele sei der Leser zudem auf die Abschnitte 9.3.1 und 9.3.1 verwiesen.

## Abschnitte

## 5.1 Überblick

Die Idee dieser Anwendung ist es, Analytiker bei der Durchführung der RV, durch die Teilautomatisierung der Erstellung des RV-Plans, zu unterstützen. Im Folgenden wird beschrieben, wie dies unter Verwendung von formalisiertem Wissens aus dem RV Prozess und unter Berücksichtigung von MPs, erreicht werden kann. Im Unterschied zum fehleranfälligen manuellen Vorgehen wird gezeigt, wie die Analysenauswahl automatisiert und damit Unterstützung bei der Entscheidungsfindung während der RV erbracht werden kann. Der Ansatz basiert auf und nutzt eine Funktion der in Abschnitt 4 beschriebenen Plattform: Abbildung von MPs auf entsprechende Ontologien. Dies geschieht zum einen, um die Möglichkeit zu schaffen, semantische Anfragen gegen MPs zu stellen und des weiteren, um deren Integration in den RV Prozess zu verbessern. Es wird eine Generalisierung des Ansatzes beschrieben, was eine Anwendung auf andere Analysemethoden zulässt. Die wesentlichen Beiträge dieses Ansatzes sind:

- Verringerung der Komplexität des RV Prozesses
- Formalisierung von Domänenwissen in OWL
- Methodik zur Propagierung von Komponenteneigenschaften

## 5.2 Ablauf

Zunächst werden ein MP der betrachteten Komponente, dessen für die Analyseauswahl relevante Merkmale, sowie Hintergrundwissen spezifiziert. Die Spezifikationen sind daraufhin in OWL Ontologien zu überführen. Anschließend werden die Ontologien, welche die Komponentenattribute und das Hintergrundwissen darstellen, zur Analyseauswahl herangezogen. Abschließend werden anhand der Auswahl der Analysen passende MPs identifiziert mit dem Ergebnis der Liste der Analysen, sowie zugehörigen Stimuli in Form von MP Daten. Für eine schematische Abbildung sei der Leser auf Abbildung 5.1 verwiesen.

## 5.3 Metadaten-Extraktion zur Anreicherung von MPs

Um später eine richtige Entscheidung treffen zu können, welcher Datensatz für eine Analyse tatsächlich benötigt wird, ist es erforderlich Metadaten aus einem MP selbst und darüber hinaus noch weiteren Datenquellen zu extrahieren. Zum Zeitpunkt der Entwicklung stellte das MPFO eine reine Datenstruktur mit nur schwach definierter Semantik dar. Funktionale Lasten konnten zwar bereits detailliert spezifiziert

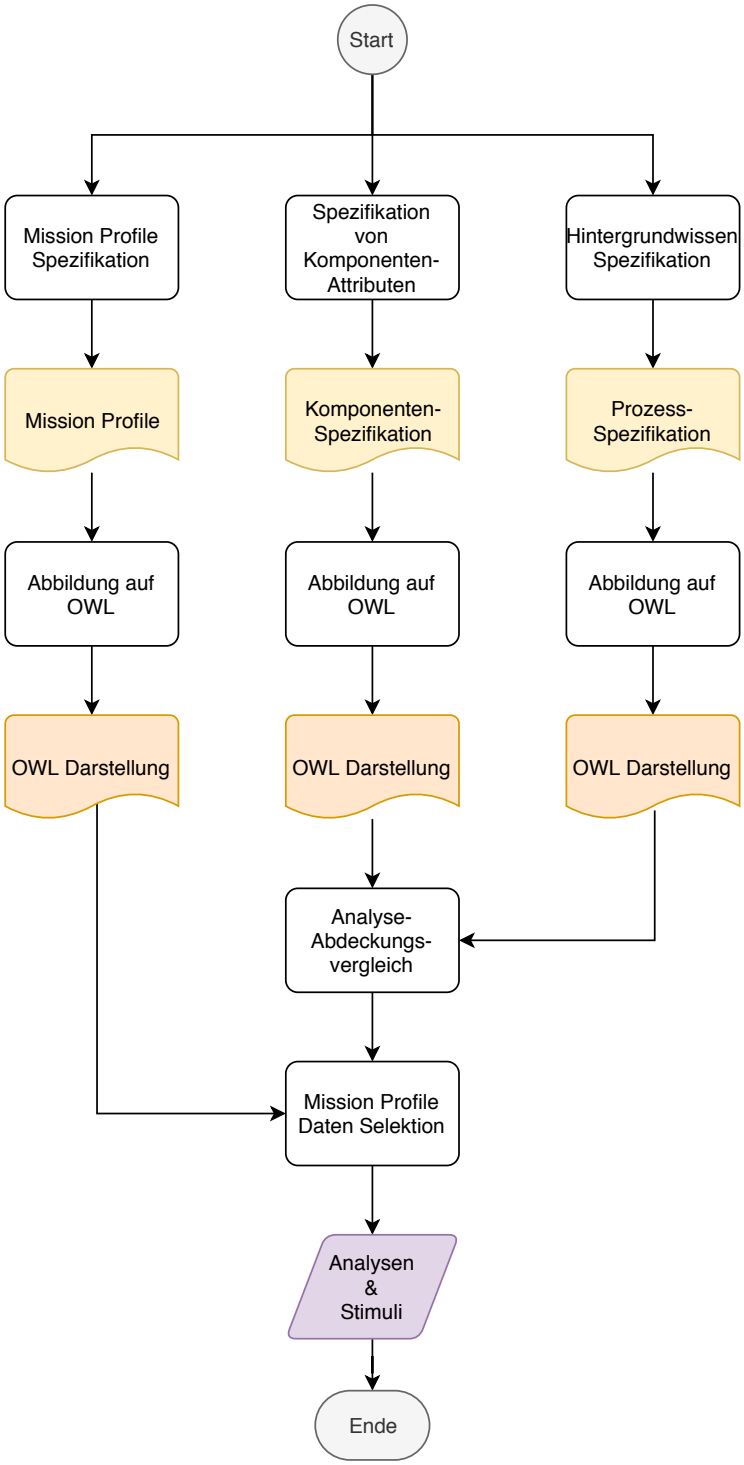



**Abbildung 5.1** – Überblick über den Ansatz, der alle Schritte und Ergebnisse darstellt

werden, jedoch fehlten wichtige Zusatzinformationen. Dies wird im Folgenden an einem Beispiel näher erklärt.

Angenommen, ein MP enthalte ein *TemperatureTimeProfileTemplate*. Diese Datenstruktur zur Abbildung von Temperaturprofilen besteht aus einer Matrix, die wiederum aus zwei Vektoren zusammengesetzt ist: dem *TemperatureRangeVector*, welcher Temperaturbereiche enthält und dem *PercentVector*, der die Verteilung der Temperaturen spezifiziert. Abbildung 5.2 zeigt einige der TBox-Informationen dieser Vorlage. Soll nun darüber entschieden werden, ob ein MP für eine Analyse benötigt wird, so sind die zugehörigen ABox-Fakten nicht ausreichend. Sie erlauben beispielsweise keine Aussage darüber, welche Umgebungsbedingungen zum Zeitpunkt der Messung herrschten, die durch klimatische Umstände oder angrenzende Systemkomponenten verursacht wurden. Strikt gesehen, ist nicht einmal die Bedeutung der Werte selbst klar, da das MPFO zu diesem Zeitpunkt der Entwicklung noch keine Einheiten für die Werte dieser Vorlage vorgesehen hatte. Eine Klassifizierung eines MPs wurde dadurch deutlich erschwert. Die Abszenz jener Metadaten im MPFO erforderte also die Metadaten Extraktion.

## 5.4 Analysenauswahl über Komponenteneigenschaften

Für spezifische Eigenschaften von Komponenten empfiehlt das Handbook for Robustness Validation bestimmte Analysearten. Hat beispielsweise eine Komponente mehr als 64 Pins oder eine Länge von mehr als 2.54 cm per Seite, so ist eine *Physical Stress Analysis* durchzuführen [10]. Die Zuordnung von derartigen Komponenteneigenschaften zu Analysearten bildet die Basis für die korrekte Auswahl von passenden Analysen. Im Folgenden soll die Menge der Eigenschaften, welche eine Analyseart erfordert, als *Abdeckung* bezeichnet werden. In den anschließenden Abschnitten 5.4.1, 5.4.2 und 5.4.3 wird beschrieben, wie die Analyseauswahl durch die Modellierung relevanter Aspekte ermöglicht wird.

### 5.4.1 Komponentenmodellierung

Um Merkmale einer Komponente im Analyseauswahlprozess heranziehen zu können, ist ein Modell der Komponente erforderlich, welches vom Entwickler der Komponente bereitgestellt werden muss. Quelltext 5.1 zeigt ein Beispiel für ein solches Modell. Konkret bezieht sich das Beispiel auf den ON Semiconductor® FTCO3V455A1[1], ein 3-Phasen Inverter für den Einsatz im Automobil.

[1]Ein Datenblatt zu dieser Komponente ist unter `https://www.onsemi.com/PowerSolutions/product.do?id=FTCO3V455A1` zu finden.

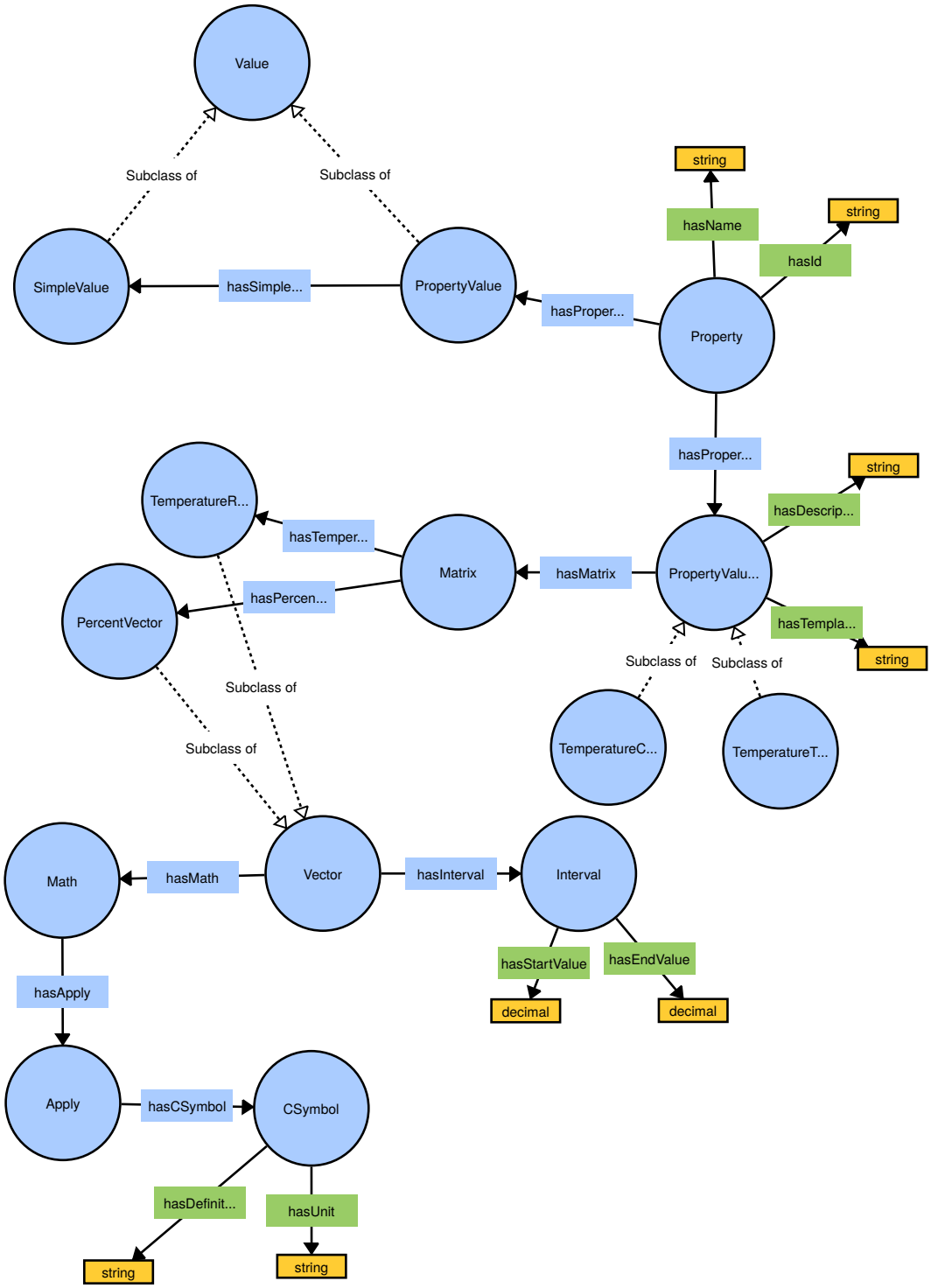



**Abbildung 5.2** – Auszug des MP Modellgraphen zur Komponenteneigenschaftsspezifikation als VOWL Darstellung

```
Individual: FTCO3V455A1

    Types:
        Component

    Facts:
        hasPurpose              Streering,
        mountedOn               EngineCompartmentMountingPoint,
        partOf                  MechanicalComponent1,
        hasHeight               29.0,
        hasPinCount             19,
        hasPowerDissipation     115.0,
        hasWidth                44.0
```

**Quelltext 5.1** – Spezifikation von Komponenteneigenschaften des ON Semiconductor® FTCO3V455A1 Inverters als OWL Individual dargestellt in Manchester Syntax

Insgesamt sollte das Komponentenmodell in ein übergeordnetes Systemmodell integriert werden, welches Komponenten in Beziehung setzt und die Möglichkeit schafft, Einbauorte zu definieren. Dies vereinfacht die zuvor, in Abschnitt 5.3 beschriebene Metadaten Extraktion und erlaubt außerdem die Propagierung von einbauortspezifischen Charakteristika, wie in Abschnitt 5.5 beschrieben.

### 5.4.2 Modellierung von Hintergrundwissen

Als *Hintergrundwissen* soll im Folgenden die Menge aller Fakten bezeichnet werden, welche im Handbook for Robustness Validation die Richtlinien zur Überprüfung auf beziehungsweise Maßnahmen zur Gewährleistung von Robustheit abbilden. Eines dieser Fakten ist beispielsweise dass Komponenten, welche über mehr als 64 Pins verfügen, eine hohe Anzahl an Pins haben. Dies lässt sich mittels Axiom 5.1 ausdrücken:

$$\exists \mathsf{hasPinCountValue}.(>, 64) \sqsubseteq \exists \mathsf{hasPinCount}.\mathsf{HighPinCount} \tag{5.1}$$

Werden feingranularare Modelle verwendet, beispielsweise wenn jeder Pin als eine Instanz der Klasse Pin dargestellt wird, so könnten *cardinality constraints* [144] wie in Axiom 5.2 verwendet werden:

$$> 64.\mathsf{hasPin}.\mathsf{Pin} \sqsubseteq \exists \mathsf{hasPinCount}.\mathsf{HighPinCount} \tag{5.2}$$

Dies sind Beispiele, die aus der konkreten Abbildung einiger solcher Fakten aus dem „Handbook for RV“ stammen. Weitere Fakten sind analog zu spezifizieren.

### 5.4.3 Modellierung von Analyseabdeckungen

Um eine Abdeckung für eine *Physical Stress Analysis* zu modellieren, kann als nächstes dann beispielsweise Axiom 5.3 verwendet werden:

$$\mathsf{Component} \sqcap \exists \mathsf{hasPinCount}.\mathsf{HighPinCount} \sqsubseteq \exists \mathsf{hasCoverage}.\mathsf{PSACoverage} \tag{5.3}$$

Wie zuvor sind weitere Abdeckungen analog zu formulieren. Dieses zweistufige Vorgehen folgt dem Entwurfsprinzip der Separierung von Anliegen und ermöglicht die zunächst unabhängige Definition von Abdeckungen und anschließende Zuweisung von Analysen.

## 5.5 Propagierung einbauortspezifischer Charakteristika

Um bestimmte Eigenschaften wie beispielsweise Elektromagnetische Störung (Englisch: Electromagnetic Interference, EMI), Temperatur oder Vibration, welche durch die Präsenz örtlich angrenzender Komponenten induziert werden, zu berücksichtigen, werden komplexe Rollen-Inklusionsaxiome (siehe [145]) eingesetzt, wie in Axiom 5.4:

$$\mathsf{mountedOn} \circ \mathsf{closeTo} \circ \mathsf{mountedOn}^{-} \circ \mathsf{hasEMI} \sqsubseteq \mathsf{hasEMI} \tag{5.4}$$

Die Verwendung derartiger Axiome emöglicht einem Reasoner EMI Werte für Nachbarkomponenten abzuleiten. Ein Beispiel hierfür: Wenn in einem Elektrofahrzeug eine Komponente nahe des elektrischen Antriebs verbaut ist, so ist diese Komponente hoher EMI ausgesetzt. Abbildung 5.3 stellt das Prinzip anhand eines Beispiels für die Propagierung des EMI Wertes für zwei Komponenten $C1$ und $C2$ dar, welche an den Einbauorten $M1$ beziehungsweise $M2$ verbaut sind. Bezugnehmend auf das vorher angeführte Beispiel, könnte Komponente $C1$ für einen Elektromotor stehen. Unter Einsatz von Axiom 5.4 kann ein Reasoner dann ableiten, dass Komponente $C1$ ebenfalls einem hohen EMI Wert ausgesetzt ist, was in der Abbildung durch den breiteren Pfeil dargestellt wird.

### 5.5.1 Einbauort-Normalisierung

Die im vorherigen Abschnitt beschriebene Propagierung von Einbauorteigenschaften erfordert fest definierte Einbauorte. Um dennoch herstellerspezifische Einbauorte berücksichtigen zu können, wird die *Einbauort-Normalisierung* vorgeschlagen. Die Idee dabei ist, dass spezialisierte Einbauort-Definitionen normale Einbauorte referenzieren, welche von diesem Ansatz definiert werden. Beispiele für normale Einbauorte wären: Fahrzeugheck, Fahrgastzelle, Unterboden, Dach etc. Hersteller könnten ihre

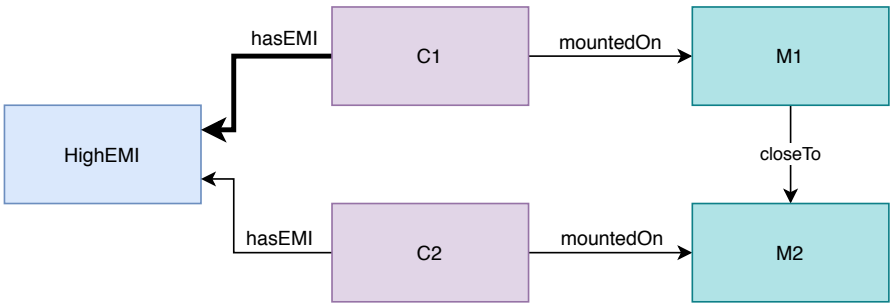



**Abbildung 5.3** – Beispiel für die Propagierung von einbauortspezifischen Charateristika

eigenen Einbauorte definieren und anschließend entweder rdfs : subClassOf oder closeTo Relationen spezifizieren, um die Einbauort-Normalisierung einzusetzen. Die konkrete Gestaltung der Referenzierung hängt vom gewählten Modellierungsstil ab. Es wäre beispielsweise auch möglich, den Mechanismus der Einbauort-Normalisierung dadurch umzusetzen, dass spezialisierte Einbauorte durch Instanziierung einer MountingPoint Klasse und Spezifikation von Eigenschaften jener Instanz, definiert werden.

Ein Beispiel für herstellerspezifische Einbauort-Definitionen sind die Definitionen von Audi und VW, welche in der Norm VW 80000 [146] vorgeschlagen werden. Im Verlauf des Projekts ResCar 2.0[2] wurde die Anzahl der Einbauort-Definitionen auf 17 erweitert, siehe Abbildung 5.4. Diese Einbauort-Definitionen enthalten zwei separate Einbauorte im Motorraum (Einbauort 03 beziehungsweise 04), die sich auf den gesamten Motorraum beziehungsweise auf Komponenten beziehen, welche direkt auf dem Motor angebracht werden. Diese Einbauorte können dann folgende Eigenschaften aufweisen: (1) beide Einbauorte sind MountingPoint-Instanzen, oder die Instanzen gehören zu einer Unterklasse davon und (2) die Nähe dieser Einbauorte zum Motorraum wird durch eine closeTo Relation ausgedrückt. Ein Reasoner kann dann den normalisierten Einbauort ableiten und entsprechende weitere Merkmale zu jedem dieser herstellerspezifischen Einbauorte hinzufügen.

### 5.5.2 Regelbasierte Propagierung

Die SWRL kann eingesetzt werden, um einer OWL-Ontologie Regeln hinzuzufügen, die mit den Sprachkonstrukten von OWL alleine nicht ausgedrückt werden können. Weitere Vorteile der Einführung regelbasierter Beziehungsdefinitionen sind, dass Regeln einfacher zu verstehen sind, weil sie stärker mit der intuitiven Formulierung von Beziehungen übereinstimmen und dass es einfacher ist, ein Werkzeug zur Verwaltung der Regeln zu implementieren. In diesem Ansatz wird die Verwen-

[2]Siehe `https://www.edacentrum.de/rescar/`

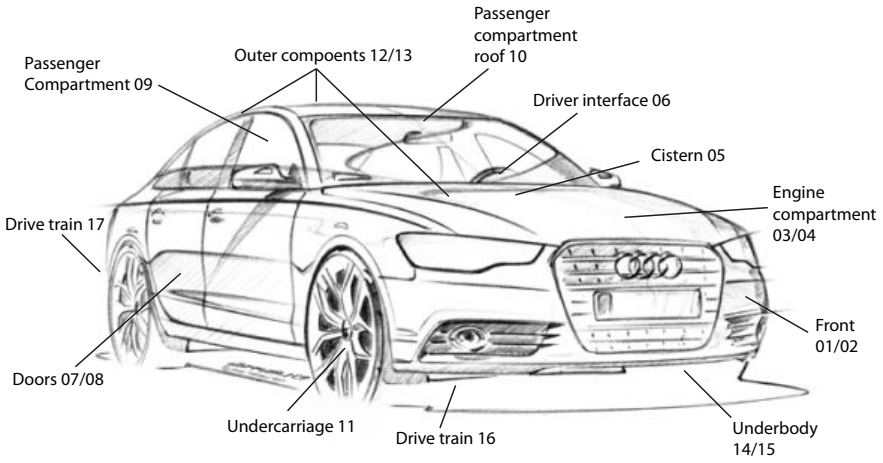



**Abbildung 5.4** – Einbauort-Definitionen eines Automobils, adaptiert aus [147]

dung von SWRL-Regeln zur Realisierung der Propagierung von Charakteristika vorgeschlagen – insbesondere für die Transformation von Werten. Ein Nachteil des zuvor im Abschnitt 5.5 beschriebenen Propagierungsmechanismus ist, dass es schwierig ist, konkrete Werte für materialisierte Electromagnetic Interference (EMI) Merkmale abzuleiten. Unter den Prädikaten, welche verwendet werden können, um eine Regel in SWRL auszurdrücken, gibt es eine Menge so genannte *Built-Ins* (siehe [44]) die verwendet werden können, um beispielsweise Vergleiche, mathematische Operationen und Manipulationen von Zeichenketten auf Variablen zu implementieren. Diese Built-Ins können benutzt werden, um eine Transformation bei der Propagierung von Werten zu modellieren. Weiter besteht die Möglichkeit, diese um benutzerdefinierte Built-Ins zu erweitern, welche speziell für die Berechnung von charakteristischen Werten implementiert wurden. Die folgende Regel 5.5 ist ein Beispiel für eine solche regelbasierte Transformation, bei welcher computeEMI im Rumpf der Regel ein benutzerdefiniertes Prädikat (Built-In) ist, welches das Ergebnis seiner Berechnung in der ungebundenen Variable z speichert, die dann wiederum der Component über das hasComputedEMI Prädikat im Kopf der Regel zugewiesen wird.

$$\mathsf{Component}(x) \wedge \mathsf{hasEMI}(x, y) \wedge \mathsf{computeEMI}(y, z) \rightarrow \mathsf{hasComputedEMI}(x, z) \quad (5.5)$$

Zur Realisierung dieses Ansatzes wurde die SWRLAPI[3] [148; 149] eingesetzt, um die regelbasierte Werte-Transformation bei Propagierung zu implementieren. Dieses Application Programming Interface (API) erlaubt es sogar, benutzerdefinierte Built-Ins zu transportieren, was eine Voraussetzung für die Schaffung der notwendigen

[3]Siehe `https://github.com/protegeproject/swrlapi`

Interoperabilität ebenjener Systeme, welche die Propagierung verarbeiten, und weiterhin zwischen letzteren und jenen Systemen, welche für die Entwicklung konkreter Transformationen verwendet werden.

### 5.5.3 Regel- und schwellwertbasierte Prüfung auf Störausstrahlung

Die Verwendung der im vorherigen Abschnitt angeführten Regel 5.5 ermöglicht die Berechnung konkreter Werte zur Prüfung auf EMI. Allerdings handelt es sich dabei lediglich um eine abstrakte Beschreibung der dazu erforderlichen Mittel. Im Rahmen unserer Untersuchungen wurde ein Regel- und Schwellwertbasierter Mechanismus zur Prüfung der EMI-Konformität entwickelt, welcher ausschließlich mittels Standard Built-Ins implementiert wurde.

Das Prinzip des Prüfmechanismusses ist, *Vergleichspaare* (englisch: *comparison pairs*) einzuführen und diesen anschließend Komponenten zuzuweisen, zusammen mit einem Maximalwert für die Rauschsignalstärke. Dies verlangt die Generierung von Instanzen einer ComparisonPair Klasse, was mit OWL oder SWRL alleine nicht möglich ist. Nachdem die Instanzen generiert und ihre Eigenschaften spezifiziert wurden, kann beispielsweise Regel 5.6 verwendet werden, um eine Exceeded Instanz einem Vergleichspaar zuzuweisen, wenn ein Schwellwert überschritten wird:

$$\begin{aligned}
&\mathsf{ComparisonPair}(\mathsf{p}) \wedge \\
&\mathsf{hasFirstComponent}(\mathsf{p}, \mathsf{c1}) \wedge \mathsf{hasSecondComponent}(\mathsf{p}, \mathsf{c2}) \wedge \\
&\mathsf{hasInterferingSignal}(\mathsf{c1}, \mathsf{is1}) \wedge \mathsf{hasInterferingSignal}(\mathsf{c2}, \mathsf{is2}) \wedge \\
&\mathsf{swrlb} : \mathsf{add}(\mathsf{resSig}, \mathsf{is1}, \mathsf{is2}) \wedge \mathsf{hasMaximumInterference}(\mathsf{p}, \mathsf{max}) \wedge \\
&\mathsf{swrlb} : \mathsf{lessThanOrEqual}(\mathsf{resSig}, \mathsf{max}) \rightarrow \mathsf{hasEMI}(\mathsf{p}, \mathsf{Exceeded})
\end{aligned} \tag{5.6}$$

Eine weitere Regel zur Assoziierung einer NotExceeded Instanz zu Vergleichspaaren, bei welchen der Schwellwert nicht überschritten wird, ist analog zu formulieren. In diesem Fall würde man jedoch swrlb : greaterThan anstelle des swrlb : lessThanOrEqual SWRL Built-In verwenden. Der beschriebene Prüfmechanismus ließe sich außerdem durch die Einführung von *Vergleichsgruppen*, oder einem anderen Ansatz zur Berücksichtigung der örtlichen Distanz von Komponenten weiter verbessern.

## 5.6 Auswahl passender Daten für Analysen

Durch die in Abschnitt 5.3 beschriebene Metadaten-Extraktion, sowie die in Abschnitt 5.4 beschriebene Analysenauswahl, können einem Analyst direkt passende Daten geliefert werden. Dies geschieht dann in Form vom MPs, die alle funktionalen und umgebungsbedingten Lasten enthalten, wie in Abschnitt 2.2.2 beschrieben.

Es wurde ein System-Prototyp entwickelt, der automatisch MP-Dokumente auf OWL-Ontologien abbildet, um semantische Anfragen an diese Dokumente zu erlauben, siehe Abschnitt 8.9. Dadurch können gezielt spezielle Inhalte von MPs abgerufen werden, die für eine bestimmte Analyse relevant sind. Ein Beispiel wäre die Bereitstellung von Temperatur- und Vibrationsdaten für den Fall, dass eine *Physische Stress Analyse* durchgeführt wird. Der Vorteil in der Verwendung semantischer Anfragen liegt in der Möglichkeit Anfragen sehr gezielt und in Abhängigkeit von der Umgebung, wie beispielsweise dem Systemmodell, stellen zu können. Darüber hinaus erlaubt dies, Anfragen nach implizit enthaltenen Informationen zu stellen. Ein Beispiel hierfür wäre eine Anfrage über die Temperaturdaten einer gewissen Komponente, welche auf einem spezifischen Einbauort verbaut ist und für den Fall, dass das Gesamtsystem über ein Verbrennungsmotor verfügt. Nach dem Prinzip von LD wird jede Entität durch RDF Links verbunden, wodurch ein globaler Datengraph entsteht, was die Entdeckung weiterer relevanter Datenquellen ermöglicht [57].

## 5.7 Verallgemeinerung

Eine Verallgemeinerung des Ansatzes und somit die Anwendung auf andere Analysemethoden ist immer dann möglich, wenn Analysenauswahl und zugehörige Datenselektion durchgeführt wird, siehe Abbildung 5.5. Die Grundlagen dafür bilden eine zu untersuchende Komponente, dieser entsprechende MP-Daten und Literatur, welche die Richtlinien und Prozeduren für die Analyseprozesse beschreibt. Der Kern des Ansatzes sind Abbildungsmechanismen, welche Komponentenmodelle, MPs und die besagten Literaturinhalte auf zugehörige Ontologien abbilden. Die semantisch anreicherbaren und feingranular interkorrelierbaren Repräsentationen dienen dann als Eingabe für den Analysenauswahlprozess. Ergebnisse der jeweiligen Analysemethoden sind Metriken, während die Ergebnisse des vorgestellten Ansatzes Mengen von Analysen und zugehörigen Stimuli sind, welche für die Berechnung der Metriken benötigt werden.

## 5.8 Nächste Schritte zur Bereitstellung

Um das beschriebene System bereitzustellen wurden drei wesentliche Aufgaben identifiziert, die dazu durchzuführen sind:

**Formalisierung von Komponenteneigenschaften** (Semi-)Automatische Extraktion von Komponenteneigenschaften aus existierenden Beschreibungen, um den Übergang vom manuellen zum automatisierteren RV-Prozess zu vereinfachen.

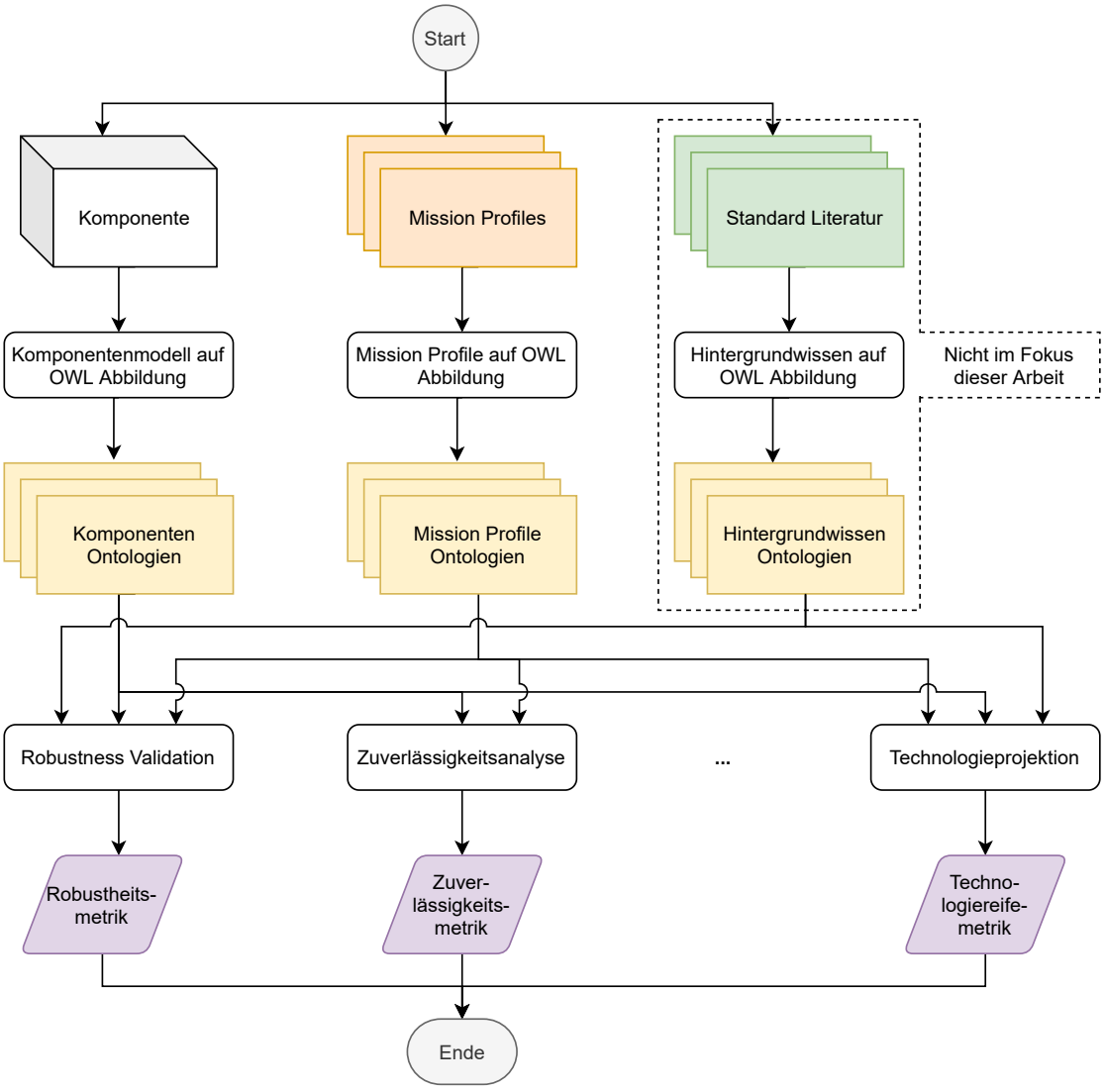



**Abbildung 5.5** – Verallgemeinerter Prozessablauf

**Erweiterung der Analysenmenge** Mit der Komplexität der Ausarbeitung des RV-Plans steigt auch die Nützlichkeit des Ansatzes. Durch die in Abschnitt 5.7 beschriebene Generalisierung können mehrere Analysemethoden integriert werden.

**Implementierung der Metadaten Extraktion** Zur Verbesserung der Auswahl passender Daten für die Analysen zu verbessern, wie in Abschnitt 5.3 beschrieben.

# 6 Ontologie-gestützte Change Impact Analysis

Dieses Kapitel stellt eine Methode zur Durchführung von Change Impact Analysis (CIA) unter Nutzung von Ontologien und der in Kapitel 4 vorgestellten Plattform vor, die bereits in [3] vom Autor dieser Dissertation beschrieben wurde.

Mittels dieser Methode können Entwurfsparameter verwaltet und mit Anforderungen verknüpft werden. Ziel dieses Ansatzes ist die Ermöglichung einer CIA durch Anforderungs-Rückverfolgbarkeit und gesammeltem Expertenwissen über Entwurfsparameter. Der Ansatz ist sowohl auf Software- wie auch auf Hardwareentwürfe anwendbar. Anforderungs-Rückverfolgbarkeit ist insbesondere bei der Entwicklung sicherheitskritischer Systeme von Bedeutung und wird von Standards wie ISO 26262 verlangt. Nur wenige CIA-Ansätze berücksichtigen mehrere Bereiche (wie Quelltexte, Architektur und diverse Entwurfsartefakte), weshalb Lehnert in [102] schlussfolgerte, dass der Verbindung von Anforderungen, Architekturen und Quelltexten mehr Aufmerksamkeit zukommen sollte. Die Methode nutzt die in Kapitel 4 vorgestellte Plattform für die Abbildung von MPs und Anforderungen auf Ontologien zu deren Verarbeitung, und ist somit als eine Anwendung dieser Plattform zu verstehen. Darüber hinaus nutzt die vorgestellte Anwendung ein weiteres Rahmenwerk, welches in Abschnitt 6.3 beschrieben wird. Ein entsprechendes Fallbeispiel zu dieser Anwendung ist weiterhin in Abschnitt 9.3.3 zu finden.

## Abschnitte

## 6.1 Überblick

Der Hauptzweck dieser Anwendung realisiert ein System, welches eine Antwort auf die Frage „Welche Entwurfsparameter sind betroffen, wenn eine Anforderung geändert wird?“ geben kann. Die Vorzüge des Ansatzes sind:

- Vermeidung von Fehlern und Verbesserung der allgemeinen Produktqualität
- Reduktion der Kosten durch die Verringerte Komplexität der CIA
- Verbesserung des Verständnisses des Entwurfs und der Beziehung zwischen Entwurfsparametern und Anforderungen

Üblicherweise werden Anforderungen entlang der Lieferkette weitergegeben, verfeinert und den Bedürfnissen des jeweiligen Zulieferers angepasst. Entwurfsartefakte oder Systeme können durch *Entwurfsparameter* beschrieben werden. Ein Entwurfsparameter ist in diesem Kontext ein beliebiges Merkmal eines Entwurfsartefakts oder eines Systems, wie beispielsweise die Reichweite eines Radarmoduls, eine Systemkonfigurationseinstellung oder die Laufzeit einer bestimmten Funktion einer Softwarekomponente. Entwurfsparameter haben übergreifende Wechselwirkungen, die als *Wirkzusammenhänge* bezeichnet werden. So steht beispielsweise die Reichweite eines Radarmoduls in Verbindung mit dessen Verstärkerleistung. Anforderungen können mit Entwurfsparametern verknüpft werden, siehe Abbildung 6.1.

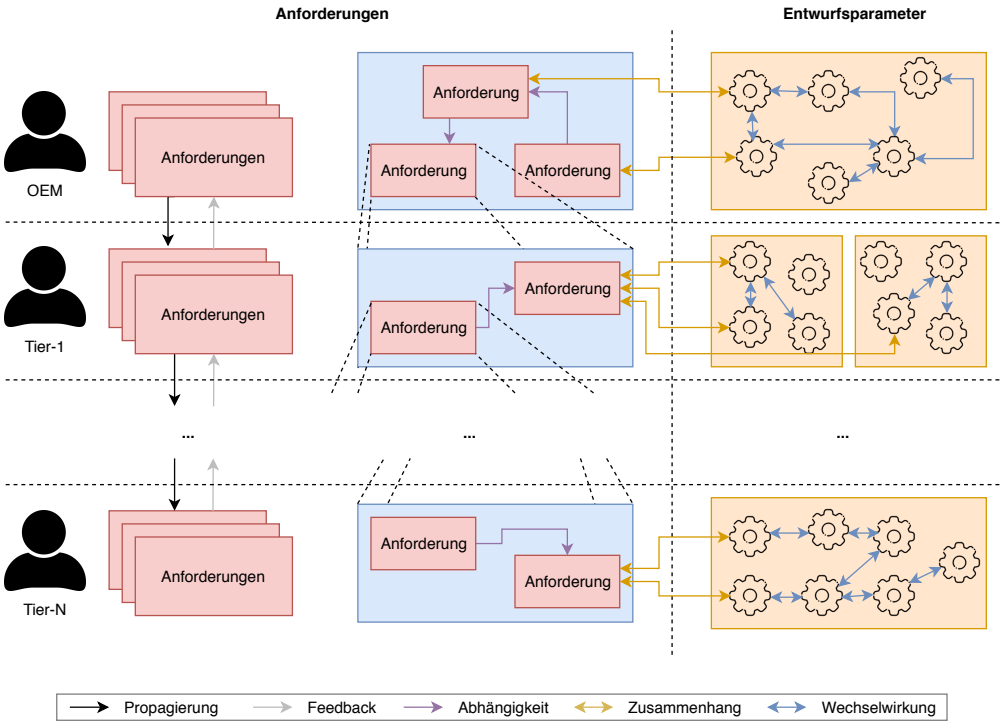



**Abbildung 6.1** – Zusammenhänge zwischen Anforderungen und Entwurfsparametern

## 6.2 Konzept

Da Wirkzusammenhänge in der Praxis größtenteils unbekannt sind, könnten die Auswirkungen von Änderungen an Anforderungen nicht berücksichtigt werden und dadurch Probleme verursachen. Zudem ist CIA komplex und erfordert signifikanten Aufwand. Grundsätzlich bietet dieser Ansatz Unterstützung bei der Entscheidungsfindung während der Entwicklung unter Verwendung von Expertenwissen über Wirkzusammenhänge. Die Kombination aus dem Einsatz von Trace Links zwischen Anforderungen und Entwurfsartefakten beziehungsweise deren Repräsentationen und der Nutzung von Expertenwissen über Entwurfsparameter vereinfacht die CIA und trägt dazu bei, die richtige Entscheidung mit adäquatem Aufwand zu treffen. Formalisiertes Systemwissen über Entwurfsparameter und ontologische Darstellungen von Systemen sind essenziell für automatisiertes Schlussfolgern, was beispielsweise in der Validierung, Transformation oder dem Austausch von Modellen und Daten verwendet werden kann. Darüberhinaus berücksichtigt dieser Ansatz MPs im Entwurfsablauf und ermöglicht so MPAD. Insgesamt wird dies durch die Integration von MPs, Anforderungen und der Entwurfsparameter auf der Basis der in Abschnitt 4 beschriebenen Plattform erreicht, siehe Abbildung 6.2.

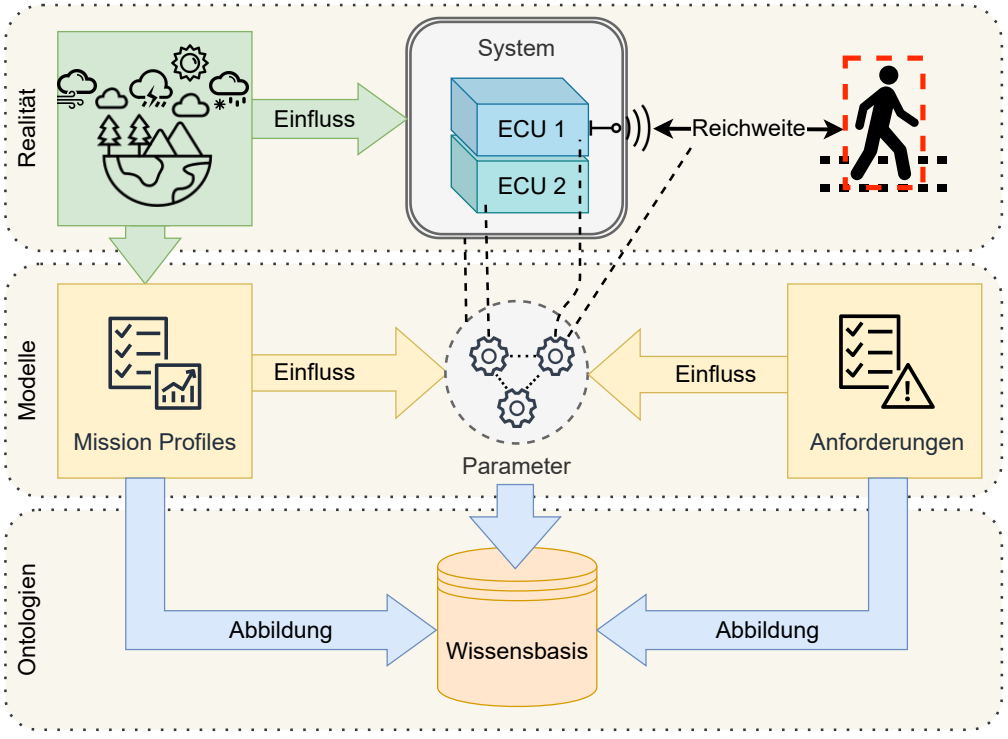



**Abbildung 6.2** – Integration von MPs, Anforderungen und Entwurfsparametern

## 6.3 Entwurf eines unterstützenden Rahmenwerks

CTEF

Für die Beschreibung der vielen Parameter von Komponenten und Systemen ist es wichtig Wirkzusammenhänge zwischen diesen zu beschreiben. Hierfür wird das bestehende Rahmenwerk Collaborative Technology Evaluation Framework (CTEF) eingesetzt. In diesem Abschnitt wird die Architektur und Nutzung des CTEF beschrieben, mit welchem dann eine konkrete Instanz einer Wissensbank realisiert wurde, siehe Abbildung 6.5. Die Repräsentation der Daten wird in [5] von Ahari et al. beschrieben. CTEF stellt die Mittel bereit, um Expertenwissen über Wirkzusammenhänge zu erfassen und wurde speziell für diese Aufgabe entwickelt. Das zentralisierte System verfügt über einen Server und einen Client, welche durch lose Kopplung miteinander verbunden sind, was diese einfach austauschbar macht. Während der Server beispielsweise von einer vertrauenswürdigen neutralen Partei betrieben würde, sollten die Clients domänenübergreifend von allen Teilnehmern einer Lieferkette genutzt werden. Abbildung 6.3 zeigt ein Bildschirmfoto der CTEF Client Applikation.

Erweiterung

Im Rahmen dieser Arbeit wurde CTEF um die Möglichkeit erweitert das gesammelte Wissen über Wirkzusammenhänge zwischen Parametern in eine Wissensbasis zu überführen.

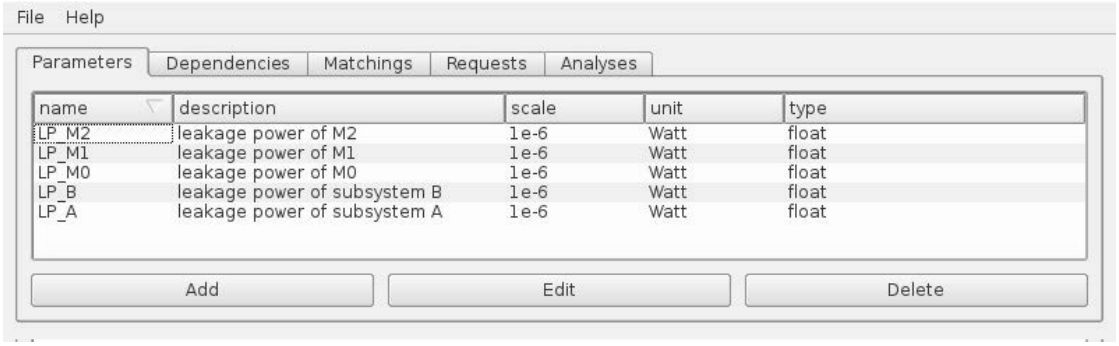



**Abbildung 6.3** – Bildschirmfoto der CTEF Client Applikation, mit aktivem Parameter-Reiter für die Parameter-Spezifikation. Dieser bietet die Möglichkeit, Parameter-Definitionen anzulegen, zu editieren und zu entfernen. Andere Reiter enthalten Abhängigkeits-, Übereinstimmungs- und Anfragenverwaltungsfunktionen, sowie Werkzeuge für die Durchführung von Analysen

### 6.3.1 Abhängigkeiten und Übereinstimmungen von Parametern

Sammel von Expertenwissen

Gesammeltes Expertenwissen wird Server-seitig gespeichert und in Form eines Abhängigkeits-Übereinstimmungs-Graph (AÜG) (aus dem Englischen *Dependency Matching Graph (DMG)*) dargestellt. Neben der Spezifikation von Entwurfsparametern, die entweder *privat* oder *öffentlich* sein können, erlaubt CTEF die Definition

von Abhängigkeiten zwischen den Entwurfsparametern. Die Definition von Relationen zwischen Entwurfsparametern ist ein grundlegendes Konzept des Rahmenwerks. Da es sich um ein kollaboratives Rahmenwerk handelt, können diese Relationen entweder *Abhängigkeiten* oder *Übereinstimmungen* sein. Übereinstimmungen sind Beziehungen zwischen Entwurfsparametern und Entwurfsparametern von *Anbietern*. Die Idee dahinter ist, dass eine Komponente oder ein System über einige Entwurfsparameter auf höchster Ebene verfügt, wie beispielsweise die Reichweite eines Radarsystems, die selbst wiederum von anderen Entwurfsparametern abhängen, welche von anderen Anbietern bereitgestellt werden. Ein Radarsystem könnte beispielsweise eine Verstärker-Komponente aufweisen, welche von einem anderen Anbieter in der Lieferkette hergestellt wird. Da die Reichweite eines Radarsystems auch von der Leistung dessen Verstärkers abhängt, wird der Anbieter der Verstärker-Komponente auch ein *Anbieter* eines zugehörigen Entwurfsparameters. Ein Paar übereinstimmender Entwurfsparameter wird in diesem Kontext *Übereinstimmung* genannt und bildet zusammen mit den Abhängigkeiten den AÜG. Ein konkretes Beispiel für einen AÜG wird im Kapitel zur Evaluation in Abbildung 9.4 auf Seite 119 dargestellt.

### 6.3.2 Übertragung von Wissen

Um das mit CTEF gesammelte Wissen tatsächlich zu Nutzen, verfügt die Serverkomponente zum einen über Funktionalitäten, welche verschiedene Aspekte der AÜG Relationen analysieren können und zum anderen über die Möglichkeit, gesammeltes Wissen in Form von RDF/XML-Serialisierungen zu exportieren. Die Nutzung des RDF-Datenmodells bildet die Schnittstelle zu Wissensbanken und -Systemen, wie die in Abschnitt 4 beschriebene Plattform, und ist die Basis für semantische Anreicherung und automatischem Schlussfolgern über die Erstellung von OWL-Ontologien. Des Weiteren ermöglicht dies die Durchführung semantischer Anfragen, beispielsweise mittels SPARQL, gegen das gesammelte Wissen. Die Einführung dieses Web-Konformitäts-Merkmals erlaubt die übergreifende Vernetzung von Wissen über bereitgestellte CTEF-Systeme auf Web-Skalierung, um hochkomplexe Wirkzusammenhänge zu erfassen und zu untersuchen.

## 6.4 Systemmodell und Wissenbasis

Das Systemmodell dient dem Zweck Modellintegration zu ermöglichen. Dies geschieht durch die Spezifikation verschiedener relevanter Relationen zwischen für die Modellintegration erforderlichen Daten beziehungsweise Modellen. Das Systemodell ermöglicht es *Trace Links* einzufügen und auch die CIA zu unterstützen, was in Abschnitt 6.4.1 beschrieben wird. Dazu sind mindestens die folgenden Relationen im Systemmodell zu spezifizieren:

- zwischen *Komponenten* und *Funktionen*
- zwischen *Komponenten* und *Entwurfsparametern*
- zwischen *Komponenten* und *MPs*

Und für die Implementierung von Anforderungs-Rückverfolgbarkeit:

- zwischen *Systemen* und *Anforderungen*
- zwischen *Komponenten* und *Anforderungen*

Das Systemmodell wird als OWL-Ontologie ausgedrückt und soll die Modellintegration vereinfachen, durch die spätere Referenzierung zu einem einheitlichen Referenzmodell. Die primären Aufgaben des Systemmodells werden in den folgenden Abschnitten 6.4.1-6.4.3 angeführt.

Eine Darstellung des Systemmodells in einer OWL-Ontologie ermöglicht eine nahtlose Einbindung der Wissensbasis in welche das gesammelte Expertenwissen über die Entwurfsparameter und deren Wirkzusammenhänge abgebildet ist.

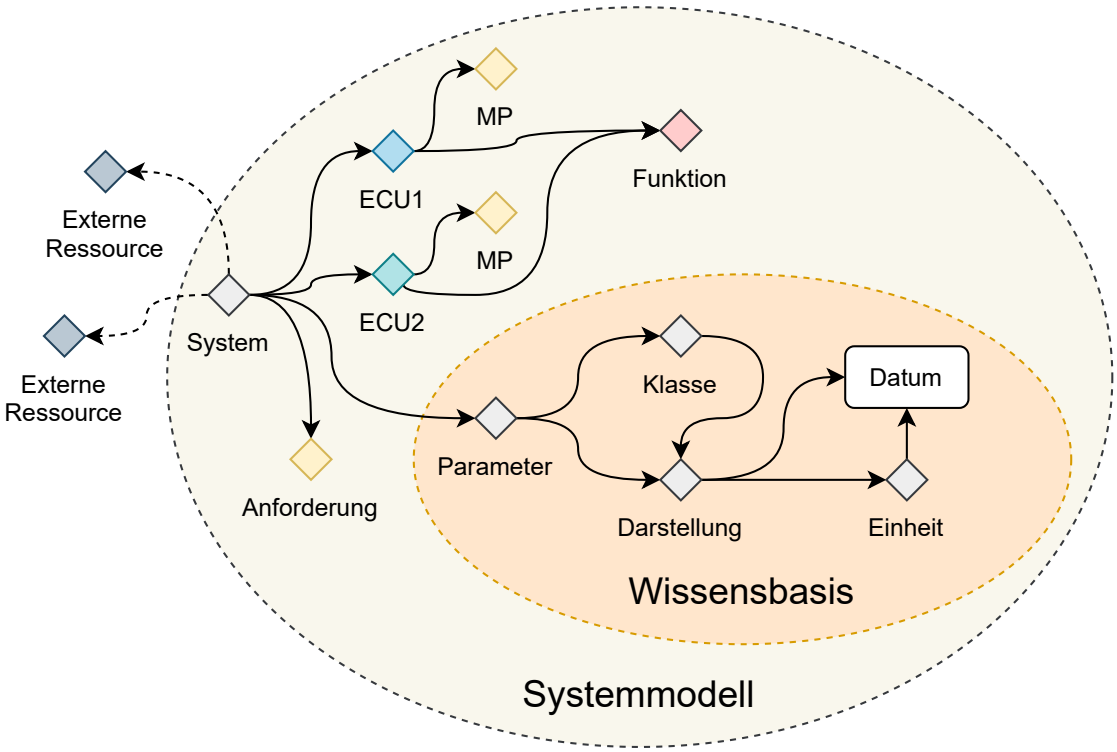



**Abbildung 6.4** – Beispiel eines Systemmodells und erforderlicher Relationen im Zusammenhang mit der Wissenbasis zu Entwurfsparamtern

### 6.4.1 Unterstützung der Change Impact Analysis

Die CIA wird in erster Linie durch die Spezifikation von Relationen zwischen *Anforderungen* und *Entwurfsparametern* über das Systemmodell unterstützt. Diese Beziehungen stellen *Trace Links* dar. Es wird eine semi-automatische Prozedur zum Sammeln dieser Beziehungen eingesetzt: Kandidaten für Links werden dem Benutzer vorgeschlagen, welcher diese entweder akzeptieren oder ablehnen kann. Eine Beschreibung einer konkreten Realisierung dieser Methode ist in Kapitel 8 Implementierung im Abschnitt 8.10 zu finden.

### 6.4.2 Ermöglichung von Anforderungs-Rückverfolgbarkeit

Das Systemmodell kann eingesetzt werden, um Anforderungs-Rückverfolgbarkeit zu ermöglichen, indem *Trace Links*, wie im vorherigen Abschnitt beschrieben, eingefügt werden.

### 6.4.3 Integration von Mission Profiles

MP-Daten können integriert werden, indem Verknüpfungen zwischen *Komponenten* und *MP-Repräsentationen* im Systemmodell eingefügt werden, siehe auch Abbildung 6.4. Es werden ontologische MP-Repräsentationen eingesetzt, gegen welche spezifische, semantische Anfragen nach Datenteilmengen gestellt werden können.

## 6.5 Ablauf

Ein Kernkonzept dieses Ansatzes ist die Verbindung von Entwurfsparametern und Anforderungen. Dies wird durch das *Lifting* von Expertenwissen, welches in einer Wissensbank gespeichert wurde und Anforderungen die in Anforderungsspezifikationsdokumenten zu finden sind, zu Ontologien erreicht, siehe Abbildung 6.5. Expertenwissen über Wirkzusammenhänge zwischen Entwurfsparametern wird zu einer Parameter-Ontologie angehoben. Anforderungen werden zu einer Anforderungs-Ontologie angehoben, gefolgt vom Einfügen von Verknüpfungen zwischen Entwurfsparametern und Anforderungen. Nach einer anschließenden Änderung einer Anforderung können betroffene Entwurfsparameter bestimmt und alle relevanten MP Daten dem Analyst bereitgestellt werden. Sobald die ontologischen Repräsentationen bereit sind, werden semi-automatisch Verknüpfungen eingefügt, was in Abschnitt 6.6 beschrieben wird. Nachdem das Systemmodell, welches in Abschnitt 6.4 beschrieben wird, erstellt wurde und Verknüpfungen zu Anforderungen eingefügt wurden, können Standard-Techniken zur Graphanalyse eingesetzt werden, um von einer Anforderungsänderung betroffene Entwurfsparameter zu bestimmen.

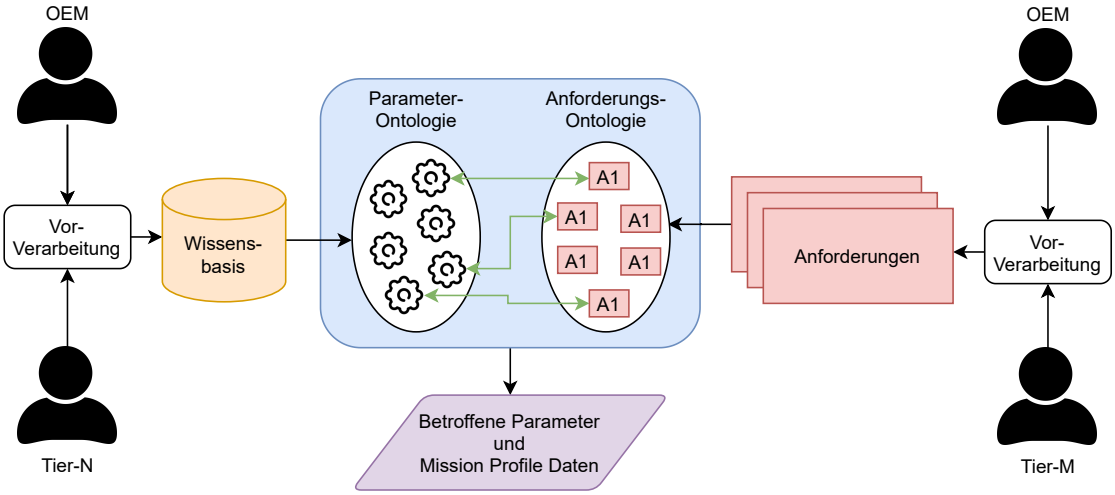



**Abbildung 6.5** – Überblick über den Informationsfluss

## 6.6 Anforderung-Entwurfsparameter-Verknüpfungen

Entwurfsparameter müssen mit zugehörigen Anforderungen verknüpft werden, damit das gesammelte Expertenwissen über Wirkzusammenhänge genutzt werden kann. Für diesen Zweck enthält das vorgeschlagene System eine Komponente für die Identifikation von Kandidaten für Verknüpfungen, wodurch eine semi-automatische Spezifizierung jener Verknüpfungen ermöglicht wird. Im Folgenden werden diese beiden Aspekte beschrieben.

### 6.6.1 Spezifizierung von Verknüpfungen

Das gesamte relevante Wissen (Systemmodell, Anforderungen und Entwurfsparameter) wird mittels Ontologien ausgedrückt, wobei W3C Empfehlungen[1] gefolgt wird. In Hinsicht auf das darunterliegende RDF Datenmodell, kann jeder Entwurfsparameter als *Ressource* betrachtet werden. Eine Verknüpfung zwischen diesen wird dann als Tripel dargestellt, wobei beide Teile durch ein Prädikat verbunden werden. Im Rahmen der Untersuchungen wurden zwei grundlegende Möglichkeiten zur Spezifikation von Verknüpfungen identifiziert, die in den folgenden Abschnitten 6.6.1.1 und 6.6.1.2 mittels Beispielen kurz beschrieben werden.

#### 6.6.1.1 Indirekte Spezifikation

Bei der indirekten Spezifikation von Verknüpfungen wird die Verknüpfung durch Generierung von die Verknüpfung repräsentierenden Individuen angegeben:

[1] Siehe `https://www.w3.org/standards/techs/owl#w3c_all`

```
ex:link    rdf:type    ex:Link
ex:link    ex:hasReq   req:requirement
ex:link    ex:hasDp    dp:designParameter
```

Da ein, die Verknüpfung repräsentierendes Individuum generiert wird, ist es einfach diesem später weitere Attribute hinzuzufügen. Es wurde diese Form der Repräsentierung von Verknüpfungen bei der Realisierung des Ansatzes gewählt, um möglichst flexibel beim Experimentieren mit Verknüpfungen sein zu können.

#### 6.6.1.2 Direkte Spezifikation

Bei der direkten Spezifikation wird eine unmittelbare Relation zwischen der Anforderung und den Entwurfsparameterinstanzen hinzugefügt:

```
req:requirement dpr:relatedTo   dp:designParameter
```

Im Unterschied zur indidrekten Spezifikation ist diese Möglichkeit zwar etwas einfacher was die Implementierung anbelangt, erschwert jedoch später das Hinzufügen weiterer Merkmale.

# 7 Ontologie-basierte Transformation von Anforderungen

Die in diesem Abschnitt beschriebene Methode zur Transformation von Anforderungen in Form von MPs wurde bereits in [4] vom Autor dieser Dissertation beschrieben.

Der hier vorgestellte Ansatz zielt ebenfalls auf die Optimierung des Entwicklungsprozesses von automobilelektronischen Systemen ab, was in diesem Fall durch die Unterstützung bei Transformationen von Anforderungen, welche als MPs gegeben sind, erreicht wird. Die Transformation von MPs ist ein, im Entwicklungsprozess automobilelektronischer Systeme notwendiger Schritt, bei der Weitergabe der Anforderungen entlang der Lieferkette. Neben der Unterstützung des Transformationsprozesses an sich, wird eine verbesserte Integration der Transformation in existierende MBSE Prozesse durch die vereinheitlichende Verwendung von OWL Ontologien zur Darstellung von Modellen ermöglicht. Die vorgeschlagene Methodik berücksichtigt weiterhin beim Transformationsprozess auch die Lieferkette in Hinsicht auf die spätere Anwendbarkeit von Fehlermodellen auf nachfolgenden Ebenen der Lieferkette. Der Ansatz nutzt die in Kapitel 4 beschriebene Plattform, um MPs auf entsprechende Ontologien abzubilden und kann somit ebenfalls als eine Anwendung der Plattform betrachtet werden.

Das Kapitel ist in drei Abschnitte aufgeteilt. Einem Überblick über den Ansatz, in Verbindung mit dem Hintergrund dieser Arbeit in Abschnitt 7.1, beziehungsweise 7.1 folgt eine Definition von allgemeinen Anforderungen an ein Transformationssystem für MPs in Abschnitt 7.2, in welchem auch entsprechende Evaluationskriterien in Abschnitt 7.2.2 definiert werden. Letztlich folgt diesen Abschnitten eine Beschreibung des Lösungsansatzes in Abschnitt 7.3.

## Abschnitte

## 7.1 Überblick

Das realisierende System ermöglicht eine Integration in existierende MBSE-Prozesse durch die Konstruktion und Nutzung von Modellen. Die konsequente Nutzung der standardisierten Sprache OWL drückt die Modellrepräsentationen aus und verbessert die Integration von Wissen und dessen Transfer zwischen heterogenen Systemen. Weiter begünstigt dies die projektübergreifende Wiederverwendung von Wissen, was die Gesamtkosten verringern kann. Das System ermöglicht zudem das Entfernen von irrelevanten Informationen aus MPs, was den Schutz von geistigem Eigentum verbessert. Bei der Transformation von Anforderungen in Form von MPs wird außerdem die Lieferkette berücksichtigt, sodass etwaige Fehlermodelle anwendbar bleiben. Die wesentlichen Beiträge des Ansatzes sind:

- Verbesserter Schutz geistigen Eigentums durch die Entfernung nicht benötigter Informationen aus MPs
- Verringerter Kommunikationsaufwand durch eine Verringerung der Gesamtanzahl benötigter Transformationen
- Projektübergreifende Teilbar- und Wiederverwendbarkeit von Transformationen
- Vermeidung von Fehlern im Transformationsprozess
- Reduzierte Kosten aufgrund verringerter Komplexität

### Hintergrund

Typischerweise werden Anforderungen entlang der Lieferkette weitergegeben und dabei verfeinert und auf die Anforderungen der entsprechenden Zulieferer heruntergebrochen. In diesem Prozess müssen MPs transformiert werden, wie beispielhaft in Abbildung 7.1 dargestellt.

Die Parametrisierung, sowie die Anwendung von Transformationen auf die MP Daten kann durch Ontologien und automatisches Schlussfolgern unterstützt werden. Nutzung von MBSE Methoden wird gemeinhin als effektiver Weg im Entwicklungsprozess von komplexen Systemen in der Automobil- und Luftfahrtindustrie angesehen [12; 13]. Der vorgestellte Ansatz folgt der Idee Modelle in der standardisierten Ontologiesprache OWL auszudrücken und zu nutzen, da diese Sprache die Spezifikation expliziter Semantik ermöglicht. Explizite Semantik fehlt meist in Entwurfsmodellen, ist aber erforderlich um automatische Schlussfolgerungen auf deren Basis ziehen zu können. Die explizite Semantik verbessert außerdem das Teilen und Wiederverwenden von Wissen unter heterogenen Systemen. Dadurch kann der Ansatz auch in existierende MBSE-Methoden integriert werden.

Die Entwicklung von komplexen Systemen stellt mehrere Ingenieursherausforderungen dar. Ein wesentliches Problem ist, dass SE-Daten primär in Dokumenten abgelegt werden, die über verschiedene Entwicklungscomputer verteilt sind und schlecht verwaltet werden [12]. In Hinsicht auf Anforderungen werden MPs beispielsweise auf nicht-standardisierte Weise transportiert, zum Beispiel über Excel- oder Word-Dokumente. Diese automatisch zu transformieren ist schwer, da jeweils eine angepasste Lösung für jede MP-Repräsentation implementiert werden muss. Ein weiteres Problem ist die Berücksichtigung von Systeminformationen und die Integration von relevanten Daten in den Transformationsprozess.

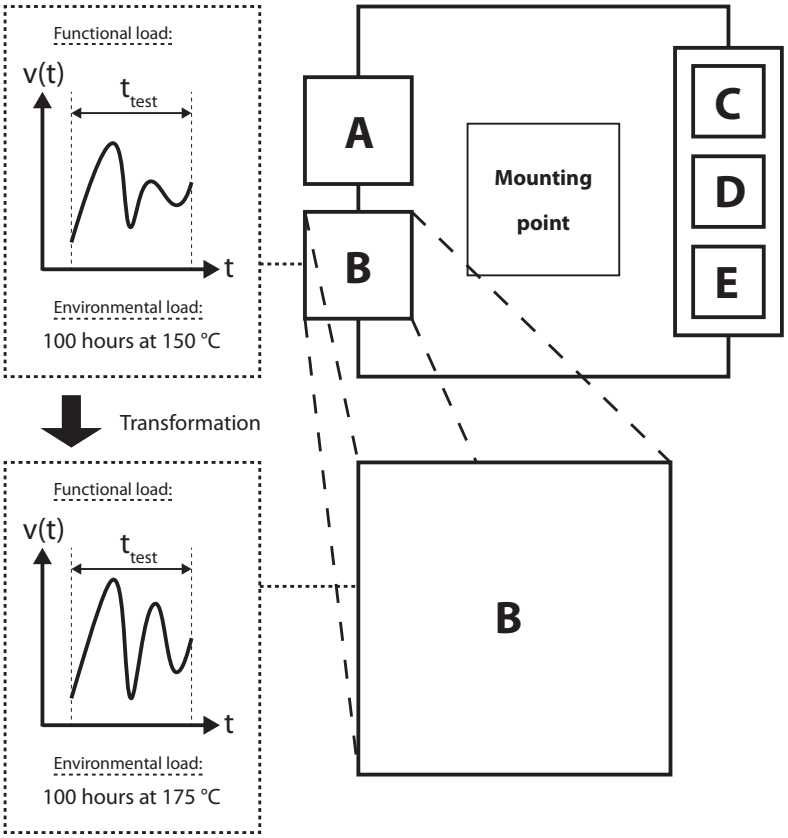



**Abbildung 7.1** – Transformation von Mission Profile Daten der funktionalen und umgebungsbedingten Lasten von Port B einer Komponente mit einem einzigen Einbauort

## 7.2 Anforderungen an das Transformationssystem

In diesem Abschnitt werden Hypothesen angeführt, von denen wichtige Schlussfolgerungen in Form von identifizierten abstrakten Schlüsselanforderungen, abgeleitet wurden. Basierend auf den Schlüsselanforderungen werden außerdem folgende Evaluationskriterien definiert. Die Schlüsselanforderungen sind bewusst sehr abstrakt gehalten und stellen eine Art Zwischenschritt dar.

### 7.2.1 Schlüsselanforderungen

Es werden drei Hypothesen aufgestellt, welche als Ausgangspunkt für weitere Betrachtungen genutzt werden. Die Hypothesen stützen sich auf Erfahrungen im praktischen Umgang mit MPs und auf Einblicke in die Praxis des MPAD im Bereich der Entwicklung von automobilelektronischen Systemen.

**Berücksichtigung der Lieferkette** Auf unterschiedlichen Ebenen der Lieferkette werden verschiedene Teile der ursprünglichen MP Daten benötigt. Um beispielsweise ein Fehlermodell auf der Ebene des Technologielieferanten anwenden zu können, müssen bestimmte MP Parameter auf vorhergehenden Ebenen beibehalten werden, siehe Abbildung 7.2.

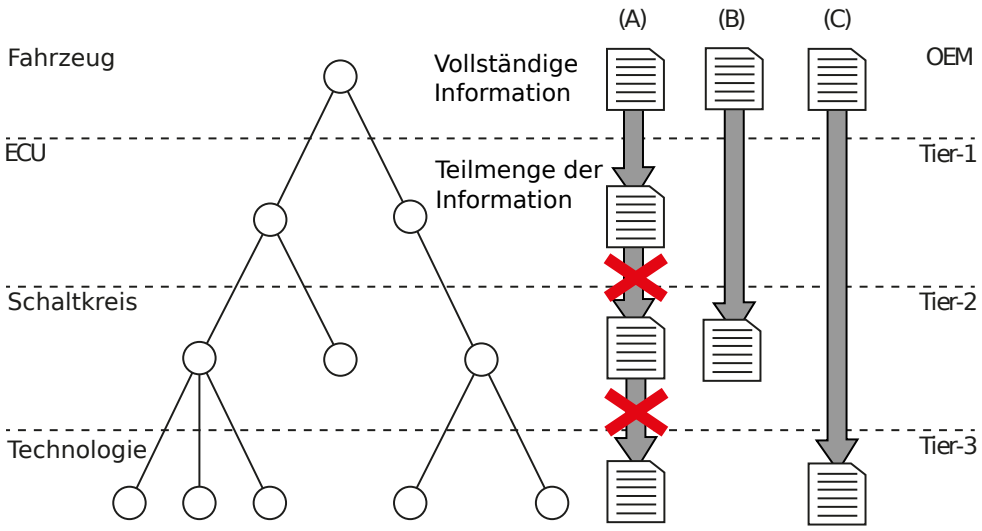



**Abbildung 7.2** – Beispielhafte Lieferkette und MP-Weitergabe. Auf der OEM Ebene sind alle Informationen als MP-Daten verfügbar. Transformation des Basis MPs für den Tier-1 resultiert typischer Weise in einer Untermenge der MP-Informationen. Ein nachfolgender Transfer von MP-Daten vom Tier-1 zum Tier-2 ist nicht möglich, da aufgrund der ersten Transformation Informationen fehlen könnten (A). Daher muss ein umständlicher Prozess gestartet werden, welcher die Transformationen des Basis-MPs auf die entsprechenden Anforderungen des Tier-1 und -2, beinhaltet (B), (C)

**Generische Lastspezifikationen** Um eine allgemeine Lösung für die Spezifikation von Umgebungslasten bereitstellen zu können, müssen die Mittel zur Beschreibung der Lasten vielseitig sein. Dies gilt dann auch für die Menge der Funktionen von Werkzeugen, welche diese Lastspezifikationen verarbeiten sollen. Die Entdeckung neuer, vorher unbekannter Lasten, welche einen Einfluss auf die zu beschreibende Komponente haben, erfordert die Erweiterbarkeit von Lastspezifikationsmechanismen und verwandter Manipulationsfunktionen von Verarbeitungswerkzeugen.

**Variierender Detailgrad** Der Detailgrad von MPs kann variieren. Deswegen müssen Werkzeuge, welche MP Daten verarbeiten, skalierbar sein. Darüber hinaus sollten Werkzeuge, welche MP Daten verarbeiten, adaptiv durch anpassbare Manipulationsoperationen sein, um verschiedene Detailgrade gleich gut handhaben zu können.

Es werden die folgenden abstrakten Schlüsselanforderungen für ein MP-Transformationssystem aus den Hypothesen extrahiert:

(1) Beibehaltungsmechanismus für Parameter

(2) Vielseitige Verarbeitungsfunktionen

(3) Erweiterbare Verarbeitungsfunktionen

(4) Anpassbare Manipulationsoperationen

(5) Skalierbarkeit

### 7.2.2 Evaluationskriterien

Die Abstraktionsebene der vorherigen Anforderungen macht es schwierig, zu entscheiden, welche Maßnahmen notwendig sind, um eine Anforderung zu erfüllen, weil es schwierig ist zu sagen, ob diese erfüllt wurde oder nicht. Dies erschwert zudem die spätere Evaluation der Lösung. Um dem zu entgegnen, wurden spezifische Evaluationskriterien definiert, welche mit den Schlüsselanforderungen assoziiert sind und präziser und spezifischer sind. Diese dienten dann auch als Zielsetzung dieses Vorhabens:

1 Das Transformationsystem enthält einen Mechanismus zur Beibehaltung von Parametern auf nachfolgenden Ebenen der Lieferkette.

2 Das Transformationssystem enthält mindestens zwei aus der Literatur bekannte Transformationsansätze, welche zu verschiedenen Klassen von Transformationen gehören.

3 Das Transformationssystem stellt einen Mechanismus zur Erweiterung um neue Transformationsansätze bereit.

4 Das Transformationsverhalten von Transformationsansätzen, welche vom Transformationssystem bereitgestellt werden, sollte modifizierbar sein.

5 Das Transformationssystem sollte verschiedene Detailgrade von MP-Daten gleich gut handhaben.

## 7.3 Lösungsansatz

Dieser Abschnitt beschreibt den Vorschlag zu Ontologie-basierter MP Transformation. Einem ersten Abschnitt, welcher einen Überblick über den Ansatz bietet, folgt ein weiterer Abschnitt, in dem das Lösungskonzept beschrieben wird, gefolgt von einem Abschnitt, welcher die Architektur des vorgeschlagenen Systems beschreibt.

### 7.3.1 Überblick

Abbildung 7.3 stellt einen Überblick über den Ansatz dar. Die Ebene der Lieferkette wird manuell spezifiziert (1). Diese Information wird dann in Verbindung mit bekannten Fehlermodellen für die Identifikation von Parametern genutzt, die bei der Transformation beibehalten werden müssen (2). Die manuelle Spezifikation von Transformationsoperationen hat ausführbare Transformationsoperationen zum Ergebnis (3). Das MPFO-Basisdokument wird über die in Kapitel 4 beschriebene Plattform auf eine MPO abgebildet (4). Schließlich wird die Ontologie-basierte Transformation der Basis-MPO durchgeführt (5).

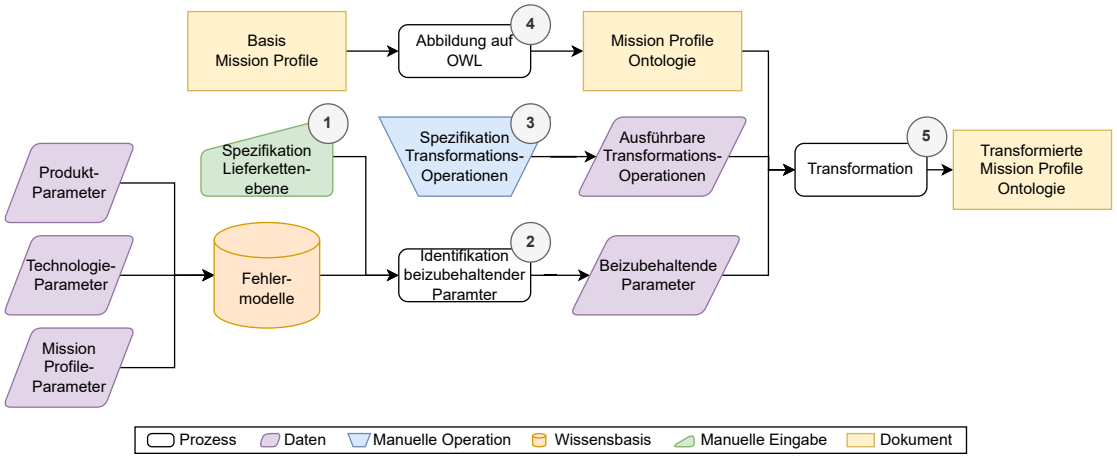



**Abbildung 7.3** – Überblick über den Ansatz

### 7.3.2 Konzept

Ein grundlegendes Merkmal der vorgeschlagenen Lösung ist die durchgängige Verwendung einer einheitlichen Sprache um Modelle auszudrücken: OWL. Der primäre Zweck davon ist, Schwierigkeiten zu mildern und Aufwand zu verringern, der bei der Integration von Daten, Informationen und Wissen zur Erfüllung der Aufgabe der MP-Transformation anfällt und um dessen Implementierung zu unterstützen.

Um Modelle zu erhalten, die in OWL ausgedrückt sind, empfiehlt sich das so genannte *Anheben* (aus dem Englischen: *lifting*) von existierenden Modellen. Anheben bedeutet: a) Translation des Quellmodells in geeignete OWL Ausdrücke und b) Anreicherung der resultierenden Ontologie mit zusätzlichem relevantem Wissen, um die Aufgabenerfüllung zu unterstützen. Die folgenden fünf Paragraphen beziehen sich auf die zuvor festgelegten Ziele und beschreiben die Maßnahmen, welche zur Erfüllung der Anforderungen und des übergeordneten Ziels dieses Ansatzes, ergriffen wurden.

**1) Berücksichtigung der Lieferkette** Die Berücksichtigung der Lieferkette wird durch einen Mechanismus zur MP-Parameter-Beibehaltung erreicht. Die Idee dahinter ist, Parameter als *benötigt* zu markieren, wenn diese Parameter Bestandteil von Fehlermodellen sind, die auf nachfolgenden Ebenen der Lieferkette angewendet werden könnten. Dieser Prozess referenziert ein Ontologiebasiertes Verzeichnis von Fehlermodellen, welches Teil des vorgeschlagenen Systems ist.

**2) Mehrzahl an Transformationsansätzen** Um zwischen Transformationsansätzen zu unterscheiden wird, auf die Klassifikation von Czarnecki und Helsen in [87] zurückgegriffen. Es wurde ein *direct-manipulation* und ein *relational* M2M Ansatz integriert.

**3) Erweiterbarkeit** Das Konzept setzt Erweiterbarkeit durch einen Plug-in Mechanismus um. Das konkrete Design dieses Mechanismusses hängt von der gewählten Technologie ab, mit welcher das System implementiert wird. Im vorliegenden Fall wurde eine multi-paradigmatische (objektorientiert und funktional) Programmiersprache genutzt, um das Transformationssystem zu implementieren. Weitere Transformationsansätze können dem System durch die Implementierung vordefinierter Schnittstellen hinzugefügt werden.

**4) Modifizierbarkeit** Der Ansatz berücksichtigt Modifizierbarkeit des Verhaltens von Transformationsansätzen im Design des implementierten Transformationssystems. Dies wird primär dadurch erreicht, dass die Spezifikation einer Transformation nach dem gewählten Transformationsansatz bereits eine Definition des Verhaltens impliziert. Zusätzlich dazu können Transformationsan-

sätze parametrisiert werden, was den positiven Nebeneffekt einer verbesserten Wiederverwendbarkeit ergibt.

**5) Gleichbleibende Leistung** Um verschiedene Detailgrade in MPs handhaben zu können, verlässt sich das vorgeschlagene System auf etablierte Bibliotheken. Die Definitionen von Transformationen haben jedoch direkten Einfluss auf die Leistungsfähigkeit, was vom Benutzer zu beachten ist.

### 7.3.3 Architektur

Dieser Abschnitt beschreibt die Fehlermodell- und Transformations-Ontologie, sowie die Architektur des vorgeschlagenen MP-Transformationssystems.

#### 7.3.3.1 Fehlermodell- und Transformations-Ontologie

Um die Lieferkette zu berücksichtigen und damit Ziel  zu erriechen, wird eine Ontologie eingeführt, welche eine bestimmte Angelegenheit adressiert. Diese Ontologie hat den Zweck, Parameter zu identifizieren, welche bei einer Transformation beibehalten werden müssen. Die OWL-Ontologie enthält SWRL-Regeln für die automatische Schlussfolgerung, welche Parameter beibehalten werden müssen. Es wird das automatische Schlussfolgern, das eine Kombination von OWL-Axiomen und SWRL-Regeln nutzt, durch die Hervorhebung von dazu bedeutsamen Axiomen und Regeln der Ontologie, beschrieben. Abbildung 7.4 zeigt die Taxonomie der Fehlermodell- und Transformations-Ontologie. Definitionen der aufgeführten OWL-Properties, -Klassen und -Instanzen sind in Anhang B zu finden.

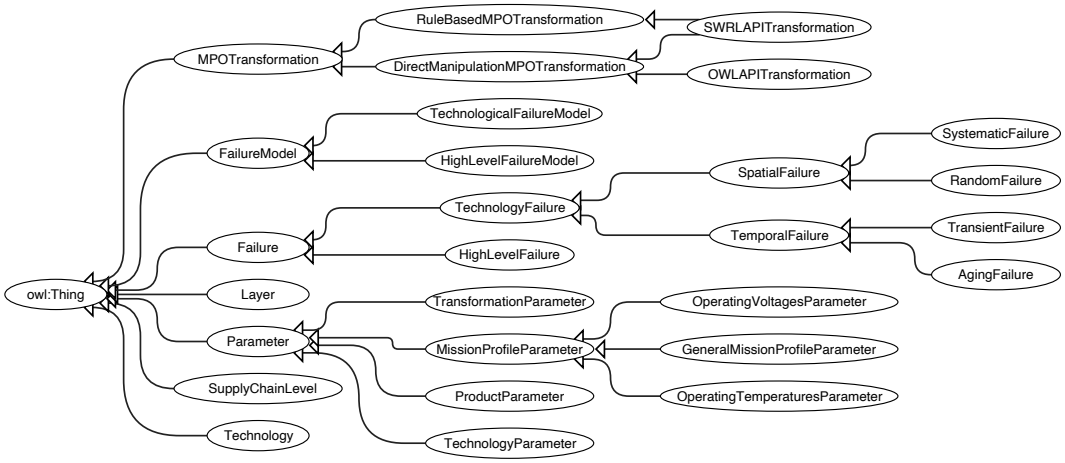



**Abbildung 7.4** – Taxonomie der Fehlermodell- und Transformations-Ontologie

FailureModel Das Konzept eines Fehlermodells wird durch eine Klasse modelliert, welche eine Unterklasse der Klasse ist, die durch Axiom 7.1 ausgedrückt wird:

$$\begin{aligned} &\exists \mathsf{hasEquation.xsd{:}string} \sqcap \\ &\exists \mathsf{hasMissionProfileParameter.MissionProfileParameter} \sqcap \\ &\exists \mathsf{ofFailure.Failure} \end{aligned} \tag{7.1}$$

TechnologicalFailureModel Ein technologisches Fehlermodell ist eine Spezialisierung und damit eine Unterklasse eines Fehlermodells. Zusätzlich ist es eine Unterklasse der Klasse, die durch Axiom 7.2 ausgedrückt wird:

$$\begin{aligned} &\exists \mathsf{hasProductParameter.ProductParameter} \sqcap \\ &\exists \mathsf{hasTechnologyParameter.TechnologyParameter} \sqcap \\ &\exists \mathsf{ofFailure.TechnologicalFailure} \sqcap \\ &= 1\ \mathsf{hasLayer.Layer} \end{aligned} \tag{7.2}$$

MPOTransformation Eine Transformation wird als Klasse modelliert, welche ebenfalls Unterklasse der Klasse ist, die durch Axiom 7.3 ausgedrückt wird:

$$\begin{aligned} &\geqslant 0\ \mathsf{maintainsParameter.MissionProfileParameter} \sqcap \\ &= 1\ \mathsf{ofSupplyChainLevel.SupplyChainLevel} \end{aligned} \tag{7.3}$$

Das automatische Schlussfolgern nutzt darüber hinaus, wie bereits erwähnt, SWRL Regeln, um Parameter zu markieren, die bei einer Transformation beibehalten werden müssen. Dies wird im Folgenden beschrieben.

MaintainParametersOEM Regel 7.4 legt MP Parameter fest, die bei einer Instanz von MPOTransformation auf dem OEMLevel stattfinden, wenn es eine Instanz von TechnologicalFailureModel gibt, die diese Parameter nutzt. In Abhängigkeit von der Beschaffenheit der Lieferkette müssen analog Regeln für jede Ebene der Lieferkette formuliert werden.

$$\begin{aligned} &\mathsf{MPOTransformation}(\mathsf{trans}) \wedge \\ &\mathsf{ofSupplyChainLevel}(\mathsf{trans}, \mathsf{OEMLevel}) \wedge \\ &\mathsf{TechnologicalFailureModel}(\mathsf{fm}) \wedge \\ &\mathsf{hasMissionProfileParameter}(\mathsf{fm}, \mathsf{param}) \\ &\rightarrow \mathsf{maintainsParameter}(\mathsf{trans}, \mathsf{param}) \end{aligned} \tag{7.4}$$

MaintainParametersHighLevel Es müssen außerdem alle Parameter markiert werden, die beibehalten werden müssen, in Hinsicht auf Fehlermodelle von hohen

Abstraktionsebenen. Regel 7.5 zeigt, dass dies erreicht werden kann, indem man überprüft, ob ein Fehlermodell einen Fehlermechanismus hat, der auf die Technologieebene zurückzuführen ist.

$$\begin{aligned}
&\mathsf{MPOTransformation}(\mathsf{trans}) \wedge \\
&\mathsf{ofSupplyChainLevel}(\mathsf{trans}, \mathsf{OEMLevel}) \wedge \\
&\mathsf{HighLevelFailureModel}(\mathsf{fm}) \wedge \\
&\mathsf{ofFailure}(\mathsf{fm}, \mathsf{failure}) \wedge \\
&\mathsf{hasRootFailure}(\mathsf{failure}, \mathsf{rootFailure}) \wedge \\
&\mathsf{TechnologicalFailure}(\mathsf{rootFailure}) \wedge \\
&\mathsf{hasMissionProfileParameter}(\mathsf{fm}, \mathsf{param}) \\
&\rightarrow \mathsf{maintainsParameter}(\mathsf{trans}, \mathsf{param})
\end{aligned} \tag{7.5}$$

Des Weiteren wurde eine Taxonomie für Fehlermodelle aus den Kategorien extrahiert, die Maricau und Gielen in [150] vorgestellt haben. Complementary Metal-Oxide-Semiconductor (CMOS) Fehlermodelle wie Electromigration (EM), Hot-Carrier Injection (HCI) oder Time-Dependent Dielectric Breakdown (TDDB) wurden als Instanzen entsprechender Klassen hinzugefügt.

#### 7.3.3.2 System

Siehe Abbildung 7.5 für eine Übersicht über die Typenhierarchie. Dabei gilt es zu beachten, dass die Scala Programmiersprache eingesetzt wurde, welche so genannte *Traits* im Unterschied zu klassischen Schnittstellen, bereitstellt. Scala Traits erlauben Mehrfachvererbung und können außerdem auch über Attribute verfügen. UML hat kein semantisches Konzept für diese Art von Vererbung. Jedes Objekt im Transformationssystem ist ein *TransformationObject*, welches ein *metadata* Attribut von beliebigem Typ bereitstellt, basierend auf Schlüssel-Wert Zugriff. Um den Zielen 2 und 3 nachzukommen, wurde ein Trait für Transformationen eingeführt: *MPOTransformation*. Dieser Trait enthält *Mappings*, die selbst wiederum *sources* und *targets* Listenattribute enthalten. Als Beispiel ist ein *OneToOneMapping* enthalten, welches vom *Mapping* Trait erbt und die Quell- und Ziel-Attribute überschreibt, sodass diese nur ein einzelnes Element enthalten. Diese Art des Entwurfs ermöglicht die Definition von One-to-One, One-to-Many, Many-to-One und Many-to-Many Abbildungen. Dies wird benötigt, um die notwendige Flexibilität bei MP Transformationen zu erhalten, um Anwendungsfälle abzudecken wie beispielsweise das Zusammenführen mehrerer MPs in ein Einziges. Konkrete Daten wie Dateipfade, Uniform Resource Locator (URL) Angaben oder Datenbankadressen, die benötigt werden, um die Quellen beziehungsweise Ziele zu nutzen, können über die Metadaten transportiert werden.

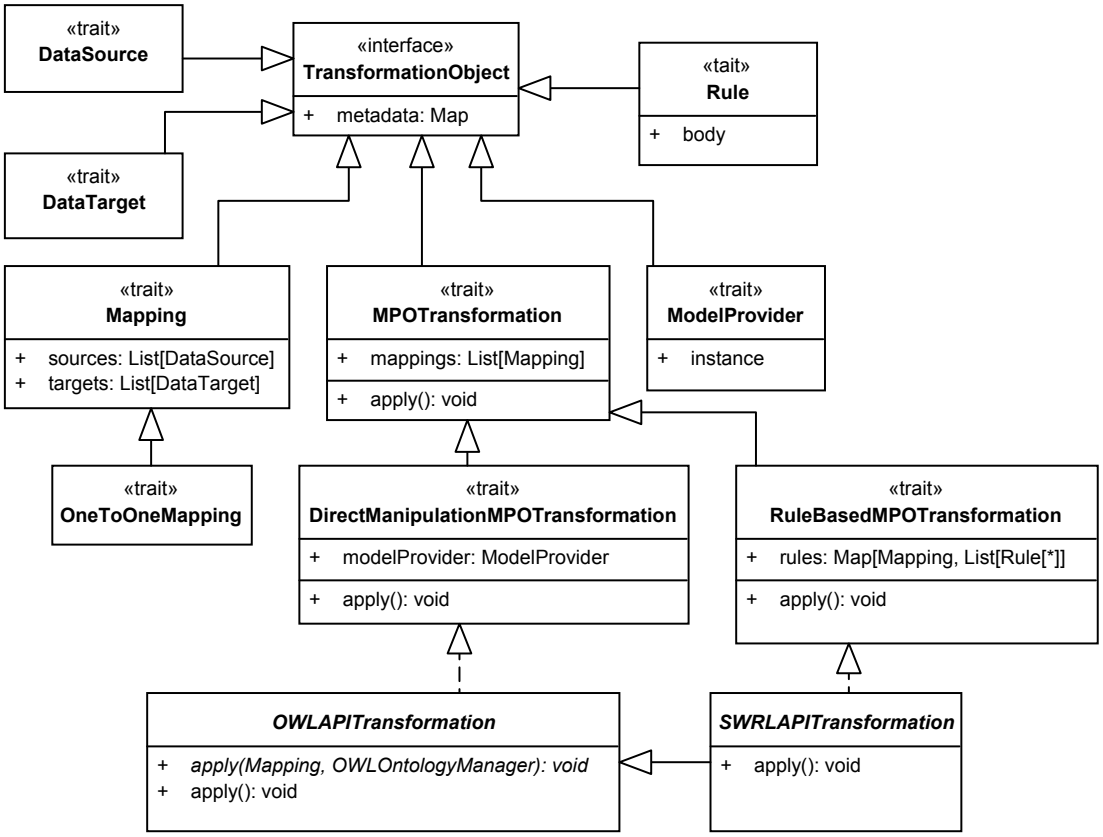



**Abbildung 7.5** – UML Diagramm des vorgeschlagenen Systems

Ein *DirectManipulationMPOTransformation*-Objekt hat ein *modelProvider* Attribut, welches eingesetzt wird, um auf ein Objekt zuzugreifen, welches eine Modellmanipulations-API bereitstellt, was von *Direct Manipulation*-Ansätzen gefordert wird. *RuleBasedMPOTransformation*-Objekte enthalten weiterhin ein *rules* Attribut, was im Wesentlichen eine Abbildung von *Mapping* Objekten auf eine Liste von Regeln darstellt. Dies erlaubt die spezifische Assoziierung von Regeln mit bestimmten Abbildungen. Ein *Rule* Objekt selbst verfügt über ein *body* Attribut von beliebigem Typ. Während der Trait *OWLAPITransformation* von *DirectManipulationMPOTransformation* erbt, erbt eine *SWRLAPITransformation* ebenfalls von diesem Trait und weiterhin von *RuleBasedMPOTransformation*.

Um Ziel 4 zu erreichen, lässt das System Benutzer das Transformationsverhalten durch das Überschreiben der *apply* Methode von Transformations-Traits in den implementierenden Klassen oder spezialisierenden Traits definieren. Das vorgeschlagene Transformationsssystem stützt sich zur Erreichung von Ziel 5 auf etablierte Bibliotheken. Jedoch haben auch durch den Benutzer definierte Transformationen direkten Einfluss auf die Leistungsfähigkeit des Transformationssystems.

# 8 Implementierung

Im Rahmen dieser Arbeit wurde ein Prototyp des in Kapitel 4 vorgestellten Plattform-Konzepts als eine einheitliche Basis für MPAD Anwendungen entwickelt. Diese nutzt Ontologien, sowohl um Daten zu integrieren, als auch zu verarbeiten. Die Plattform ermöglicht den Anwendern, MP-Daten und Anforderungen aus Dokumenten zu beziehen und zu verarbeiten, ohne die Details der jeweiligen Format-Realisierung zu kennen. Dies wird durch die Abbildung der Dokumente auf Ontologien ermöglicht, was die Kernidee der Plattform darstellt. Die Ontologien fungieren dabei im Weiteren dann auch als *Referenzmodelle*.

Die Plattform wurde mit der Programmiersprache Scala [151] als eine Play Framework[1] Anwendung implementiert. Dabei wurde modular entwickelt, um die Wartbarkeit zu erhöhen und Austauschbarkeit von Komponenten zu ermöglichen. Die gesamte Applikation kann später komplett mit allen benötigten Bibliotheken bereitgestellt werden, was die Inbetriebnahme vereinfacht. Die Web-Graphical User Interface (GUI) des Plattform Prototypen bietet zudem Möglichkeiten Ontologien, Mission Profiles und Anforderungsdokumente zu verwalten. Abbildung C.1 zeigt Ausschnitte der grafischen Benutzerschnittstelle der Plattform.

## Abschnitte



[1] Siehe `https://www.playframework.com/`

## 8.1 Alternativen

Die auf dem in Kapitel 4 vorgestellten Plattform-Konzept basierenden Anwendungen, welche in den Kapiteln 5, 6 und 7 beschrieben wurden, hätten auch basierend auf einem anderen, als dem Plattform-Ansatz realisiert werden können. Neben dem gewählten Lösungsansatz standen noch weitere Alternativen zur Auswahl, deren Vor- und Nachteile im Vergleich zum Plattform-Ansatz im Folgenden zur Diskussion kurz diskutiert werden.

### 8.1.1 Alleinstehende, unabhängige Anwendungen

Auf eine vereinheitlichende Plattform hätte auch verzichtet werden können. Es wäre möglich gewesen, einzelne, voneinander vollständig unabhängige Anwendungen zu implementieren. Grundsätzlich hätte dabei individuelle Methodik und Technik zum Einsatz kommen können.

+ Keine Vorbereitung zur Umsetzung der Anwendungen erforderlich
+ Größere Flexibilität in der Realisierung der Anwendungen
− Keine Wiederverwendung von wiederkehrenden Elementen

### 8.1.2 Anwendungen als Erweiterungen des Mission Profile Framework

Anstelle der Plattform-Lösung hätte eine Erweiterung des Mission Profile Framework (MPF) implementiert werden können, die bis auf einige Einschränkungen den Funktionsumfang der Plattform erreicht hätte. Die Erweiterung hätte dann den, zum Zeitpunkt der Entwicklung jedoch noch rudimentär ausgeprägten, Plug-in Mechanismus dieses Rahmenwerks nutzen können. Auch wenn diese Möglichkeit in Betracht gezogen wurde, so erschwerte dann allerdings die stagnierende Entwicklung des MPF eine konkrete Umsetzung dieses Ansatzes.

+ Wiederverwendung bestimmter MPF Funktionen zur MP Verwaltung
− Hoher Wartungsaufwand aufgrund fortschreitender MPF Entwicklung
− Keine Unterstützung von Änderungsverwaltung im MPF
− Keine Unterstützung von Ontologien im MPF

### 8.1.3 Alleinstehendes Rahmenwerk

Statt einer Plattform hätte auch ein Rahmenwerk entwickelt werden können, um Anwendungen zu realisieren. Dieser ebenfalls in Hinsicht auf die spätere Beschleunigung der Entwicklung in Kombination mit der stärker ausgeprägteren Kompatibilität

attraktive Ansatz hätte jedoch im Unterschied zur Plattform lediglich eine gemeinsame, wiederverwendbare Struktur zur Realisierung von Anwendungen dargestellt [152]. Eine vollständige Abstraktion über die jeweiligen MP Formate bzw. MPFO-Versionen im Speziellen, wäre zwar denkbar, jedoch keine Notwendigkeit für ein Rahmenwerk.

+ Wiederverwendung bestimmter Funktionen
– Keine zwingende Abstraktion über MP Formate
– Schlechtere Einbindung in existierende Arbeitsabläufe, da ein Rahmenwerk keine eigenständig lauffähige Anwendung darstellt

## 8.2 Architektur

Grundlegend für das hier beschriebene System, sowie dessen Peripherie, ist eine Client-Server-Architektur. Diese ermöglicht den Aufbau eines zentralen Verzeichnisses von MP und ReqIF Ontologien, auf der Server-Seite, was deren Verwaltung und Verarbeitung vereinfachen soll. Clients fungieren dabei als Datenlieferanten sowie als -bezieher von Verarbeitungsergebnissen. Durch eine einheitliche Schnittstelle sind Server und Client voneinander getrennt und dadurch austauschbar. Diese Schnittstelle wird in Abschnitt 8.3 näher beschrieben.

Als Beitrag zur Erreichung von Ziel (3), wie in Abschnitt 4.2 definiert, werden zur Verbesserung der Interoperabilität zwischen Systemen Zugriffs- und Manipulationsoperationen ausschließlich auf HTTP-Abfragemethoden eingeschränkt. Dies verringert die Komplexität der Kommunikation zwischen den Systemen und das HTTP-Protokoll ist üblicherweise in Arbeitsplatzsystemen und -umgebungen auch zulässig. Die Basistechnologie für die Implementierung der Plattform sollte somit Representational State Transfer (REST)-Konformität aufweisen. Das eingesetzte Play Framework folgt dem REST-Architekturstil in Hinsicht auf seine Web-Funktionalität und -Schnittstellen.

Das Play Framework setzt zudem zur Realisierung von Anwendungen auf das Model-View-Controller (MVC) Architekturmuster. Das MVC Architekturmuster ermöglicht es, Datenmodelle, Darstellung und Steuerungslogik in der Entwicklung sauber zu trennen und zu verbinden. Dieser modulare Entwurf ermöglicht auch ein einfaches, späteres Austauschen von entsprechenden, diese Aspekte realisierenden Komponenten, was einen Beitrag zur Erfüllung der Anforderung P4 darstellt.

Um eine konkrete MPAD Anwendung zu realisieren, ist die Plattform dann um entsprechende benötigte Subkomponenten zu erweitern. Dies kann und sollte unter Wiederverwendung der existierenden Subkomponenten geschehen. Abbildung 8.1 zeigt den modularen Aufbau der Plattform aus Komponentensicht. Dabei stellt die durch eine gestrichelte Linie markierte Gruppe (rechts unten im Block der Plattform)

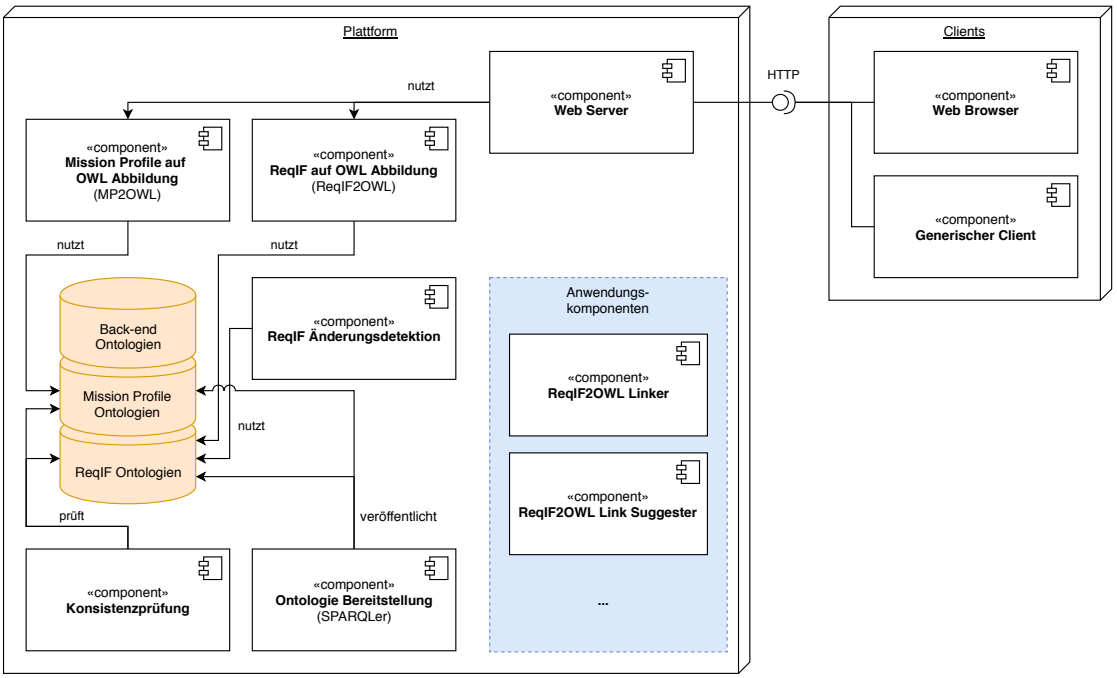



**Abbildung 8.1** – UML Komponentendiagramm, welches die Gesamtheit der einzelnen Subkomponenten der Plattform, sowie deren Beziehungen zueinander darstellt

Subkomponenten dar, welche für die Umsetzung einer MPAD Anwendung, die in Abschnitt 6 beschrieben wird, von Bedeutung sind.

## 8.3 Systemintegrationsmittel

Um die in Abschnitt 4.3.2 definierte Anforderung P6 zu adressieren, wurden verschiedene Maßnahmen ergriffen, welche in diesem Abschnitt erläutert werden. Diese Maßnahmen haben die Absicht, zunächst die Bereitstellung auf unterschiedlichen Betriebssystemen und schließlich Interaktionen zwischen der Plattform und Anwendern und weiteren Systemen zu ermöglichen. Dadurch wird die Möglichkeit der Einbindung der Plattform in existierende Arbeitsabläufe erreicht.

### 8.3.1 Minimierung von Nutzungsvoraussetzungen

Um Barrieren bei der Bereitstellung der Anwendung zu vermeiden, wird auf der Server-Seite lediglich eine einzige Abhängigkeit vorausgesetzt: Das System muss über eine Java Runtime Environment (JRE) verfügen. Es wäre technisch möglich, eine JRE mitzuliefern, sodass auch dies ausgeräumt wäre. Jedoch ist das aus

sicherheitstechnischer Sicht nicht ratsam, da eine JRE regelmäßig aktualisiert werden und auch zum Zeitpunkt der Installation der Anwendung aktuell sein sollte.

Auf der Client-Seite gibt es keine Voraussetzungen zur Nutzung, außer entweder das Vorhandensein eines standardkonformen Browsers oder einer anderen generischen HTTP-fähigen Software, sofern es sich zum Beispiel nicht um einen menschlichen Benutzer handelt. Die HTTP-Clients benötigen darüber hinaus keinerlei zusätzliche Plug-ins oder Ähnliches, um die Anwendung nutzen zu können.

### 8.3.2 Einheitliche Schnittstelle

Die Plattform stellt eine einheitliche Schnittstelle über HTTP bereit. Diese kann dann sowohl direkt von Benutzern über einen Browser genutzt werden, wie auch von anderen generischen Clients. Generische Clients könnten bspw. Electronic Design Automation (EDA) Werkzeuge, Messequipment etc. sein. Solche Clients werden im Folgenden *Anwendungs-Clients* genannt. Die einheitliche Schnittstelle vereinfacht insgesamt die Implementierung der Plattform.

Die Plattform verfügt über einen Mechanismus, um zwischen Benutzer-Clients und Anwendungs-Clients zu differenzieren. Die unterschiedlichen Ausgabemöglichkeiten in HTML in Verbindung mit JavaScript und JavaScript Object Notation (JSON) bzw. RDF/XML werden in Abhänigkeit vom Client-Typ genutzt. So erhalten Benutzer-Clients direkt eine HTML- und JavaScript-Ansicht, während Anwendungs-Clients direkt mit JSON bzw. XML Inhalten arbeiten können.

### 8.3.3 Datenformate und Kommunikationsprotokolle

Bei den Datenformaten und Kommunikationsprotokollen wurde darauf geachtet – so weit möglich – Standards einzusetzen. Wie im vorherigen Abschnitt erwähnt, wird so ausschließlich HTTP als Kommunikationsprotokoll verwendet. In Verbindung damit stehen außerdem XML und JSON als Datenformate, für Parameterüber- und Ergebnisrückgaben. Für Anforderungsdokumente wird ebenfalls der etablierte Standard ReqIF verwendet. Einzig bei MPs wird das zum Zeitpunkt noch in Entwicklung befindliche MPFO genutzt, welches jedoch zum Ende dieser Untersuchungen ebenfalls in den Standardisierungsprozess übergegangen ist. Alle Resultate der Verarbeitung von MP- oder Anforderungsdokumenten werden mittels OWL dargestellt.

## 8.4 Grundfunktionen

Wesentliche Bestandteile der Plattform sind Werkzeuge zur Abbildung von MP Dokumenten und von Anforderungsdokumenten auf korrespondierende Ontologien. Weiterhin stellt die Plattform ein Werkzeug bereit, welches Änderungen zwischen

verschiedenen Versionen von Anforderungsdokumenten im ReqIF Format erkennen kann. Auch dieses Werkzeug gibt erkannte Änderungen in Form von Ontologien aus. Für eine Übersicht über alle Basis-Komponenten der Plattform sei auf Tabelle 8.1 verwiesen.

Die konsequente Verwendung der standardisierten Ontologiesprache OWL vereinfacht den Einsatz von Reasoning bei der Verarbeitung von MPs und Anforderungen und erlaubt die Verknüpfung der Daten mit anderen Wissensquellen.

| Funktionsart | Zugehörige Komponenten |
|---|---|
| Allgemeine Verwaltung | Back-end Ontologien Verwaltung<br>MP Ontologien Verwaltung<br>ReqIF Ontologien Verwaltung |
| Abbildung | MP2OWL, ReqIF2OWL |
| Änderungsverwaltung | ReqIFCompare, OWLDiff |
| Persistenz | SPARQLer, Persistence |

**Tabelle 8.1** – Plattformkomponenten nach Funktionsart

## 8.5 Module und Bibliotheken der Plattform

Die Plattform verwendet drei Module, welche zur Laufzeit mittels Guice[2] eingebunden werden. Tabelle 8.2 zeigt eine Übersicht über die Module. Diese Module stellen lediglich die Kern-Module dar. Weitere Module, die zu Plattform-Anwendungen gehören, sind in der Liste nicht enthalten. Selbes gilt für die Bibliotheken, welche in Tabelle 8.3 aufgelistet sind.

| Modul | Funktionsbeschreibung |
|---|---|
| ChangeDetection | Prüft Anforderungskataloge auf Änderungen |
| ConsistencyChecks | Prüft Ontologien auf Konsistenz |
| DatabaseManagement | Verwaltung einer internen Datenbank |

**Tabelle 8.2** – Übersicht über Kern-Plattform-Module

[2]Siehe `https://github.com/google/guice`

| Bibliothekname | Funktionsbeschreibung |
|---|---|
| MP2OWL | Bildet MPs im MPFO auf Ontologien ab |
| OWLDiff | Vergleicht OWL Ontologien und stellt Änderungen fest |
| Persistence | Intern verwendete Datenbank für Persistenz |
| ReqIF | Schnittstelle zur Verarbeitung von ReqIF Dokumenten |
| SPARQLer | Bereitstellung von Ontologien auf SPARQL Endpoints |

**Tabelle 8.3** – Übersicht über Kern-Plattform-Bibliotheken

## 8.6 REST-Schnittstelle der Plattform

Die primäre externe API der Plattform ist eine REST-Schnittstelle. Tabelle 8.4 zeigt eine Übersicht über die Kernfunktionen dieser Schnittstelle. Diese Angaben beziehen sich nur auf die Basisfunktionalität der Plattform. Plattform-Anwendungen können zusätzliche Funktionen bereitstellen.

| Pfad | HTTP Methode | Beschreibung |
|---|---|---|
| / | GET | Liefert Index Seite |
| /public/A | GET | Liefert Medium A |
| /ontologies | GET | Liefert eine Liste von Ontologien |
| /addOntology | GET | Formular zum Hinzufügen einer Ontologie |
| /ontology | POST | Hinzufügen einer Ontologie |
| /ontology/B | GET | Abrufen von Ontologie B |
| /expose | POST | Bereitstellung einer Ontologie |
| /mp | GET | Liefert MP Ontologien Liste |
| /mp/C | GET | Liefert Mission Profile Ontologie C |
| /mp | POST | Hinzufügens eines MP |
| /remove-mp | POST | Entfernen einer MP Ontologie |
| /consistent-mp | POST | Konsistenzprüfung einer MP Ontologie |
| /reqs | GET | Liefert Liste von Anforderungsontologien |
| /reqs/D | GET | Liefert Anforderungsontologie D |
| /reqs | POST | Hinzufügen von Anforderungskatalogen |
| /remove-reqs | POST | Entfernen von Anforderungsontologien |

**Tabelle 8.4** – REST-Schnittstelle der Plattform (Basisfunktionalität)

## 8.7 Mission Profile Format Einsatz

Da im Zeitraum der Entwicklung der Plattform noch kein standardisiertes Format für MPs existierte, wurde eine Version des MPFO Entwurfs, welcher im Projekt autoSWIFT entwickelt wurde, herangezogen. Grundlage für die Arbeiten war die bisher nicht veröffentlichte MPFO Version 0.2.1, siehe Abbildung 8.2.

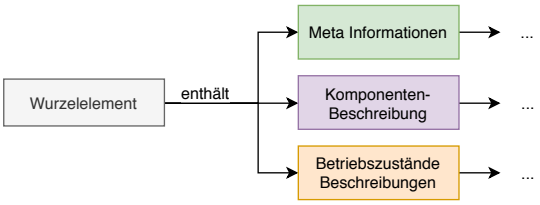



**Abbildung 8.2** – Grundlegende Elemente im MPFO Version 0.2.1

## 8.8 Mission Profile Framework Anbindung

Um die Anbindung an das MPF zu ermöglichen, wurde der Ansatz verfolgt, Bindungen für die Simple Object Access Protocol (SOAP)-Schnittstelle des MPF über die Web Services Description Language (WSDL) zu generieren, siehe Abbildung 8.3. Für die Generierung der Bindungen wurde das Werkzeug *scalaxb*[3] verwendet. Dieses Werkzeug generiert einer WSDL oder XSD Datei entsprechenden, ausführbaren Java Code, welcher es dann ermöglicht, die Funktionen der SOAP-Schnittstelle zu nutzen. Dieser generierte Code kann dann bspw. in einer Bibliothek bereitgestellt werden.

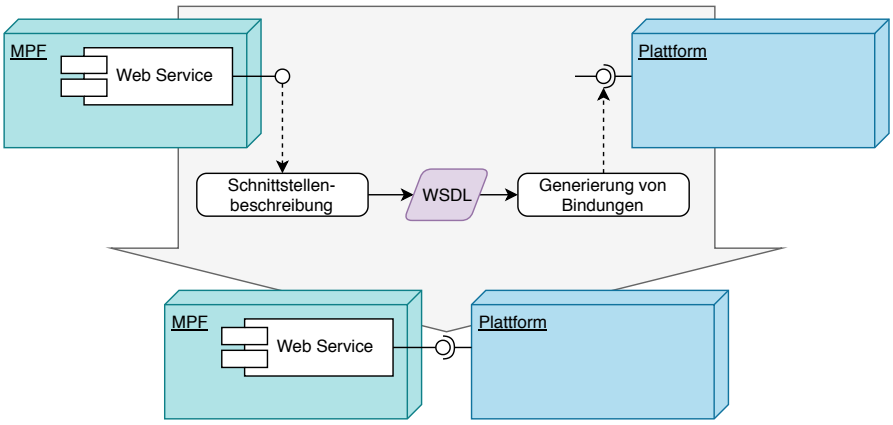



**Abbildung 8.3** – Anbindung an das MPF mittels WSDL

[3]Siehe `http://scalaxb.org/`

## 8.9 Mission Profile Abbildung

Um eine Abbildung von MPs auf Ontologien realisieren zu können, wurden verschiedene Ansätze untersucht. Die Methoden sind [153] entnommen, wurden aber noch ergänzt. Im Vorfeld wurde eine Einschätzung vorgenommen, wie die Ansätze sich in Hinsicht auf Aufwand und Wartbarkeit verhalten, siehe Tabelle 8.5.

| **Methode** | Aufwand | Wartbarkeit |
|---|---|---|
| Transformationssprachen | ++++ | ++++ |
| XSL Transformation | ++ | + |
| Information Integration | - | ++ |
| Direct Semantic Programming | - | ++ |
| Indirect Semantic Programming | - | +++ |
| Ontology Mapping | ++ | – |
| Semantic Annotations | ++++ | + |

**Tabelle 8.5** – Einschätzung des Implemtierungsaufwands und der Wartbarkeit von Abbildungsmethoden

Da die Wartbarkeit größer gewichtet wurde, wurden nur Ansätze, die mindestens mit einem Plus bewertet wurden, experimentell umgesetzt. Im Folgenden soll argumentiert werden, dass sich der Indirect Semantic Programming (ISP) Ansatz am besten für die Implementierung der Abbildung von MPs in Ontologien eignet.

### 8.9.1 Modelltransformation

Im Rahmen der Untersuchungen, wie MPs auf Ontologien abgebildet werden können, wurden auch die Modelltransformationssprachen ATL [139], QVT [138] und Epsilon Transformation Language (ETL) [154] eingesetzt. Aufgrund der Komplexität der MP Metamodelle entstanden dabei allerdings massive Schwierigkeiten bei der konkreten Umsetzung von Transformationen, die sich auf Fehler in den damals zur Verfügung stehenden Implementierungen dieser drei Transformationssprachen zurückführen lassen. Angesichts dessen wurde dieser Ansatz dann nicht weiter verfolgt und weitere Möglichkeiten zur Transformation von MPs untersucht.

### 8.9.2 XSL Transformation

Die Grundidee bei der XSL Transformation (XSLT) ist, sich die Tatsache zunutze zu machen, dass das MPFO auf XML basiert und sich OWL ebenfalls in RDF/XML serialisieren lässt. XSLT sollte also genutzt werden, um ein XML Dokument in ein anderes XML Dokument transformieren. Leider zeigte sich bei der Implementierung,

dass dies nicht praktikabel in Hinsicht auf die Wartbarkeit ist. Dies lässt sich im Wesentlichen auf das Fehlen der Bildung und Nutzung eines Zwischenmodells, wie beim Direct Semantic Programming (DSP) und ISP zurückführen. Das Prinzip der implementierten XSLT ist es, gezielt Daten aus dem MP XML Quelldokument in vordefinierte Vorlagen für entsprechende OWL Konstrukte einzusetzen, siehe Quelltext 8.1 für ein Beispiel. Weiterer Nachteil ist die umständliche Anbindung weiterer Verarbeitungsmechanismen. Soll etwa eine komplexe mathematische Operation zur Verarbeitung der MP Daten, die bereits in einer anderen Sprache realisiert ist, zum Einsatz kommen, muss dies getrennt von der eigentlichen XSLT geschehen. Auch nach Sendall und Kozaczynski eignet sich XSLT aufgrund der damit verbundenen Komplexität der Implementierung von Transformationen nicht für die Modelltransformation [86].

```
<xsl:for-each select="mp:DocumentHeader/mp:MountingPoint">
  <xsl:variable name="i" select="position()"/>
  <owl:NamedIndividual rdf:about="http://localhost:8080/autoSWIFT
     /missionProfile#MountingPoint{$i}">
    <rdf:type rdf:resource="http://localhost:8080/autoSWIFT/
       missionProfile#MountingPoint"/>
    <missionProfile:hasID rdf:datatype="http://www.w3.org/2001/
       XMLSchema#string"><xsl:value-of select="@mp:ID"/></
       missionProfile:hasID>
    <missionProfile:hasName rdf:datatype="http://www.w3.org/2001/
       XMLSchema#string"><xsl:value-of select="@mp:Name"/></
       missionProfile:hasName>
    <missionProfile:hasDescription rdf:datatype="http://www.w3.
       org/2001/XMLSchema#string"><xsl:value-of select="
       @mp:Description"/></missionProfile:hasDescription>
    <missionProfile:hasPropertySet rdf:resource="http://
       localhost:8080/autoSWIFT/missionProfile#PropertySet1"/>
  </owl:NamedIndividual>
</xsl:for-each>
```

**Quelltext 8.1** – Auszug des XSL Templates zur Transformation von Mission Profile Dokumenten in eine entsprechende OWL Ontologie

### 8.9.3 Information Integration

Beim Information Integration Ansatz wird das Vorgehen verfolgt, dass Informationen aus einem MP Dokument in eine Ontologie in entsprechende vordefinierte Stellen eingefügt werden. Die Wartbarkeit bei diesem Ansatz ist höher, als beim XSLT Ansatz, jedoch ist auch der Aufwand höher. Grund dafür ist, dass zum einen vorausgesetzt wird, dass die Ontologie, in der Informationen zusammengeführt werden sollen, bereits umfassendene Strukturen zur Integration der Informationen

bereitstellt. Es müssten also viele Vorlagen für entsprechende Datenstrukturen der MPs erstellt werden. Aufgrund des Aufwandes wurde dieser Ansatz nicht experimentell evaluiert.

### 8.9.4 Ontology Mapping

Da das Ontology Mapping veraussetzt, dass bereits eine Ontologie existiert, die dann auf eine andere abgebildet wird, eignet sich dieses Verfahren nicht für die unmittelbare Abbildung von MPs nach dem MPFO auf Ontologien. Jedoch ist eine Kombination mit der in Abschnitt 8.9.7 angeführten Bibilothek *EMFTriple* denkbar. Diese erzeugt eine RDF Serialisierung von Eclipse Modeling Framework (EMF) Modellen. Die Methode bestünde dann darin, zunächst eine primitive RDF Serialisierung eines MPs zu erstellen und diese dann als Quellontologie im Rahmen des Ontology Mapping Prozesses zu betrachten. Aufgrund des Aufwandes und der Voraussetzung der vorherigen Realisierung einer Abbildung mittels ISP wurde dieser Ansatz nicht experimentell evaluiert.

### 8.9.5 Semantic Annotations

Bei diesem Ansatz war die Idee, MP Dokumente mit semantischen Annotationen zu versehen, so dass diese dann einfacher in Ontologien überführt werden können. Dieser Ansatz konnte aufgrund von technischen Limitierungen des MPF nicht genutzt werden. Die Limitierung bestand darin, dass der zum Zeitpunkt verwendete Dienst *Spyne*[4] des MPF das Hinzufügen von Annotationen nicht unterstützte. Außerdem stellt diese Möglichkeit auch insgesamt mehr einen Zwischenschritt, als eine vollständige Lösung für die Abbildung dar.

### 8.9.6 Direct Semantic Programming

Der DSP Ansatz, wie in [155] beschrieben, unterscheidet sich von ISP im Wesentlichen dadurch, dass Ontologie-Entitäten direkt auf Sprachkonstrukte einer Programmiersprache, welche das Programmierparadigma Objektorientierte Programmierung (OOP) unterstützt, abgebildet werden (bspw. Ontologie-Klasse auf Java-Klasse). Dieser Ansatz erfordert also, dass es für jede Ontologie-Entität eine Entsprechung in der Programmiersprache gibt. Außerdem erschwert dieser Ansatz die Entwicklung Domänen-neutraler Software [155]. Zum Zeitpunkt der Entwicklung war zudem waren sowohl Auswahl, wie auch die Reife existierender Bibliotheken zur Realisierung eines DSP Ansatzes zur Abbildung von MPs auf Ontologien sehr begrenzt, weshalb stattdessen der ISP Ansatz verfolgt wurde, siehe nächster Abschnitt 8.9.7.

[4]Siehe `http://spyne.io`

### 8.9.7 Indirect Semantic Programming

Der ISP Ansatz stellte sich als geeignetster Ansatz zur Implementierung der Abbildung von MPs auf Ontologien heraus. Ein Grund dafür liegt darin, dass zum Zeitpunkt der Entwicklung bereits etablierte, ausgereifte Bibliotheken zur Manipulation von Ontologien existierten, die dem ISP Prinzip folgten – insbesondere die OWL API[5] [156–158], welche dann auch zur Implementierung genutzt wurde und JENA [159]. Weiter ist ISP besser geeignet, um Domänen-neutrale Software zu entwickeln [155] und fordert nicht die Existenz korrelierender Sprachkonstrukte in der zur Implementierung eingesetzten Programmiersprache. Konkret wurde dieser Ansatz unter der Nutzung der OWL API und EMF[6] implementiert. Zunächst werden dabei die XML MPs deserialisiert und dadurch in Modelle überführt. Anschließend erfolgt eine Serialisierung mittels OWL API, siehe Abbildung 8.4.

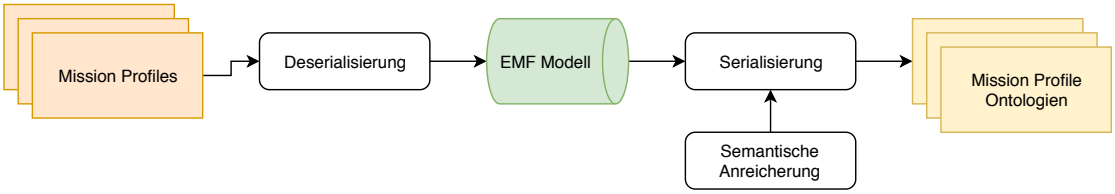



**Abbildung 8.4** – Schema zur MP Abbildung mittels EMF, als ISP basierter Ansatz

Die TBox einer auf diese Weise entstandene Ontologie wird in Abbildung 8.5 dargestellt. Während die Deserialisierung und Modellbildung vollständig automatisiert mit EMF [160] bewerkstelligt werden kann, muss die Serialisierung mithilfe der OWL API zum Großteil manuell erfolgen. Diesen Schritt zu automatisieren wurde ebenfalls untersucht. Dabei wurde die Bibliothek *EMFTiple*[7] eingesetzt, welche RDF-Bindungen für EMF-Modelle bereitstellt. Leider umfasst die EMFTriple-Implementierung allerdings keine Vollständige Unterstützung für alle EMF-Konstrukte. Insbesondere EMF *FeatureMaps* wurden von EMFTriple nicht unterstützt, was jedoch zwingende Voraussetzung für die vollständige Serialisierung der EMF-Modelle war. Aus diesem Grund wurde die Serialisierung dann manuell implementiert und dadurch eine weitgehende und semi-automatische Abbildung der MPs auf Ontologien realisiert. Eine manuelle Implementierung der Serialisierung im Vergleich zur Abbildung von MPs mittels EMFTriple auf Ontologien bietet außerdem die Möglichkeit resultierende Ontologien zusätzlich semantisch anzureichern.

[5] Siehe `http://owlcs.github.io/owlapi/`
[6] Siehe `http://www.eclipse.org/modeling/emf/`
[7] Siehe `https://github.com/ghillairet/emftriple`

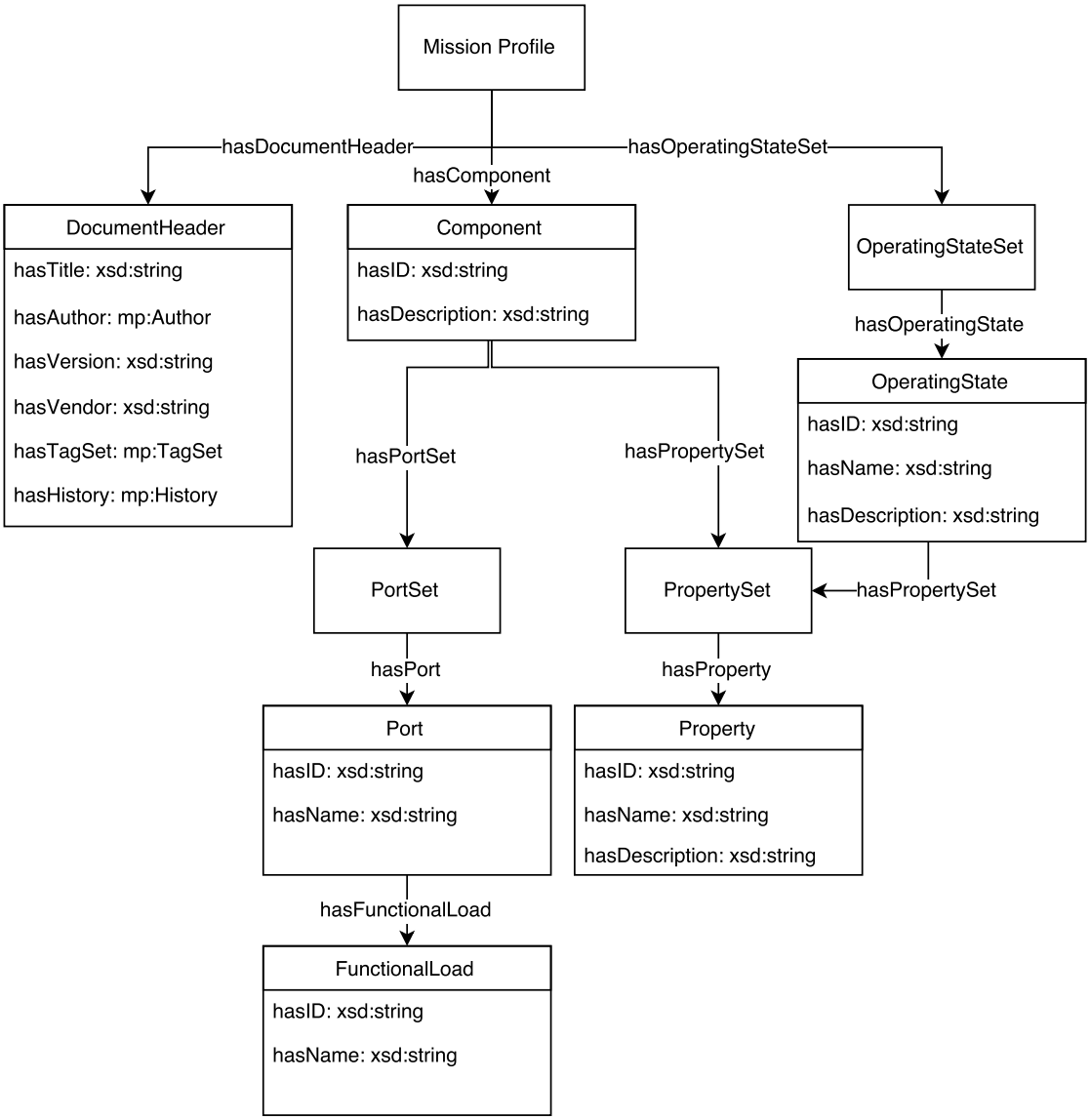



**Abbildung 8.5** – Auszug der aus dem ISP Ansatz resultierenden TBox der MP Ontologie als UML Klassendiagramm

## 8.10 Vorschlagen von Verknüpfungen

Um Benutzer im Prozess des Spezifizierens von Verknüpfungen in der in Kapitel 6 vorgestellten Ontologie-gestützten Change Impact Analyse zu unterstützen, enthält das vorgeschlagene System eine Komponente, die dem Benutzer Kandidaten für Verknüpfungen zwischen Anforderungen und Entwurfsparametern liefern kann, siehe Abbildung 8.6.

Der Benutzer kann einen vorgeschlagenen Kandidaten entweder annehmen oder ablehnen. Es wurde eine exemplarische Kandidatenauswahl implementiert, bei der Anforderungen und Entwurfsparameter als natürlichsprachliche Beschreibungen angenommen werden. Die Idee hinter dieser Implementierung ist, WordNet [161] zu nutzen, um verwandte Synonyme in Anforderungs- bzw. Entwurfsparameterbeschreibungen zu identifizieren. Für die Verarbeitung der natürlichsprachlichen Beschreibungen wurde *Stanford CoreNLP* [162] eingesetzt. Algorithmus 1 beschreibt den implementierten Mechanismus, auf der Basis der zuvor angeführten Idee.

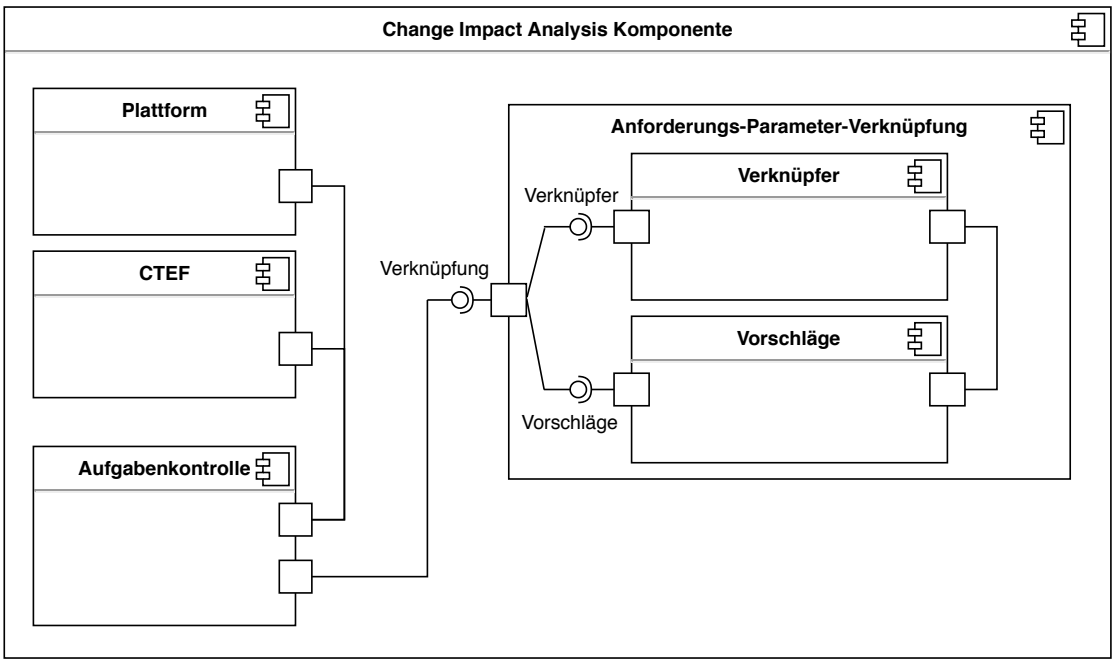



**Abbildung 8.6** – UML Komponentendiagramm des den vorgestellten Ansatz realisierenden Systems. Alle Komponenten sind über eine zentrale Kontrollkomponente verbunden und werden durch diese gesteuert. Zu beachten ist auch die Schnittstelle zur *Anforderungs-Parameter-Verknüpfung* Komponente, welche eingeführt wurde um mit weiteren Ansätzen zur Verknüpfung experimentieren zu können

Auch wenn die Leistung des Algorithmus weiter optimierbar ist und dieser bspw. um den Vergleich von Editierdistanzen von Nomen in den Zeilen 8-14 erweitert werden könnte, um bspw. Schreibfehler in Beschreibungen handhaben zu können, so kann er eingesetzt werden, um Kandidaten für Verknüpfungen zu liefern. Dies deckt nicht alle Szenarien ab, sondern nur solche, in denen natürlichsprachliche Beschreibungen von Anforderungen und Entwurfsparametern vorhanden sind.

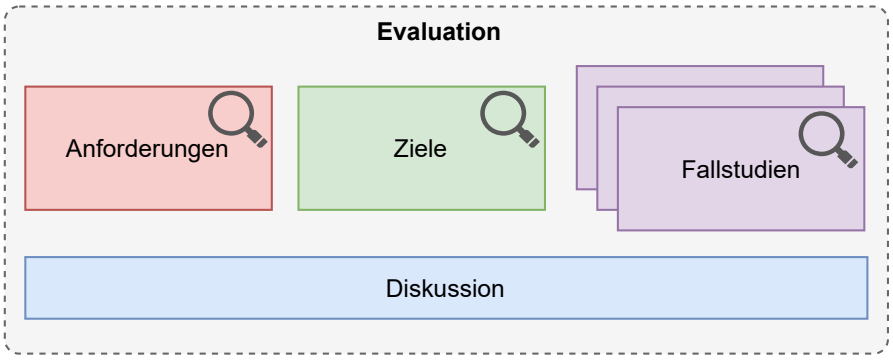
**Data:** requirement description $r_i \in \mathcal{R}$, parameter description $p_j \in \mathcal{P}$, $i, j \in \mathbb{N}_0$, $0 \leqslant i \leqslant |\mathcal{R}|$, $0 \leqslant j \leqslant |\mathcal{P}|$
**Result:** set of link suggestions $\mathcal{L}$

```
foreach r_i ∈ R do
    foreach p_j ∈ P do
        tokenize r_i and p_j ;
        part-of-speech tagging of all tokens ;
        recognize lemmas of all tokens ;
        foreach token t of tokens of r_i do
            foreach token s of tokens of p_i do
                if t and s are nouns then
                    get lemma l of t and k of s ;
                    get the set S of symmetric relationships between l and k ;
                    if S is not empty then
                        add an entry to the set of link suggestions L with a link
                          between r_i and p_i ;
                    end
                end
            end
        end
    end
end
```

**Algorithmus 1:** Verknüpfungsvorschläge

# 9 Evaluation

Dieses Kapitel stellt dar, in welchem Umfang die in den Abschnitten 4.3 und 4.4 definierten Anforderungen durch die Plattform und die Anwendungen erfüllt wurden. Weiterhin wird beschrieben, wie die in Abschnitt 4.2 definierten Ziele erreicht werden konnten. Außerdem werden Fallbeispiele dargelegt, welche auf den Anwendungen der Plattform basieren. Zur Evaluation über die Fallbeispiele werden die in den Kapiteln 5, 6 und 7 beschriebenen Anwendungen herangezogen. Anhand von insgesamt fünf Fallbeispiele, von denen sich je Anwendung mindestens eine auf eine Anwendung bezieht, wird die praktische Anwendbarkeit der Methoden untersucht. Abgeschlossen wird dieses Kapitel von einer Diskussion der Ergebnisse.




## Abschnitte

## 9.1 Anforderungserfüllung

In diesem Abschnitt wird dargestellt, ob und in welcher Form die in den Abschnitten 4.3 und 4.4 definierten Anforderungen erfüllt werden.

### 9.1.1 Anforderungserfüllung in Bezug auf die Plattform

P1 Integration heterogener Daten beziehungsweise Modelle über Ontologien

Diese Anforderung erfüllt die Plattform durch den konsequenten Einsatz von OWL-Ontologien zur Darstellung aller Modelle, sowie durch die Bereitstellung von entsprechenden Abbildungsmechanismen für MPs und Anforderungskatalogen.

P2 Verknüpfung von MP-Daten mit Anforderungen

Durch die Darstellung von MPs im MPFO-Format und Anforderungen im ReqIF-Format in OWL-Ontologien wird eine Verknüpfung von MP-Daten mit Anforderungen ermöglicht. Bereits das RDF, auf welchem auch OWL basiert, stellt Mittel bereit um beliebige Ressourcen miteinander zu verknüpfen, siehe dazu Abschnitt 2.1.2.2.

P3 Automatisches Schlussfolgern mittels Ontologien

Da die Plattform auf der Basis von OWL-Ontologien arbeitet, ist der Einsatz von etablierten Inferenzmaschinen möglich, wodurch diese Anforderung als erfüllt betrachtet werden kann.

P4 Identifikation von Unterschieden zwischen Eingabe-Modellen

Die Plattform kann Unterschiede zwischen Eingabe-Modellen durch die *OWL-Diff*-Komponente feststellen. Insbesondere kann für ein Paar von MPFO- beziehungsweise ReqIF-Dokumente sowohl die Menge von hinzugekommenen, entfernten und auch von modifizierten Axiomen beziehungsweise Anforderungen aufgestellt werden. Darüber hinaus können auf der Basis der identifizierten Änderungen auszuführende Aktionen definiert werden.

P5 Bereitstellung von resultierenden Ontologien

Die Komponente *SPARQLer*, welche Bestandteil der Plattform ist, ist zuständig für die Bereitstellung von Ontologien auf beliebigen SPARQL Endpoints. Neben gewöhnlichen *ASK*, *SELECT* und *UPDATE* Anfragen gegen SPARQL Endpoints, unterstützt diese Komponente auch die Übertragung vollständiger Ontologien mittels einer SPARQL *LOAD* Anfrage.

P6 Erweiterbarkeit

Die Erweiterbarkeit der Plattform wird primär durch die Definition vereinheitlichender Schnittstellen erreicht. Diese ermöglichen außerdem auch die Austauschbarkeit von Komponenten.

P7 Integrierbarkeit

Es wurden verschiedene Maßnahmen ergriffen, um die Plattform in existierende Arbeitsabläufe integrierbar zu machen. Diese sind in Abschnitt 8.3 wiedergegeben.

### 9.1.2 Anforderungserfüllung in Bezug auf die Anwendungen

A1 Unterstützung von MPAD

Alle drei in dieser Arbeit angeführten Plattform-Anwendungen unterstützen MPAD durch die durchgängige Berücksichtigung und Inklusion von MPs.

A2 Möglichkeit zur Integration heterogener Datenquellen

Die drei Plattform-Anwendungen ermöglichen die Integration heterogener Datenquellen durch den Einsatz von OWL Ontologien. Dabei wird das in Abschnitt 2.1.2.2 beschriebene universelle Konzept zur Verknüpfung beliebiger *Ressourcen* genutzt.

A3 Bereitstellung von Wissen in standardisierter Form

Sämtliche Resultate der drei Plattform-Anwendungen werden in OWL Ontologien dargestellt. Dadurch wird gewährleistet, dass das Wissen in standardisierter Form bereitgestellt werden kann.

## 9.2 Zielerreichung

In diesem Abschnitt wird dargestellt, ob und in welcher Form die in den Abschnitten 1.1 definierten Ziele erfüllt werden.

(1) Möglichkeit der Integration heterogener Daten beziehungsweise Modelle

Die Plattform ermöglicht eine Integration heterogener Daten beziehungsweise Modelle durch das Prinzip der Abbildung auf OWL-Ontologien. Die OWL-Ontologien ermöglichen es korrelierende Bestandteile der Daten beziehungsweise Modelle explizit zu verknüpfen. Die Plattform stellt in Hinsicht auf MPAD die Integration von MPs und Anforderungen durch deren gemeinsame Abbildung auf entsprechende OWL Ontologien bereit. Diese Abbildung wird von den dafür zuständigen Komponenten *MP2OWL* und *ReqIF2OWL* durchgeführt.

(2) Automatisches Schlussfolgern

Das automatische Schlussfolgern wird ermöglicht durch den Einsatz von etablierten Inferenzmaschinen, die auf der Basis der OWL-Ontologien arbeiten. Durch den Einsatz der standardisierten Ontologie-Sprache OWL ist ein Austausch der eingesetzten Inferenzmaschine möglich.

(3) Unterstützung von Änderungsverwaltung

Zur Unterstützung von Änderungsverwaltung stellt die Plattform eine Komponente *OWLDiff* bereit. Diese ermöglicht es, Unterschiede zwischen OWL-Ontologien zu identifizieren. Darüber hinaus wird eine weitere Komponente *ReqIFCompare* bereitgestellt, die Unterschiede zwischen ReqIF-Dokumenten identifizieren kann.

(4) Ermöglichung der Einbindung in existierende Arbeitsabläufe

Eine Einbindung der Plattform in existierende Arbeitsabläufe wird primär durch die Verwendung standardisierter Datenmodelle und -formate erreicht. Ferner sind, wie in Abschnitt 8.3 beschrieben, weitere Mittel zur Systemintegration realisiert worden.

Für die Entwicklung der Plattform ergibt sich somit, dass die Zielsetzung erfüllt werden konnte. Die Plattform stellt die einheitliche Basis für die Entwicklung von MPAD-Anwendungen dar und bildet somit den Kernbeitrag zur Erreichung der in Abschnitt 1.1 definierten Ziele. Eine Aufstellung identifizierter Vor- und Nachteile des Einsatzes von Ontologien im Umfeld des Automobilelektronikentwurfs folgt in Abschnitt 9.4.

## 9.3 Fallbeispiele

Zunächst werden in den Abschnitten 9.3.1 und 9.3.2 Fallbeispiele angeführt, welche sich auf die Anwendung in Kapitel 5 beziehen. Diesen folgt ein Fallbeispiel zur in Kapitel 6 vorgestellten Anwendung im Abschnitt 9.3.3. Schließlich werden zwei weitere Fallbeispiele zur in Kapitel 7 vorgestellten Anwendung beschrieben. Abbildung 9.1 zeigt eine Übersicht über die Anwendungen und zugehörigen Fallbeispiele.

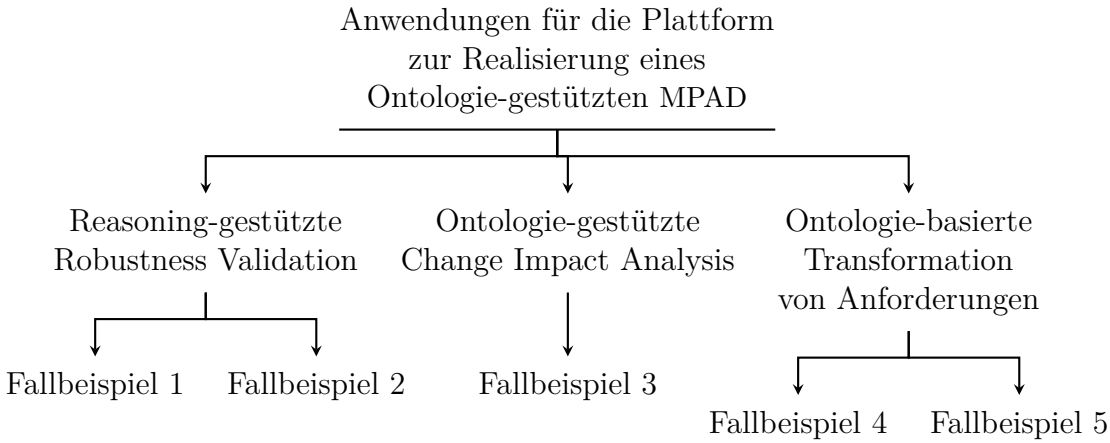



**Abbildung 9.1** – Überblick über die Fallbeispiele

### 9.3.1 Fallbeispiel 1: Robustness Validation Plan Generierung

Dieses Fallbeispiel bezieht sich auf den in Kapitel 5 beschriebenen Ansatz zur Reasoning-gestützten RV.

Der ON Semiconductor® FTCO3V455A1[1] ist ein 3-Phasen Inverter für die Anwendung im Automobil. Dieser kann eingesetzt werden, um elektronisches sowie elektro-hydraulisches Lenken zu realisieren, in elektronischen Wasserpumpen und elektronischen Ölpumpen. Um Robustheit nach dem „Handbook for Robustness Validation of Automotive E/E Modules“ [10] zu gewährleisten, muss ein RV-Plan erstellt werden. Teil dieses Plans ist eine Menge von Analysen, welche durchzuführen sind.

Zunächst wird das Wissen über diese Analysen und die Kriterien, die zur Auswahl der entsprechenden Analysen herangezogen werden, in einer OWL-Ontologie mittels dem Softwarewerkzeug Protégé formalisiert. Die Fakten werden dazu als allgemeine Klassenaxiome der Ontologie hinzugefügt, siehe Quelltexte 9.1 und 9.2.

[1] Siehe `https://www.onsemi.com/PowerSolutions/product.do?id=FTCO3V455A1`

```
hasPinCountValue some xsd:decimal[> 64.0] SubClassOf hasPinCount value
    HighPinCount
hasSelf-HeatingValue some xsd:decimal[> 10.0] SubClassOf hasSelf-Heating value
    HighSelf-Heating
Component and (hasEMI value HighEMI) SubClassOf hasCoverage value
    EMC_and_Signal_Integrity_Analysis_Coverage
Component and (hasPinCount value HighPinCount) SubClassOf hasCoverage value
    Physical_Stress_Analysis_Coverage
Component and (hasSelf-Heating value HighSelf-Heating) SubClassOf hasCoverage
    value E/E_Power_and_Load_Analysis_Coverage
Component and (hasHeight some xsd:decimal[> 2.54]) and (hasWidth some xsd:
    decimal[> 2.54]) SubClassOf hasCoverage value
    Physical_Stress_Analysis_Coverage
Component and (hasPowerDissipation some xsd:decimal[> 0.25]) SubClassOf
    hasCoverage value E/E_Power_and_Load_Analysis_Coverage
Component and (hasPurpose some EssentialVehicleControl) SubClassOf hasCoverage
    value Circuit_and_Systems_Analysis_Coverage
Component and (partOf some MechanicalComponent) SubClassOf hasCoverage value
    Physical_Stress_Analysis_Coverage
Component and (hasCoverage value Circuit_and_Systems_Analysis_Coverage)
    SubClassOf needsAnalysis value Circuit_and_Systems_Analysis
Component and (hasCoverage value EMC_and_Signal_Integrity_Analysis_Coverage)
    SubClassOf needsAnalysis value EMC_and_Signal_Integrity_Analysis
Component and (hasCoverage value Physical_Stress_Analysis_Coverage) SubClassOf
    needsAnalysis value Physical_Stress_Analysis
Component and (hasCoverage value E/E_Power_and_Load_Analysis_Coverage)
    SubClassOf needsAnalysis value E/E_Power_and_Load_Analysis
Component and (needsAnalysis value EMC_and_Signal_Integrity_Analysis) SubClassOf
     needsData value MissionProfileEMIData1
Component and (needsAnalysis value Physical_Stress_Analysis) SubClassOf
    needsData value MissionProfilePhysicalStressData1
Component and (needsAnalysis value E/E_Power_and_Load_Analysis) SubClassOf
    needsData value Temperature
Component and (needsData value Temperature) SubClassOf needsData value
    MissionProfileTemperatureData1
```

**Quelltext 9.1** – OWL-Axiome, die das Wissen über Analysen und Kriterien zur Auswahl der Analysen formalisieren

```
ElectronicEngine(?x) ^ mountedOn(?x, ?y) -> hasEMI(?y, HighEMI)
Vehicle(?x) ^ hasComponent(?x, ?y) ^ hasMountingPoint(?y, EngineMountingPoint)
    -> hasEMI(?y, HighEMI)
mountedOn(?x, ?y) ^ hasEMI(?y, ?z) -> hasEMI(?x, ?z)
```

**Quelltext 9.2** – SWRL-Regeln für die Propagierung von EMI-Eigenschaften

Zum Systemmodell gehört weiterhin eine Instanz der Klasse Vehicle, welche beschreibt, dass das modellierte Fahrzeugsystem neben dem FTCO3V455A1 über einen elektrischen Antriebsstrang verfügt, siehe Quelltext 9.3.

```
Individual: Vehicle1

    Types:
        Vehicle

    Facts:
     hasComponent            ElectronicEngine,
     hasComponent            FTCO3V455A1
```

**Quelltext 9.3** – Fahrzeug-Systemmodell als OWL-Individual in Manchester-Syntax

Um eine ontologische Repräsentation der Komponente zu erstellen, werden existierende Komponentenmodelle in OWL-Repräsentationen transformiert. Neben der Verwendung der Modelle, werden Informationen über den Kontext vom System aufgenommen, wie beispielsweise der Einbauort und dass die Komponente Teil eines mechanischen Systems ist. Quelltext 9.4 zeigt eine Serialisierung des Komponentenmodells.

```
Individual: FTCO3V455A1

    Types:
        Component

    Facts:
     hasPurpose              Steering,
     mountedOn               EngineCompartmentMountingPoint,
     partOf                  MechanicalComponent,
     hasHeight               29.0,
     hasPinCount             19,
     hasPowerDissipation     115.0,
     hasWidth                44.0
```

**Quelltext 9.4** – Komponentenmodell als OWL-Individuum in Manchester-Syntax

Verfügbare MP-Daten werden parallel auf semantisch angereicherte Repräsentationen abgebildet. Metadaten-Extraktion wird auf die MP Daten angewendet, um die Klassifikation für eine korrekte Datenauswahl zu unterstützen. Danach wird automatisches Schlussfolgern eingesetzt, um die richtigen Analysen und zugehörige Daten auszuwählen.

Abbildung 9.2 zeigt die Auswahl von Analysen anhand des vorgestellten Ansatzes und Quelltext 9.5 zeigt die abgeleiteten Axiome für den FTCO3V455A1. Der EMI Wert wird durch eine andere Komponente induziert (ElectronicEngine), welche örtlich nah am FTCO3V455A1 montiert ist, unter Nutzung der in Abschnitt 5.5 beschriebenen *property chain*.

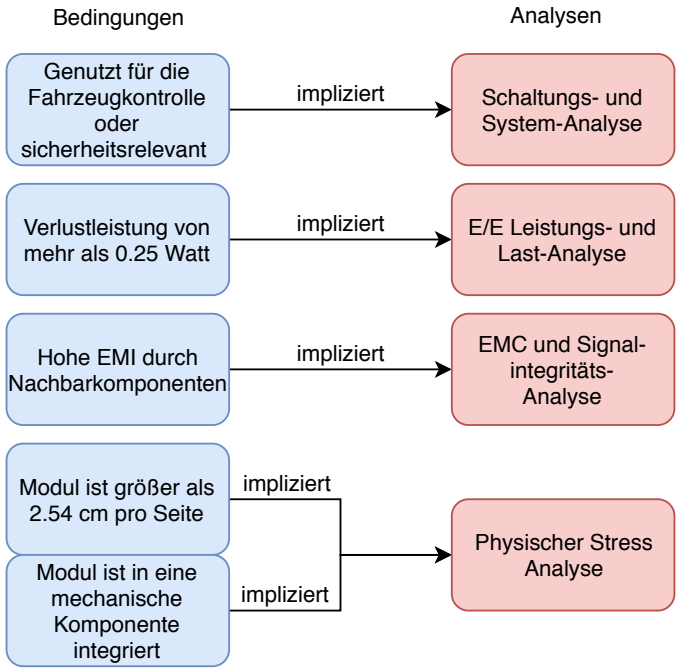



**Abbildung 9.2** – Analysenauswahl für den FTCO3V455A1

```
Individual: FTCO3V455A1

    Types:
        Component

    Facts:
     hasCoverage              CaS_Analysis_Coverage,
     hasCoverage              EMCaSI_Analysis_Coverage,
     hasCoverage              PSA_Coverage,
     hasCoverage              E/EPaL_Analysis_Coverage,
     hasEMI                   HighEMI,
     hasSiblingComponent      ElectronicEngine,
     hasSiblingComponent      FTCO3V455A1,
     needsAnalysis            CaSystems_Analysis,
     needsAnalysis            EMCaSI_Analysis,
     needsAnalysis            Physical_Stress_Analysis,
     needsAnalysis            E/E_Power_and_Load_Analysis,
     needsData                MissionProfileEMIData,
     needsData                MissionProfilePhysicalStressData,
     needsData                MissionProfileTemperatureData,
     needsData                Temperature
```

**Quelltext 9.5** – Abgeleitete Axiome in Manchester-Syntax

Der RV Prozess wird beschleunigt und Fehler, die bei der manuellen Testauswahl entstehen können, werden vermieden. Angenommen, im Laufe der Entwicklung ändert sich die Spezifikation des Inverters. Der vorgestellte Ansatz kann dann eingesetzt werden, um den RV Plan entsprechend zu aktualisieren.

Auf höherer Ebene ist ein zusätzlicher Nutzen, dass unter Einsatz einer *property chain* wie in Axiom 9.1 für eine Komponente, welche aus mehreren Subkomponenten besteht, die Analysen welche, für die Gesamtkomponente durchzuführen sind, aggregiert werden können.

$$\mathsf{partOf}^{-} \circ \mathsf{needsAnalysis} \sqsubseteq \mathsf{needsAnalysis} \tag{9.1}$$

Genauso könnten auch Robustness Indicator Figures (RIFs) (siehe [10]) aggregiert werden, um einen Überblick über die Gesamtrobustheit der Komponente zu erlangen.

#### Ergebnis

Durch die Teil-Automatisierung der Erstellung des RV-Plans könnte die Komplexität, der RV verringert werden. Außerdem könnten Fehler vermieden werden. Dadurch könnten Entwicklungskosten eingespart werden. Es zeigt sich, dass sich OWL-Ontologien eignen könnten, insbesondere teilweise bereits formalisiertes Wissen darzustellen und dieses zu verarbeiten. Neben der Mächtigkeit der zugrundeliegenden Beschreibungslogik $\mathcal{SROIQ}_{(\mathcal{D})}$ von OWL zur Darstellung von Konzepten und Fakten hat sich zudem die Kombination mit der Regel-Sprache SWRL als nützlich erwiesen. Die entwickelten Regeln, welche in dem konkreten Fallbeispiel eine Propagierung von einbauortspezifischen Charakteristika ermöglichen, sind zudem intuitiver als eine Menge von Axiomen. Insgesamt stellt das Ergebnis dieses Fallbeispiels durch diese Erkenntnisse somit ein Beitrag in Hinsicht auf die in Abschnitt 1.1 definierten Ziele dar.

### 9.3.2 Fallbeispiel 2: Stresstest-Auswahl in der AEC Q100

Es wurden verschiedene Standards durch das AEC veröffentlicht, welche unter anderem Anforderungen zur Qualifikation von elektronischen Schaltungen in der *AEC Q100* [60] definieren. Dieser Standard ist in der Automobilbranche weit verbreitet. Der in Abschnitt 5 beschriebene Ansatz kann auch im Qualifikationsprozess eingesetzt werden. Ein Unterschied zum beschriebenen Vorgehen liegt dabei allerdings darin, dass anstelle von Analysen Stresstests ausgewählt werden. Folglich entsprechen bei Anwendung des Ansatzes auf die Stresstest-Auswahl die Stresstests den Analysen, die ebenfalls eine bestimmte Abdeckung erfordern, was die Basis des Verfahrens bildet. Ein weiterer Unterschied liegt darin, dass die Definitionen

der Stresstests nach dem Standard alle Umgebungsbedingungen vorschreiben, sodass das Übergeben passender MPs nicht so nützlich für den Analysten ist, wie in der Analyseauswahl. Davon abgesehen ist der Ablauf ziemlich ähnlich: Ein Komponentenmodell dient als Eingabe für den Ansatz, was die Grundlage für die Stresstest-Auswahl darstellt. Hintergrundwissen aus den Definitionen des Standards wird in Ontologien modelliert. Anschließend werden die Abdeckungen der Stresstests mit den Komponenteneigenschaften verglichen.

Tabelle 9.1 zeigt einen Ausschnitt relevantem Wissens aus dem Überblick über die Testmethoden. Die *NOTES* Spalte listet einige Fakten auf, welche als Kriterien für die Stresstestauswahl, basierend auf den Komponenteneigenschaften, genutzt werden können.

| STRESS | NOTES | ADDITIONAL REQUIREMENTS |
|---|---|---|
| **Temperature Cycling** | H, P, B, D, G | PC before TC for surface mount devices.<br>Grade 0: -55oC to +150oC for 2000 cycles or equivalent.<br>Grade 1: -55°C to +150°C for 1000 cycles or equivalent.<br>Note: -65°C to 150°C for 500 cycles is also an allowed test condition due to legacy use with no known lifetime issues.<br>Grade 2: -55°C to +125°C for 1000 cycles or equivalent.<br>Grade 3: -55°C to +125°C for 500 cycles or equivalent.<br><br>TEST before and after TC at hot temperature. After completion of TC, decap five devices from one lot and perform WBP (test #C2) on corner bonds (2 bonds per corner) and one mid-bond per side on each device. Preferred decap procedure to minimize damage and chance of false data is shown in Appendix 3. |

**Tabelle 9.1** – Auszug der AEC Q100 Übersicht über die Stresstestmethoden [60]

Im Folgenden werden diese Kriterien nach dem englischen Original aufgeschlüsselt:

H Hermetic packaged devices only

P Plastic packaged devices only

B Solder Ball Surface Mount Packaged (BGA) devices only

Weitere Eigenschaften sind:

D Destructive test, devices are not to be reused for qualification or production

G Generic data allowed

Darüber hinaus enthält die *ADDITIONAL REQUIREMENTS* Spalte weitere Fakten, die als Kriterien herangezogen werden können. So beispielsweise, dass der *preconditioning* Test vor dem *temperature cycling* Test durchgeführt werden sollte. Diese Informationen werden in einer Ontologie formal abgebildet und für die Stresstest Auswahl herangezogen.

Unter Nutzung dieses Wissens kann selbst die Reihenfolge von Stresstests berücksichtigt werden. Außerdem können neben der Stresstestauswahl Testbedingungen, Sample-Größen und die Anzahl der Chargen, sowie Akzeptanzkriterien dem Analysten zur Verfügung gestellt werden.

### Testauswahl bei der Prozessänderungsqualifikation

Der vorgestellte Ansatz kann auch nützlich sein, wenn er auf die Testauswahl bei Prozessänderung angewendet wird. Tabelle 3 auf Seite 18 im AEC Q100 Dokument [60] zeigt, welche Stresstests durchgeführt werden müssen, wenn der Prozess geändert wird. Nicht alle Tests sollten bei einer Prozessänderung durchgeführt werden, insbesondere müssen nicht alle Tests für jedes Halbleiter-Bauteil ausgeführt werden. Liegt beispielsweise eine *circuit rerouting* Prozessänderung vor, so sollte der *temperature cycle* Stresstest nur für Devices, welche für das periphere Routing erforderlich sind, durchgeführt werden. In diesem Fall ist außerdem der *power temperature cycling* Test nur für Halbleiter-Bauteile durchzuführen, welche den folgenden Test benötigen: „Test required only on devices with maximum rated power $\geqslant 1$ watt or $\Delta T_j \geqslant 40°\mathrm{C}$ or devices designed to drive inductive loads“. Diese Anforderungen stellen die Fakten für den Abdeckungsvergleich im vorgestellten Ansatz dar. Diese werden analog zu den in Quelltext 9.1 gezeigten Axiomen formalisiert.

### Ergebnis

Aus der Betrachtung dieses Fallbeispiels geht hervor, dass sich das vorgeschlagene Verfahren auch auf weitere Einsatzgebiete übertragen lässt und somit der Anwendungsbereich über die Unterstützung, der RV hinausgeht. Weiterhin auch, dass sich das Verfahren potenziell einsetzen lässt, wenn eine Auswahl von Analysen anhand von spezifischen Kriterien beliebiger zu untersuchender Objekte durchgeführt werden soll. Ebenso wird erkenntlich, dass es mit dem Verfahren nicht nur möglich ist, die Menge durchzuführender Analysen zu bestimmen, sondern außerdem auch die Reihenfolge, in der Analysen durchzuführen sind. Letztlich ist hervorzuheben,

dass dieses Fallbeispiel zeigt, dass es das Verfahren außerdem ermöglicht, relevante Daten für die durchzuführenden Analysen bereitzustellen. Diese Tatsachen zeigen insgesamt, dass die Anwendungsmöglichkeiten von OWL-Ontologien äußerst vielfältig sind und sich nicht nur auf das WWW beschränken. Eine besondere Stärke dieser Sprache beziehungsweise eines Ontologie-basierten Ansatzes liegt somit auch in der Domänen- und Anwendungsneutralität.

### 9.3.3 Fallbeispiel 3: Energiebedarfsänderung bei Anforderungsänderungen

Dieses Fallbeispiel bezieht sich auf den in Kapitel 6 beschriebenen Ansatz zur Ontologie-gestützten CIA. In diesem Fallbeispiel wird als Beispiel für Entwurfsparameter und deren Wirkzusammenhänge das Werkzeug TAMTAMS [163] herangezogen. TAMTAMS ist ein Werkzeug zur Analyse von CMOS-Schaltungen. Das TAMTAMS Modul, welches in diesem Fallbeispiel verwendet wird, ist das Modul auf Systemebene, zur Analyse des Energiebedarfs von Logik-Gattern und Registern. Dieses Modul besteht aus einigen „low-level“ Technologieparametern, sowie „high-level“ Parametern wie der Frequenz oder dem gesamten Energiebedarf, siehe Abbildung 9.3 und 9.4.

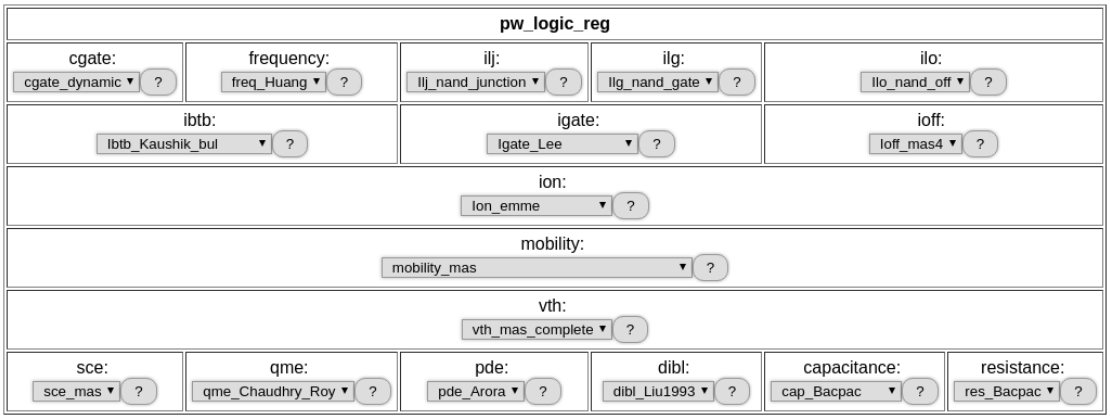



**Abbildung 9.3** – Web-GUI von TAMTAMS für die Analyse des Energiebedarfs von Logikgattern und -registern

Die Abhängigkeiten der Parameter implizieren eine Grafstruktur, welche mittels CTEF explizit spezifiziert wurde, wie in Abbildung 9.4 dargestellt.

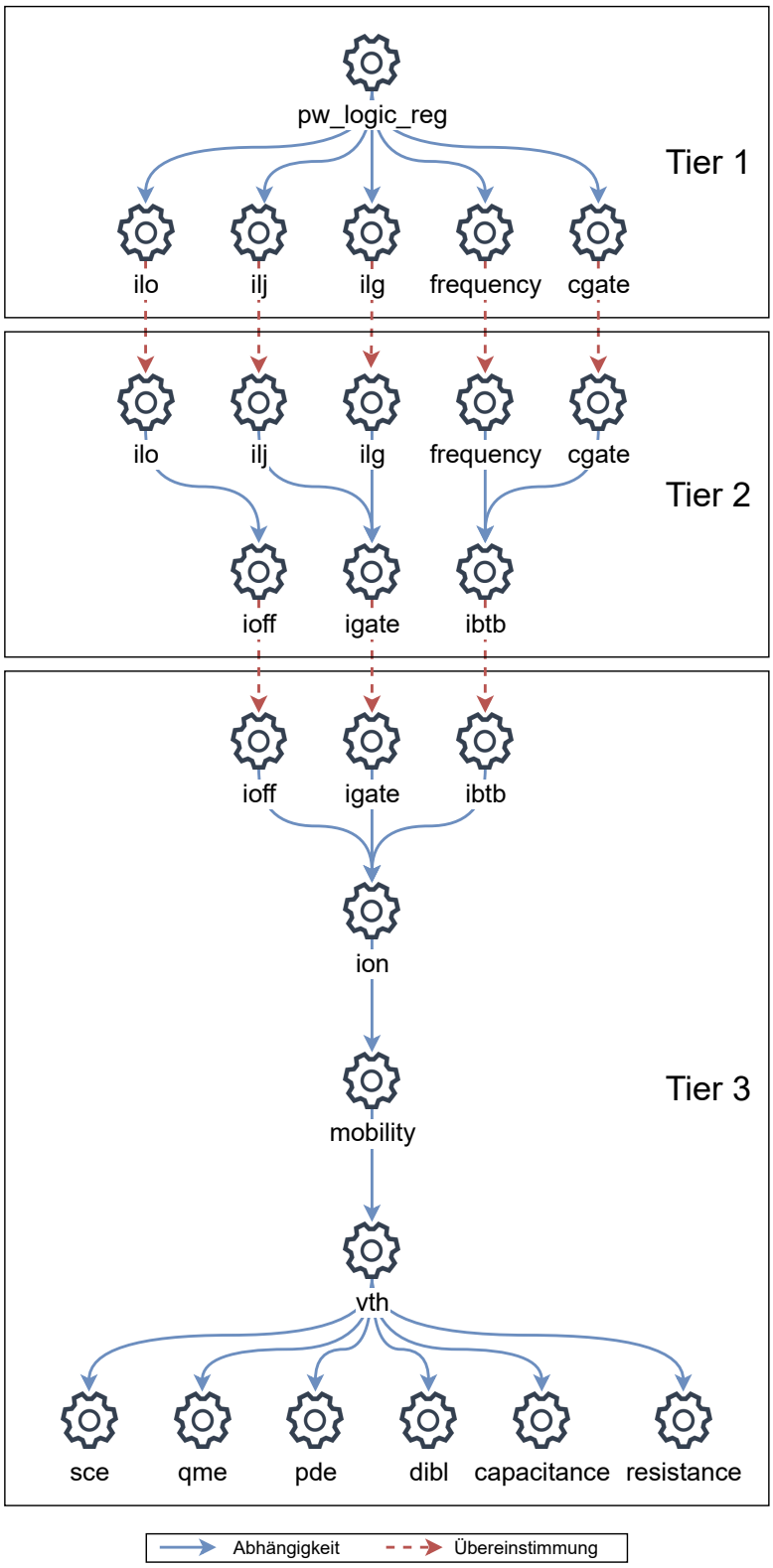



**Abbildung 9.4** – Übersetzung der impliziten Grafstruktur der Energiebedarfsanalyse von Logik-Gattern und Registern aus TAMTAMS in einen AÜG in CTEF

Die Gruppierung der Parameter in Zulieferstufen ist nur ein Beispiel und kann in der Praxis anders aussehen. Dies hat jedoch auch keinen Einfluss auf das Resultat des Ansatzes. Die Relationen zwischen Entwurfsparametern sind wichtig, sowie deren Formation zu einer Hierarchie, bei der Entwurfsparameter auf verschiedenen Ebenen ansässig sind. Die Menge der Entwurfsparameter und die Beziehungen zwischen ihnen sind das Expertenwissen, welches mittels CTEF gesammelt wurde. In dem hier vorgestellten Beispiel wurde das Wissen über die Wirkzusammenhänge zwischen Entwurfsparametern TAMTAMS entnommen.

Neben dem Expertenwissen über Wirkzusammenhänge zwischen Entwurfsparametern, welche mittels CTEF gesammelt wurden, wird Wissen über Anforderungen in entsprechende Ontologien überführt. Sobald die Ontologien bereitstehen, beginnt das System Wissen über die Beziehungen zwischen Anforderungen und Entwurfsparametern, über Benutzereingaben, aufzunehmen. Die Entwickler etablieren das Systemmodell, welches Wissen vernetzt und den zentralen Wissenskorpus bildet. Verknüpfungen zwischen Komponenten und MPs werden eingefügt.

Betrachten wir nun den Fall, dass die Anforderung, welche die Systemfrequenz spezifiziert, während der Entwicklung geändert wird. Dies wird als Änderung an einem ReqIF Dokument, welches Anforderungen an das zu entwickelnde System spezifiziert, erkannt. Das System löst die Abhängigkeiten zwischen den Anforderungen auf, um festzustellen, ob und wenn ja, welche weiteren Anforderungen von der Änderung betroffen sind. In Konsequenz kann das System detektieren, welche Entwurfsparameter von der Änderung betroffen sind. Dies geschieht unter Nutzung von Wissen über Wirkzusammenhänge zwischen Entwurfsparametern, welches als Ontologie aus CTEF exportiert und mit dem Systemmodell verbunden wurde. Abschließend werden dem Analysten von der Änderung betroffene Entwurfsparameter präsentiert: *frequency, ibtb, ion, mobility, vth, sce, qme, capacitance, pde, resistance* und *dibl*, siehe Abbildung 9.5.

### Ergebnis

Es ergibt sich durch dieses Fallbeispiel, dass es möglich ist, den Aufwand einer CIA durch die Automatisierung der Analyse betroffener Entwurfsparameter zu verringern. Dadurch könnten Kosten eingespart werden. Mit der Automatisierung geht außerdem einher, dass Fehler, die bei manueller Analyse auftreten können, vermieden werden. Überdies kann die Überführung gesammelten Expertenwissens über Wirkzusammenhänge in explizites Wissen mittels Darstellungen in OWL-Ontologien einen Beitrag zur Verbesserung des Verständnisses über den Entwurf leisten. Die explizit dargestellten Wirkzusammenhänge zwischen Entwurfsparametern können dadurch einfacher weiter analysiert werden. Ergänzend dazu sind auch Korrelationen zu Anforderungen Bestandteil resultierender OWL-Ontologien und können ebenfalls zum besseren Verständnis beitragen und in Analysen einfließen.

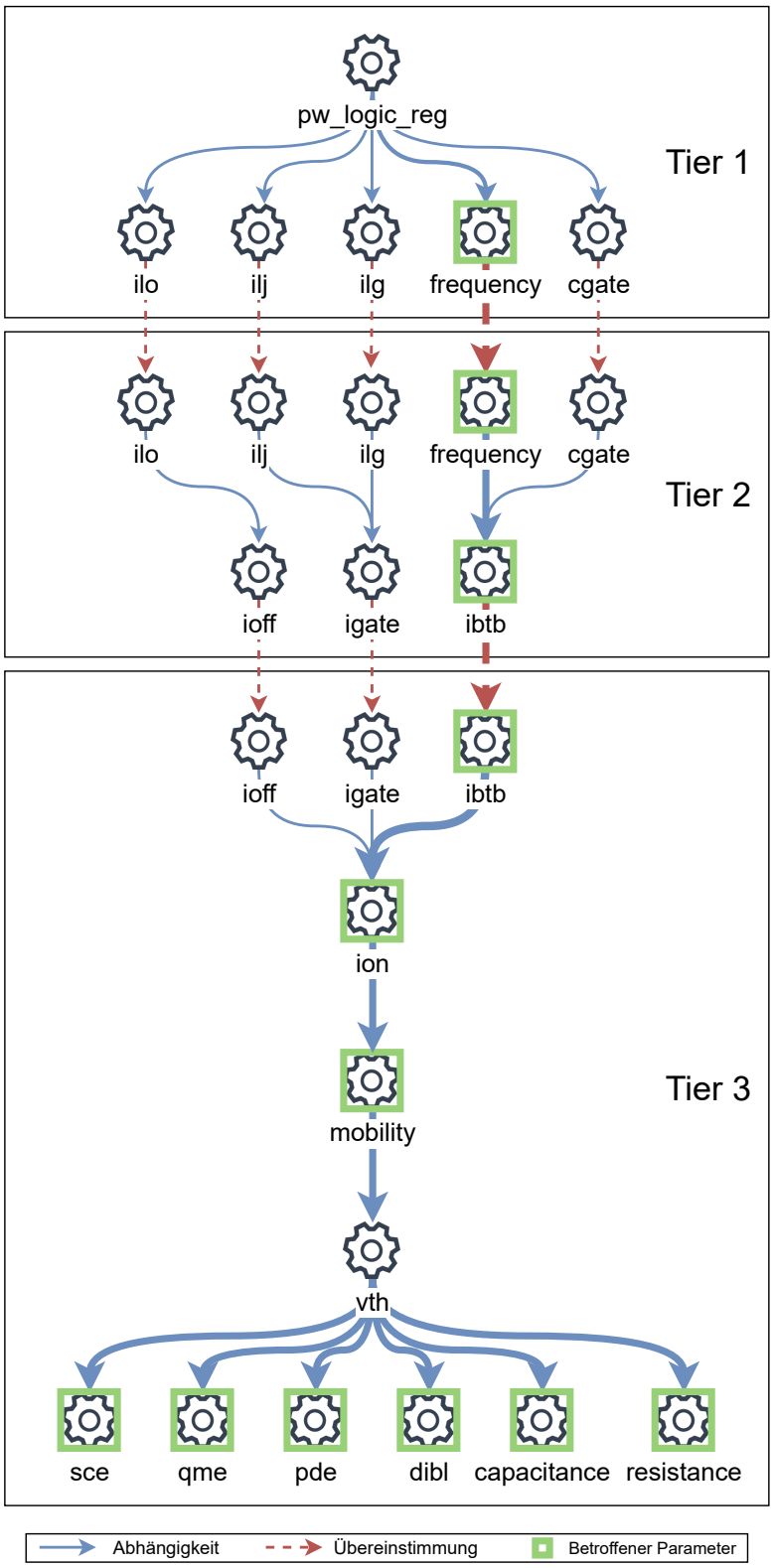



**Abbildung 9.5** – Von der Änderung betroffene Parameter

### 9.3.4 Fallbeispiel 4: Anwendung von Technologie-Fehlermodellen

Dieses Fallbeispiel bezieht sich auf die in Kapitel 7 beschriebene Applikation zu Ontologie-basierter Anforderungs-Transformation. In diesem Fallbeispiel wird betrachtet, dass für die Herstellung einer Komponente mit vier Stufen in der Lieferkette ermittelt werden muss, welche Parameter bei der Weitergabe und der damit verbundenen Transformation des MP beizubehalten sind. Das heißt, es erfolgt eine Weitergabe des MP vom OEM an einen Zulieferer, der selbst einen Zulieferer hat, der wiederum vom Technologielieferanten beliefert wird, siehe Abbildung 9.6.

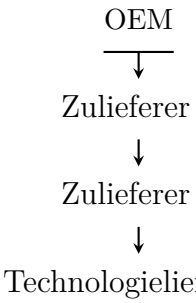



**Abbildung 9.6** – MP Weitergabe entlang der Lieferkette in dieser Fallbeispiel

Um ein Fehlermodell auf der Ebene des Technologielieferanten auswerten zu können, müssen bestimmte MP-Parameter bei einer Transformation auf den vorhergehenden Ebenen beibehalten werden. In dem Fallbeispiel wurde ein MP-Dokument auf der Basis des MPFO Version 0.2.1 zur Beschreibung der Parameter eingesetzt. Dieses MP-Dokument enthält sieben Parameter $t_0$, $t_1$, $t_2$, $v_0$, $v_1$, $v_2$, $failureRate$ sowie Temperaturbereiche für Stresstests einer Komponente. Die Temperaturbereiche werden als acht Intervalle ausgedrückt, die sich insgesamt über einen Bereich von –10 °C bis 90 °C erstrecken, siehe Tabelle 9.2.

| Untere Grenze | obere Grenze |
|---|---|
| -10.0 | 20.0 |
| 20.0 | 30.0 |
| 30.0 | 40.0 |
| 40.0 | 50.0 |
| 50.0 | 60.0 |
| 60.0 | 70.0 |
| 70.0 | 80.0 |
| 80.0 | 90.0 |

**Tabelle 9.2** – Intervalle im verwendeten MP

Nutzung der Fehlermodell- und Transformations-Ontologie

Vor einer eigentlichen Transformation des Basis-MP wird die *Fehlermodell- und Transformations-Ontologie* (siehe Abschnitt 7.3.3.1) genutzt, um herauszufinden, welche Parameter beibehalten werden müssen. Dies erfordert die manuelle Benutzerangabe der Ebene der Lieferkette, auf welcher die Transformation durchgeführt wird. Weiterhin wird die Existenz einer Wissensbasis, die Fehlermodelle enthält, vorausgesetzt. Es wurde eine Ontologie eingesetzt, die fünf Front-end-of-line (FEOL) und Back-end-of-line (BEOL) Fehlermodelle beinhaltet: *FEOL-TDDB*, *FEOL-HCI*, *FEOL-BTI*, *BEOL-TDDB* und *BEOL-EM*. Darüber hinaus wurden entsprechende Technologie-, Produkt- und MP-Parameter, die für die Auswertung der Fehlermodelle erforderlich sind, in der Ontologie definiert und mit den Fehlermodellen assoziiert. Quelltext 9.6 zeigt eine Repräsentation des TDDB-Fehlermodells in der Fehlermodell- und Transformations-Ontologie in Form eines OWL-Individuums.

```
Individual: failureModel-FEOL-TDDB

    Types:
        TechnologicalFailureModel

    Facts:
     hasMissionProfileParameter  failureRate,
     hasMissionProfileParameter  t0,
     hasMissionProfileParameter  v0,
     hasProductParameter  gateArea,
     hasTechnologyParameter  accExponent,
     hasTechnologyParameter  activationEnergy,
     hasTechnologyParameter  charLifeTime,
     hasTechnologyParameter  refGateArea,
     hasTechnologyParameter  refTemperature,
     hasTechnologyParameter  refVoltage,
     hasTechnologyParameter  weibullSlope,
     ofFailure  TDDB,
     ofLayer  FEOL,
     hasEquation  "charLifeTime * AF_V * AF_T * (gateArea/
        refGateArea)**(-1/weibullSlope) * (-ln(1-(failureRate/1e6
        )))**(1/weibullSlope)"^^xsd:string
```

**Quelltext 9.6** – OWL Individuum, welches das technologische Fehlermodell TDDB repräsentiert in Manchester-Syntax

Die Fakten umfassen 13 Objekt-Eigenschaften, welche die TDDB-Fehlermodell Repräsentation mit MP-, Produkt- und Technologie-Parametern, dem zugrundeliegenden Fehlermechanismus und der Ebene assoziieren sowie eine Daten-Eigenschaft, welche das Fehlermodell in Form einer Gleichung mathematisch beschreibt. Die Assoziationen zu Parametern beziehen sich auf diese mathematische Gleichung. Auf diese Weise wird das Wissen über die Menge der für die Auswertung des

Fehlermodells erforderlichen Parameter formalisiert. Die Menge der MP-Parameter dieses Fehlermodells umfasst Spannungs- und Temperatur-Parameter sowie die Fehlerrate unter Einsatz einer Inferenzmaschine kann auf Basis der Fehlermodell- und Transformations-Ontologie ermittelt werden, dass die Spannungs-, Temperatur-Parameter und die Fehlerrate beibehalten werden müssen. Quelltext 9.7 zeigt eine Erklärung des Ontologie-Editors und Anwendungsframeworks Protégé[2] [132] für die Inferenz des Fakts, dass ein Beispiel-Temperatur-Parameter beibehalten werden sollte.

```
1 failureModel-FEOL-TDDB Type TechnologicalFailureModel
2 DirectManipulationMPOTransformation SubClassOf MPOTransformation
3 failureModel-FEOL-TDDB hasMissionProfileParameter failureRate
4 example-transform-oem ofSupplyChainLevel oemLevel
5 example-transform-oem Type OWLAPITransformation
6 MPOTransformation(?trans), ofSupplyChainLevel(?trans, oemLevel),
     TechnologicalFailureModel(?fm), hasMissionProfileParameter(?fm
     , ?param) -> maintainsParameter(?trans, ?param)
7 OWLAPITransformation SubClassOf
     DirectManipulationMPOTransformation
```

**Quelltext 9.7** – Protégé Erklärung für eine OEM-Transformation welche einen Beispiel-Temperatur-Parameter beibehält

Durchführung der MP-Transformation

Ist die Identifikation von beizubehaltenden Parametern erfolgt, so können die Transformationsoperationen spezifiziert werden. Dies geschieht im Beispiel durch die manuelle Implementierung einer Unterklasse des *OWLAPITransformation* Traits (siehe Abschnitt 7.3.3.2, sowie Abbildung 7.5 auf Seite 89). Quelltext A.1 in Anhang A zeigt einen Ausschnitt der Implementierung, aus Platzgründen jedoch ohne die Transformationsoperationen.

Um die Temperaturbereiche zu erweitern, wurde eine Transformation implementiert, welche jeweils die unteren Grenzen der Intervalle verringert und die oberen Grenzen erhöht. Zur Ausführung der Transformation wird das Eingabe-MP zunächst auf eine ontologische Repräsentation, unter Einsatz der in Kapitel 4 beschriebenen Plattform, abgebildet.

Tabelle 9.3 führt einige Metriken der Ein- und Ausgabe-MP-Ontologien auf. Wenngleich einige Axiome durch die Transformation entfernt wurden, bleiben jedoch alle *object property* und *data property* Aussagen (siehe Abschnitt 2.1.3.3) erhalten. Die Differenzen in den Anzahlen der Axiome, Klassen, data- und object properties, sowie den Individuen sind dadurch zu erklären, dass für das hier behandelte Beispiel irrelevante MP Informationen wie Betriebszustände und Port-Spezifikationen beabsichtigt durch die Transformation entfernt wurden.

[2] Siehe `https://protege.stanford.edu/`

| Merkmal | Eingabe-MP-Ontologie | Ausgabe-MP-Ontologie |
|---|---|---|
| Axiome | 3874 | 3633 |
| Logik-Axiome | 2889 | 2773 |
| Deklarations-Axiome | 911 | 860 |
| Klassenanzahl | 52 | 36 |
| Objekt-Eigenschaften | 39 | 31 |
| Daten-Eigenschaften | 30 | 19 |
| Individuen | 775 | 774 |
| Objekt-Eigenschaftsaussagen | 779 | 779 |
| Daten-Eigenschaftsaussagen | 1216 | 1216 |

**Tabelle 9.3** – Metriken von Ein- und Ausgabe-MP-Ontologien

#### Ergebnis

Dieses Fallbeispiel zeigte, dass das Verfahren den Schutz geistigen Eigentums verbessern könnte, indem für weitere Transformationen nicht erforderliche Informationen aus MPs entfernt werden. Gleichzeitig stellt das Verfahren einen Weg dar, mit dem für Transformationen auf nachfolgenden Ebenen der Lieferkette benötigte MP-Parameter beibehalten werden können. Dadurch nimmt insgesamt der notwendige Kommunikationsaufwand ab, da die Gesamtanzahl der Transformationen in der Lieferkette verringert wird. Durch die Definition der Transformationen als Spezialisierungen eines entsprechenden Traits kann die Transformation zudem wiederverwendet und geteilt werden. Insgesamt ergibt sich so eine Vermeidung von Fehlern im Transformationsprozess in Verbindung mit verringerten Kosten durch die reduzierte Komplexität.

### 9.3.5 Fallbeispiel 5: Subkomponenten MP Ableitung

Dieses Fallbeispiel beschreibt die Ableitung eines MP für eine Subkomponente von einem Basis-MP. Als Beispiel betrachten wir die Entwicklung einer Electronic Control Unit (ECU). Ein Tier-1 baut das ECU-Modul, während ein Tier-2 eine Integrated Circuit (IC)-Subkomponente für die ECU herstellt, welche wiederum auf Technologie eines Tier-3 basiert. MPs werden bei der Entwicklung der ECU, der IC Komponente und der darunterliegenden Technologie verwendet. In diesem Fallbeispiel ist die ECU nahe dem Motor verbaut. Abbildung 9.7 zeigt verschiedene ECU-Temperaturmesspunkte, wobei die für dieses Fallbeispiel relevante Innenlufttemperatur als $\mathsf{T}_{\text{EEM Internal}}$ bezeichnet wird.

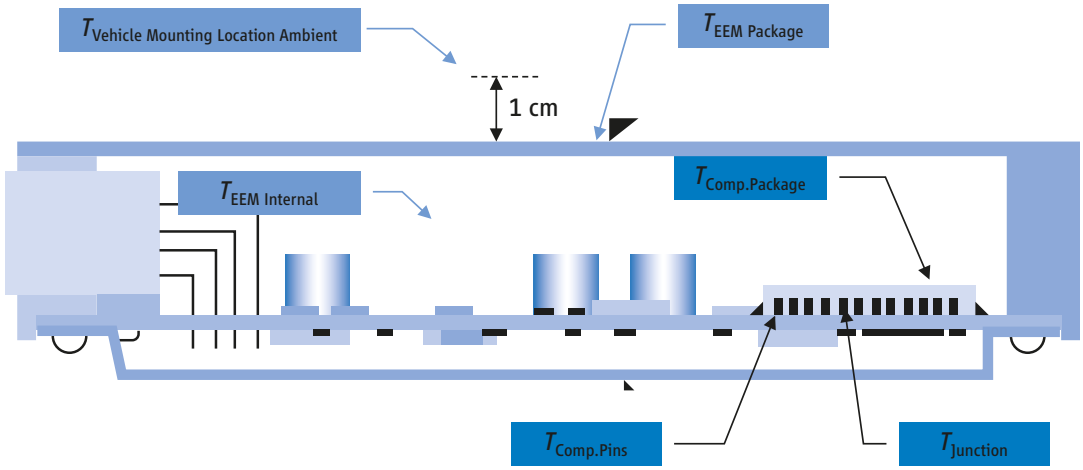



**Abbildung 9.7** – Definitionen verschiedener Temperaturmesspunkte einer ECU (hier Electrical/Electronic Module, EEM genannt), aus dem Handbook for Robustness Validation [10]

| ECU Innenlufttemperatur | **Typische Last** | |
|---|---|---|
| | Fahrzeugkarosserie, Trennwand, Erweiterung nahe dem Motor | |
| -40 °C...10 °C | 6.0 % | 480 h |
| 10 °C...45 °C | 20.0 % | 1600 h |
| 45 °C...60 °C | 33.0 % | 2640 h |
| 60 °C...70 °C | 18.0 % | 1440 h |
| 70 °C...80 °C | 9.0 % | 720 h |
| ...85 °C | 3.0 % | 240 h |
| ...90 °C | 2.0 % | 160 h |
| ...95 °C | 1.7 % | 136 h |
| ...100 °C | 1.5 % | 120 h |
| ...105 °C | 1.4 % | 112 h |
| ...110 °C | 1.3 % | 104 h |
| ...115 °C | 1.2 % | 96 h |
| ...120 °C | 1.0 % | 80 h |
| ...125 °C | 0.9 % | 72 h |
| **Summe** | **100 %** | **8000 h** |

**Tabelle 9.4** – Typisches Temperaturlastprofil der Innenlufttemperatur einer ECU eines PKW im aktiven Betriebszustand, aus [164]

Unter Verwendung einer öffentlich zugänglichen typischen Temperaturlastverteilung [164] kann ein MP für die ECU definiert werden. Dieses Temperaturprofil der

Innenlufttemperatur der ECU wird als Histogramm von 14 Temperaturbereichen zwischen –40 °C und 125 °C dargestellt, siehe Tabelle 9.4.

Thermalwiderstand

Um dem Tier-3 ein MP zur Verfügung stellen zu können, muss der Tier-1 das Temperaturprofil-MP transformieren. Der JEDEC Standard JESD51-2 definiert eine Methodik zur Durchführung von Messungen des Thermalwiderstands von ICs bei natürlicher Konvektion. Der Thermalwiderstand zwischen Verbindungsstelle und Umgebungsluft wird durch Gleichung 9.2 bestimmt. In deren zweiten Form in Gleichung 9.3, mit der Umgebungstemperatur $T_\mathrm{A}$, der Gesamtverlustleistung $P$ und dem Gesamtthermalwiderstand zwischen Verbindungsstelle und der Umgebungsluft $\vartheta_\mathrm{JA}$, kann man die Verbindungsstellentemperatur $T_\mathrm{J}$ berechnen:

$$\vartheta_\mathrm{JA} = (T_\mathrm{J} - T_\mathrm{A})/P \quad \Leftrightarrow \quad (9.2)$$

$$T_\mathrm{J} = T_\mathrm{A} + (P \times \vartheta_\mathrm{JA}) \quad \Leftrightarrow \quad (9.3)$$

$$= \vartheta_\mathrm{JC} + \vartheta_\mathrm{CA} \quad (9.4)$$

wobei $\vartheta_\mathrm{JC}$ der Thermalwiderstand zwischen der Verbindungsstelle und dem Gerätegehäuse und $\vartheta_\mathrm{CA}$ der Thermalwiderstand zwischen Gerätegehäuse und Umgebungsluft sind. Gleichungen 9.3 and 9.4 werden auch in der Industrie für die Verbindungsstellentemperaturberechnung eingesetzt[3]. Diese Gleichungen werden verwendet, um ein entsprechendes Temperaturprofil für den Tier-3, basierend auf dem ECU-MP, zu berechnen.

MP-Ableitung

Unter Nutzung des in Kapitel 7 beschriebenen Systems, kann nun ein MP für die Technologie der IC-Subkomponente vom Basis ECU-MP abgeleitet werden. Dies geschieht unter Beibehaltung der für die Auswertung von Fehlermodellen auf Technologieebene notwendigen Parameter. Die konkrete Spezifikation der Transformationsoperationen erfordert Expertenwissen über die spezifischen Beziehungen zwischen den Lasten der Basiskomponente und der Subkomponente. Diese Beziehungen spiegeln sich in der Transformation wider. Die Berechnung der Verbindungsstellentemperatur, wie zuvor beschrieben, ist ein Beispiel für solche Beziehungen. Parameter, welche nicht als „benötigt“ identifiziert werden, können aus dem Basis-MP entfernt werden, um IP zu schützen und den benötigten Speicherplatz zu reduzieren. Da für jede Subkomponente der ECU, MPs benötigt werden, kann die Spezifikation der Transformationsoperationen außerdem wiederverwendet werden, was Zeit spart und Fehler vermeidet.

Ablauf

Angenommen, die *Fehlermodell- und Transformations-Ontologie* (siehe Abschnitt 7.3.3.1) enthält (1) Informationen über die Ebene in der Lieferkette, auf der die Transformation durchgeführt wird und (2) entsprechende Fehlermodelle. In diesem Fall wäre ein beispielhafter Ablauf, ein MP für die IC-Subkomponente, unter

[3]Zum Beispiel in der Anleitung MT-093 von Analog Devices, in dem Anwendungsbericht SPRA953C von Texas Instruments, dem allgemeinen Informationsdokument TB379 von Renesas oder der Einführung in Temperaturspezifikation AN4017 von Cypress.

Berücksichtigung der Parameter für die Fehlermodell-Auswertung, abzuleiten wie folgt:

1. Der Benutzer spezifiziert die Ebene in der Lieferkette, auf welcher die Transformation durchgeführt wird, in der *Fehlermodell- und Transformations-Ontologie*, beispielsweise über einen Ontologie-Editor, siehe Quelltext 9.8:

```
Individual: example-transform-oem

    Types:
        OWLAPITransformation

    Facts:
     ofSupplyChainLevel  oemLevel
```

**Quelltext 9.8** – OWL Individuum, welches eine Transfromation auf OEM-Ebene repräsentiert in Manchester-Syntax

2. Das System identifiziert Parameter, die beibehalten werden müssen, unter Verwendung der *Fehlermodell- und Transformations-Ontologie* und einer OWL-Inferenzmaschine. Parameter, die beibehalten werden müssen, werden als Objekte dargestellt, die zum Transformations-Individuum durch die OWL-Property maintainsParameter in Beziehung stehen, siehe Quelltext 9.9.

```
Individual: example-transform-oem

    Types:
        OWLAPITransformation

    Facts:
        ofSupplyChainLevel      oemLevel
        maintainsParameter      MPParam_failureRate
        maintainsParameter      MPParam_v0
        maintainsParameter      MPParam_v1
        maintainsParameter      MPParam_v2
        maintainsParameter      MPParam_t0
        maintainsParameter      MPParam_t1
        maintainsParameter      MPParam_t2
```

**Quelltext 9.9** – OWL-Individuum, welches eine Transformation auf OEM-Ebene repräsentiert in Manchester-Syntax. Sieben MP Parameter werden beibehalten, was das abgeleitete Wissen darstellt

3. Der Benutzer spezifiziert die Transformationsoperationen durch die Implementierung eines *MPOTransformation* Traits oder eines der davon spezialisierten

Traits. Dies beinhaltet die Definition der Daten-Quellen und -Ziele der Transformation, siehe Abbildung A.1. Gleichungen 9.3 und 9.4 werden eingesetzt, um die Transformationsoperationen zu implementieren.

4. Das System bildet das Basis-MP der ECU im MPFO-Format auf eine OWL Ontologie ab. Diese Aufgabe wird unter Einsatz der in Kapitel 4 beschriebenen Plattform durchgeführt.

5. Das System führt schließlich die Transformation aus, was schließlich das IC-MP zum Ergebnis hat.

**Ergebnis**

Im vorgestellten Fall ist eine einzige Definition der Transformation ausreichend, um beliebig viele Submodul-MPs vom Basis-MP abzuleiten. Dies zeigt, dass die vom vorgeschlagenen Transformationssystem bereitgestellten Mechanismen zur Wiederverwendung effektiv sind. Dieses Fallbeispiel bestätigt somit die Ergebnisse der Fallbeispiel 4 und ergänzt diese um die Erkenntnis, dass sich die Mechanismen zur Wiederverwendung eigenen, um MPs für Subkomponenten abzuleiten. Ansonsten zeigt das Fallbeispiel, dass auch komplexe Sachverhalte in Bezug auf die zu transformierenden MP-Parameter sich in Transformationsdefinitionen darstellen und auch wiederverwenden lassen.

## 9.4 Diskussion der Ergebnisse

Reasoning-gestützte RV

In Bezug auf die in Kapitel 5 vorgestellte Arbeit und die zugehörigen Fallbeispiele in den Abschnitten 9.3.1 und 9.3.2, gibt es nennenswerte Limitierungen. Aufgrund der OWL zugrunde liegenden Open World Assumption (OWA) ist es zwar möglich auszudrücken, dass eine Komponente ein bestimmtes Merkmal *nicht* hat, beispielsweise über *Closure Axiome*, jedoch nicht ohne weitere Probleme einzuführen [133]. Ein weiterer Punkt ist, dass die Menge der zur Auswahl stehenden Analysen groß genug sein muss, damit das System wirklich nützlich wird. Wenn es nur eine kleine Menge von Analysen gibt, wird auch der Nutzen gering sein, da ein Entwickler dann auch leicht manuell den RV Plan erstellen könnte. Die Skalierbarkeit der Methode hängt primär von der Leistung der eingesetzten Inferenzmaschine ab. Da der Ansatz aber nicht von einer speziellen Implementierung abhängt, kann prinzipiell jede OWL Inferenzmaschine eingesetzt werden. Es könnte auch sein, dass durch Optimierungen, die Effizienz weiter gesteigert werden kann.

Ontologie-gestützte CIA

Was die in Kapitel 6 vorgestellte Arbeit und das Fallbeispiel im Abschnitt 9.3.3 betrifft, so gäbe es verschiedene Möglichkeiten, den Ansatz weiter zu verbessern. Eine Möglichkeit wäre die Erweiterung um existierende Ansätze aus der Literatur,

wie durch das s.g. *Repository Mining*, um den vergangenen Verlauf von Änderungen zu berücksichtigen. Der Ansatz erwartet zudem, dass sämtliche Eingaben in der abstraktesten Form vorliegen – in natürlicher Sprache. Würden stattdessen auch Modelle zum Einsatz kommen, so würde dies die Verknüpfung von Artefakten stark verbessern, da die Modelle nach spezifischen Eigenschaften abgefragt werden könnten, die für die Entscheidung herangezogen werden könnten, ob zwei Artefakte zueinander in Beziehung stehen. Dies steht im Zusammenhang mit einer weiteren Verbesserungsmöglichkeit: Dem Einsatz von *Ontology Mapping* oder *Entity Matching* Ansätzen, um Anforderungen mit Entwurfsparametern zu verknüpfen, was essenziell für den Ansatz ist.

Ontologiebasierte Transformation von Anforderungen

Bei dem in Kapitel 7 vorgestellten Ansatz und den beiden zugehörigen Fallbeispielen in den Abschnitten 9.3.4 und 9.3.5, wurden im Vorfeld andere technische Lösungsmöglichkeiten betrachtet. Dazu zählen beispielsweise das EMF [160] zum Einsatz von Modellen anstatt Ontologien und Transformationssprachen wie ETL [154] oder ATL [139]. Letztlich fiel die Entscheidung dann aber für ein möglichst einfaches Transformationssystem aus, um die Komplexität zu verringern, was die Ergebnisse aus der Literatur bestätigte [96].

Eine Limitierung des Ansatzes ist, dass die SWRL-basierte Transformation nur monotone Inferenz unterstützt. Daher können SWRL-basierte Transformationen nur neue Fakten zu einer Ontologie hinzufügen, nicht jedoch bestehende Fakten modifizieren oder entfernen. Dem kann allerdings beispielsweise durch zusätzliche Vor- und Nachverarbeitungsschritte entgegnet werden. In diesen Extra-Schritten würde dann die Modifikation oder Entfernung von Fakten vonstattengehen. Ein weiterer limitierender Faktor, der von der in Kapitel 4 vorgestellten Plattform her rührt, ist, dass die Abbildung von MPs auf ontologische Repräsentationen *in-Memory*-basiert ist. Das bedeutet, dass die Konstruktion des Ontologie-Modells im Hauptspeicher stattfindet, was für riesige MPs problematisch sein könnte. Derart große MPs kommen in der Praxis jedoch eher selten vor.

Es gibt auch Möglichkeiten, diesen Ansatz zu verbessern. Allgemeine Verbesserungen wären Ansätze zur Wiederverwendung von M2M Transformationen und deren Verifikation [90; 165; 166]. Eine Verbesserung in Bezug auf die Menge der unterstützten Transformationsansätze wäre die Integration von Muster-basierter Transformation [167]. In Hinsicht auf die Validierung der ontologischen Abbildungen von MPs könnte eine Modell-Validierung wie in [168] zum Einsatz kommen.

### 9.4.1 Vorteile des Einsatzes von Ontologien

**Integrationsmöglichkeit in modellbasierte Entwicklungsprozesse**

Eine Schlüsselkompetenz von SWT, insbesondere Ontologien, ist die Ermöglichung *semantischer Interoperabilität*. Dies begünstigt den Einsatz in modellbasierten Entwicklungsprozessen. Ontologien können als gemeinsames Vokabular

oder als Referenzmodelle dienen und somit Relationen zwischen heterogenen Modellen spezifizieren, zu deren weiterer Verarbeitung. Wie Oberle feststellte, vereinen nur Ontologien konzeptuelle Modellierung mit Web Compliance, Formalisierung und Reasoning [169]. Diese Tatsache trägt auch dazu bei, dass eine Integration von Ontologie-basierten Anwendungen in modellbasierte Entwicklungsprozesse begünstigt wird.

**Verbessertes Information Retrieval**

Das *Information Retrieval* kann durch SWT, insbesondere Ontologien in Verbindung mit Abfragesprachen wie SPARQL verbessert werden. Dies kann durch die Abfrage impliziten Wissens mittels semantischer Anfragen erreicht werden, was einen klaren Vorteil gegenüber Datenbank-basierten Ansätzen darstellt.

**Integration von Anforderungsmanagement und Wissensmanagement**

Wie in dieser Arbeit gezeigt wurde, lassen sich Aufgaben des Anforderungsmanagements mit dem Wissensmanagement verknüpfen. Dies kann insgesamt zu einer Vermeidung von Fehlern und einer Verbesserung der Gesamtproduktqualität führen. Außerdem kann es helfen, das Verständnis des Entwurfs und dessen Beziehungen zu Anforderungen und Entwurfsparametern zu verbessern.

### 9.4.2 Nachteile des Einsatzes von Ontologien

**Open World Assumption**

Die OWA kann ein Hindernis für Entwickler bei der Adoption Ontologie-basierter Ansätze darstellen. OWL liegt die OWA zugrunde, was einen entscheidenden Unterschied zu klassischen Datenbanken darstellt. Wie sich bei der Untersuchung der Fallbeispiele 9.3.1 und 9.3.2 zeigte, kann dies ein Hindernis darstellen. Da es aber Möglichkeiten gibt, den aus der OWA resultierenden Problemen zu entgegnen, ist dieser Punkt keinesfalls als „Showstopper“ anzusehen.

**Monotone Inferenz von OWL und SWRL**

Ebenfalls zeigte sich, dass die *Monotonie* von OWL ein Hindernis bei der Realisierung von Ansätzen darstellen kann. Die Inferenzmechanismen von OWL können demnach keinesfalls vorhandenes Wissen modifizieren oder entfernen. Wie auch beim vorherigen Punkt lässt sich dieser Tatsache aber bspw. durch zusätzliche Vor- und/oder Nach-Verarbeitungsschritte entgegnen. Insgesamt ist aber damit zu rechnen, dass dies mit einem Mehraufwand bei der Entwicklung verbunden ist.

**Geringe Skalierbarkeit verbreiteter Werkzeuge**

Zum Zeitpunkt dieser Arbeit verfügbare Werkzeuge zur Erstellung und Ver-

arbeitung von Ontologien skalieren meist nicht gut. Das rührt in der Regel daher, dass Ontologie-Modelle *in-Memory* konstruiert werden und somit deren Umfang auf die Größe des Hauptspeichers des Entwicklungssystems begrenzt ist. Es existieren jedoch auch für dieses Problem Ansätze, wie die Konstruktion der Ontologie-Modelle auf Datenbanken zu verlagern [170; 171]. Daher ist wohl absehbar, dass dieses Problem in der Zukunft gelöst werden kann.

### 9.4.3 Schlussfolgerung

Zusammenfassend lässt sich feststellen, dass die zentrale Forschungsfrage damit beantwortet werden kann, dass es vielfältige Möglichkeiten zum Einsatz von Ontologien zur Verbesserung des Entwurfsprozesses von automobilelektronischen Systemen gibt. Die im vorherigen Abschnitt angeführten Nachteile des Einsatzes von Ontologien für die Entwurfsoptimierung automobilelektronischer Systeme sind jedoch ebenfalls stets zu berücksichtigen. Da sich die Identifikation von Vor- und Nachteilen des Einsatzes von Ontologien in der Entwurfsoptimierung automobilelektronischer Systeme auf die Erfahrungen aus der Entwicklung der Plattform und den drei Anwendungen beschränkt, ist außerdem mit weiteren Vor- und Nachteilen zu rechnen. Es zeigte sich aber, dass eine Entwicklungsplattform, welche als einheitliche Basis in der Entwicklung dient, ein praktikabler Weg ist MPAD Anwendungen zur Unterstützung verschiedener Entwicklungsaufgaben zu realisieren.

# 10 Zusammenfassung und Ausblick

Diese Arbeit zielt auf die Beantwortung der Forschungsfrage ab, ob sich Schlussfolgerungen, unter Nutzung von Ontologien, auf der Basis von Entwurfsmodellen zur Verbesserung des Entwurfsprozesses automobilelektronischer Systeme, ziehen lassen. Da in der Entwicklung von automobilelektronischen Systemen MPs eine wichtige Rolle spielen, werden auch diese bei den Betrachtungen in dieser Arbeit berücksichtigt. Das primäre Ziel dieser Arbeit ist die Entwicklung exemplarischer Methoden und diese begleitenden Werkzeuge, welche auf der Basis von Entwurfsmodellen Schlussfolgerungen zur Verbesserung des Entwurfsprozesses ziehen können. Sekundäres Ziel ist die Identifikation von Vor- und Nachteilen, die sich durch den Einsatz von Ontologien zum automatischen Schlussfolgern auf der Basis von Entwurfsmodellen bei dem Automobilelektronikentwurf ergeben. Es wird ein Ontologie-basierter Ansatz zur Optimierung des Entwurfsprozesses von automobilelektronischen Systemen vorgestellt. Dazu wird beschrieben, wie dieser Ansatz Ontologien nutzt, um Entwurfsmethoden und -modelle zu konsolidieren und zu integrieren. Wesentlicher Beitrag ist der Vorschlag einer einheitlichen Basis für die Ontologie-basierte Entwicklung und Ausführung MPAD-Anwendungen – die semantische MPAD Plattform. Dazu wird beschrieben, wie das Konzept des MPAD um den Einsatz von Ontologien erweitert wird. Außerdem werden Anforderungen an die Plattform definiert und eine Methodik für die Entwicklung mit der Plattform zur Realisierung eines Ontologie-gestützten MPAD beschrieben. Das Lösungskonzept zur semantischen MPAD Plattform wurde weiterhin prototypisch als Softwareplattform implementiert. In Bezug auf diese Implementierung beschreibt die Arbeit, welche Alternativen sich für die Implementierung der Plattform anbieten und die grundlegende Architektur der Plattform. Weiterhin wird beschrieben, welche Mittel für die Integration der Plattform in existierende Umgebungen beziehungsweise Systeme eingesetzt werden und über welche Grundfunktionen und entsprechende Methoden die REST-Schnittstelle der Plattform verfügt. Außerdem werden verschiedene Ansätze zur Abbildung von MPs auf Ontologien beschrieben. Weitere Beiträge dieser Arbeit sind die „Reasoning-gestützte Robustness Validation“, eine Methode zur Unterstützung bei der Analyse und Sicherstellung der Robustheit von automobilelektronischen Systemen, die „Ontologie-gestützte Change Impact Analysis“, eine Methode zur Verwaltung und Korrelation von Entwurfsparametern und Anforderungen zur Untersuchung der Auswirkungen von Anforderungsänderungen sowie

die „Ontologie-basierte Transformation von Anforderungen“, eine Methode zur Transformation von Anforderungen unter Berücksichtigung der Anwendbarkeit von Fehlermodellen. Diese Methoden wurden alle als auf der semantischen Plattform zur Realisierung eines Ontologie-gestützten MPAD aufsetzenden Anwendungen implementiert. Jede dieser Methoden stellt eine exemplarische Methode dar, die unter Nutzung von Ontologien auf der Basis von Entwurfsmodellen automatische Schlussfolgerungen zur Verbesserung des Entwurfsprozesses automobilelektronischer Systeme ziehen kann. Jede der vorgestellten Methoden wurde zudem anhand konkreter Fallbeispiele untersucht und evaluiert. Die „Reasoning-gestützte Robustness Validation“ zielt auf die Automatisierung eines Schrittes im Ablauf der RV ab. Dazu wird ein Ansatz vorgestellt, der die Menge der durchzuführenden Analysen und dafür passende Daten für eine konkrete Komponente anhand deren Eigenschaften und der Umgebung im späteren Einsatz ermittelt. Es wird beschrieben, wie die Menge der durchzuführenden Analysen unter Einsatz von Ontologien und Inferenzmaschinen bestimmt wird und wie passende Daten für die durchzuführenden Analysen ausgewählt werden können. Weiterhin wird eine Verallgemeinerung des Ansatzes beschrieben, die die Einsetzbarkeit bei anderen Analysemethoden als die RV ermöglicht. Neben der Beschreibung der Methode selbst werden zwei zugehörige Fallbeispiele im Rahmen der Evaluation dargestellt. Beide Fallbeispiele zeigen, dass durch die Teil-Automatisierung der Erstellung des RV-Plans die Komplexität der RV verringert werden und Fehler vermieden werden können. Es geht hervor, dass die Ontologiesprache OWL sich eignet, bereits teilweise formalisiertes Wissen darzustellen und zu verarbeiten. Die Kombination von OWL mit der Regel-Sprache SWRL erweist sich zudem als nützlich. Die Regel-Sprache SWRL wird in einem ersten Fallbeispiel für die Propagierung von einbauortspezifischen Charakteristika eingesetzt und ist intuitiver als eine Menge von Axiomen. Das zweite Fallbeispiel zur „Reasoning-gestützten Robustness Validation“ zeigt, dass die Methode sich auch auf weitere Einsatzgebiete übertragen lässt und somit deren Anwendungsbereich über die Unterstützung, der RV hinausgeht. Die „Ontologie-gestützte Change Impact Analysis“ ermöglicht eine Analyse der Auswirkungen von Änderungen an Anforderungen und ist sowohl auf Software- als auch auf Hardware-Entwürfe anwendbar. Die Methode nutzt die semantische MPAD Plattform für die Abbildung von MPs und Anforderungen auf Ontologien. Die Arbeit beschreibt, wie durch Anforderungs-Rückverfolgbarkeit und gesammeltem Expertenwissen über Entwurfsparameter und Wirkzusammenhänge zwischen diesen eine solche Analyse erreicht wird. Dazu wird das Konzept des Lösungsansatzes, der Entwurf eines unterstützenden Rahmenwerks, der Zusammenhang zwischen Systemmodell und Wissensbasis, der grundlegende Ablauf sowie die Identifikation von Verknüpfungen zwischen Anforderungen und Entwurfsparametern beschrieben. Die Methode wird anhand eines Fallbeispiels evaluiert, welches ein Modul des Entwurfswerkzeugs TAMTAMS zur Analyse von CMOS-Schaltungen als Expertenwissen über Entwurfsparameter

und Wirkzusammenhänge zwischen diesen heranzieht. Das Fallbeispiel zeigt, dass es möglich ist, die Komplexität einer CIA durch die Automatisierung der Analyse betroffener Entwurfsparameter zu verringern, wodurch Kosten eingespart werden können. Durch die Automatisierung können zudem Fehler, die bei manueller Analyse auftreten können, vermieden werden. Durch die Darstellung der Entwurfsparameter und der Wirkzusammenhänge zwischen diesen in OWL-Ontologien können diese in weitere Analysen einfließen und zum besseren Verständnis des Entwurfs beitragen. Die „Ontologie-basierte Transformation von Anforderungen“ unterstützt bei der Transformation von Anforderungen, die in Form von MPs gegeben sind. Die Methode nutzt die semantische MPAD Plattform für die Abbildung von MPs auf Ontolgien. Die Transformation von MPs ist ein, im Entwicklungsprozess von automobilelektronischen Systemen notwendiger Schritt, bei der Weitergarbe von Anforderungen entlang der Lieferkette. Im Rahmen der Beschreibung der Methode werden Anforderungen an ein Transformationssystem für MPs definiert und ein Lösungsansatz für die Umsetzung eines solchen Systems vorgestellt. Die Beschreibung des Lösungsansatzes stellt dar, wie eine Berücksichtigung der Lieferkette erreicht werden kann. Genauer, wird zur Berücksichtigung der Lieferkette beschrieben, wie es erreicht wird, dass MP-Parameter identifiziert werden können, die auf späteren Ebenen der Lieferkette zur Evaluation von Fehlermodellen benötigt werden. Die Methode wird anhand von zwei Fallbeispielen evaluiert. Das erste Fallbeispiel zeigt, dass der Schutz geistigen Eigentums verbessert werden kann, indem für weitere Transformationen nicht erforderliche Informationen aus MPs entfernt werden. Dieses Fallbeispiel bezieht sich auf den Fall, dass für die Herstellung einer Komponente mit vier Stufen in der Lieferkette ermittelt werden muss, welche Parameter bei der Weitergabe und der damit verbundenen Transformation des MP für eine Komponente beizubehalten sind, damit Fehlermodelle auf der Technologieebene ausgewertet werden können. Das Fallbeispiel zeigt, dass MP-Parameter, die auf nachfolgenden Ebenen der Lieferkette für die Evaluation von Fehlermodellen benötigt werden, durch die beschriebene Methode beibehalten werden können. Dadurch kann der Kommunikationsaufwand insgesamt verringert werden. Im zweiten Fallbeispiel zu dieser Methode soll anhand eines Basis-MP für eine ECU ein MP für eine integrierte Schaltung, in der ECU für den entsprechenden Zulieferer abgeleitet werden. Das Fallbeispiel zeigt, dass die vom vorgeschlagenen Transformationssystem bereitgestellten Mechanismen zur Wiederverwendung von Transformationen effektiv sind und sich eignen, um MPs für Subkomponenten abzuleiten. Zudem zeigt das Fallbeispiel, dass auch komplexe Sachverhalte in Bezug auf die zu transformierenden MP-Parameter sich in Transformationsdefinitionen darstellen und auch wiederverwenden lassen.

## Ausblick

Zukünftige Arbeiten können sich um die Verbesserung und Erweiterung der Plattform drehen. Eine Möglichkeit zu deren Verbesserung wäre deren Beschreibung durch den Einsatz von Web Ontology Language for Web Services (OWL-S) [172; 173]. Die Plattform verfügt bereits über eine Web-Schnittstelle. Es wäre daher auch denkbar, die Plattform nicht nur lokal, sondern auch global bereitzustellen, wodurch sie bei einem breiteren Anwenderkreis eingesetzt werden könnte. In diesem Zusammenhang wäre ein Repository verschiedener auf der Plattform basierender Werkzeuge möglich, was die Wiederverwendung von Werkzeugen verbessern würde. Auch in Hinsicht auf die speziellen Anwendungen sind Fortsetzungen der Arbeiten möglich. Die Reasoning-gestützte RV könnte durch die in Abschnitt 5.8 beschriebenen nächsten Schritte fortgesetzt werden. Die Ontologie-gestützte CIA könnte in weitere Aktivitäten der Anforderungsverwaltung integriert werden. Eine Möglichkeit dazu wäre bspw. die Sicherstellung der Vollständigkeit von Anforderungskatalogen, was als die schwerste Aufgabe in der Anforderungsanalyse identifiziert wurde [174]. Dies könnte bspw. dadurch geschehen, dass Entwurfsparameter identifiziert werden, welchen keine Anforderungen zugeordnet sind. Bei der Ontologie-basierten Transformation von Anforderungen ließe sich eine unmittelbare Auswertung von Fehlermodellen untersuchen. Dies würde den Prozess der Anforderungstransformation mit dem der Fehlermodellauswertung verknüpfen. In Bezug auf die Benutzerfreundlichkeit wären außerdem GUIs für die verschiedenen Anwendungen denkbar. Dies könnte auch im Zusammenhang mit einer Erweiterung der GUI der Plattform stehen.

# Anhang A OWLAPI-basierte Transformation

```
class IncreaseTemperatureCycleRange(
    sourcePath: String,
    targetPath: String,
    rangeIncrease: Float) extends OWLAPITransformation {
  override val metadata = Map("description" -> "temperature range increase")
  override val mappings = List(new OneToOneMapping(
    new DataSource { override val metadata = Map("path" -> sourcePath) },
    new DataTarget { override val metadata = Map("path" -> targetPath) })
  )
  override def apply(mapping: Mapping, manager: OWLOntologyManager) = {
    assert(mapping.sources.size == 1 && mapping.targets.size == 1)
    assert(mapping.sources(0).metadata.isDefinedAt("path"))
    assert(mapping.sources(0).metadata("path").isInstanceOf[String])
    assert(mapping.targets(0).metadata.isDefinedAt("path"))
    assert(mapping.targets(0).metadata("path").isInstanceOf[String])
    val source = manager.loadOntologyFromOntologyDocument(
      new File(mapping.sources(0).metadata("path").asInstanceOf[String]))
    val target = manager.createOntology(
      IRI.create("transformed"), Set(source).asJava)
    manager.setOntologyDocumentIRI(target, IRI.create(
      new File(mapping.targets(0).metadata("path").asInstanceOf[String])))
    manager.setOntologyFormat(target, manager.getOntologyFormat(source))

    // TRANSFORMATION OPERATIONS

    // Apply the changes and save the result
    manager.applyChanges(axiomsToRemove.map(ax =>
      new RemoveAxiom(target, ax)).asJava)
    manager.applyChanges(axiomsToAdd.map(ax =>
      new AddAxiom(target, ax)).asJava)
    manager.saveOntology(target)
  }
}
```

**Quelltext A.1** – Scala Quelltext Vorlage für eine OWLAPI-basierte Transformation

# Anhang B Definitionen der Fehlermodell- und Transformations-Ontologie

## OWL-Klassen

Die Parameter-Klasse wird durch genau einen Namen und genau einer Einheit durch die OWL-Datentyp-Eigenschaften hasName beziehungsweise hasUnit definiert.

MissionProfileParameter Diese Klasse beschreibt einen MP-Parameter und ist eine Unterklasse der Parameter-Klasse. Die MissionProfileParameter-Klasse ist disjunkt mit der Klasse TransformationParameter.

ProductParameter Diese Klasse beschreibt einen Produkt-Parameter und ist eine Unterklasse der Parameter-Klasse. Außerdem ist die ProductParameter-Klasse disjunkt mit der Klasse TransformationParameter sowie auch mit der Klasse TechnologyParameter.

TransformationParameter Diese Klasse beschreibt einen Parameter der Transformation und ist eine Unterklasse der Parameter-Klasse. Außerdem ist die Klasse der TransformationParameter disjunkt mit der Klasse MissionProfileParameter, sowie der Klasse ProductParameter und der Klasse TechnologyParameter.

TechnologyParameter Diese Klasse beschreibt einen Parameter der Technologie und ist eine Unterklasse der Parameter-Klasse. Außerdem ist die TechnologyParameter-Klasse disjunkt mit der Klasse TransformationParameter, sowie der Klasse ProductParameter.

TechnologicalFailure Diese Klasse beschreibt einen Fehlermechanismus der Technologie und ist eine Unterklasse der Failure-Klasse.

HighLevelFailure Diese Klasse beschreibt einen Fehlermechanismus auf einer höheren Ebene als der Technologie-Ebene und ist eine Unterklasse der Failure-Klasse. Außerdem enthält jede Instanz dieser Klasse gar keine, eine oder mehrere Ursachen auf Technologieebene, was durch die OWL-Objekt-Eigenschaft hasRootFailure ausgedrückt wird.

FailureModel Diese Klasse beschreibt ein Fehlermodell und wird durch mindestens eine Gleichung, mindestens einen MP-Parameter und mindestens einen Fehlermechanismus durch die OWL-Datentyp beziehungsweise Objekt-Eigenschaften hasEquation, hasMissionProfileParameterl und ofFailure definiert.

HighLevelFailureModel Diese Klasse beschreibt ein Fehlermodell auf einer höheren Ebene als der Technologie-Ebene und ist eine Unterklasse der FailureModel-Klasse.

MPOTransformation Diese Klasse beschreibt eine Transformation einer MP-Ontologie und wird durch gar keine, eine oder mehrere Transformationsparameter, gar keine, einen oder mehrere beizubehaltende MP-Parameter und genau eine Ebene der Lieferkette durch die OWL-Objekt-Eigenschaften maintainsParameter, hasTransformationParameter und ofSupplyChainLevel definiert.

Layer Diese Klasse beschreibt die Technologieebene und setzt sich aus den beiden Instanzen BEOL und FEOL zusammen.

SupplyChainLevel Diese Klasse beschreibt die Ebene der Lieferkette und setzt sich aus beliebig vielen Instanzen zusammen (OEM, Tier1, Tier2, etc.).

## OWL-Eigenschaften

### OWL-Objekt-Eigenschaften

hasMissionProfileParameter Diese Eigenschaft verbindet eine Instanz einer beliebigen Klasse mit einer Instanz der MissionProfileParameter-Klasse.

hasProductParameter Diese Eigenschaft verbindet eine Instanz einer beliebigen Klasse mit einer Instanz der ProductParameter-Klasse.

hasTechnologyParameter Diese Eigenschaft verbindet eine Instanz einer beliebigen Klasse mit einer Instanz der TechnologyParameter-Klasse.

hasTransformationParameter Diese Eigenschaft verbindet eine Instanz einer beliebigen Klasse mit einer Instanz der TransformationParameter-Klasse.

maintainsParameter Diese Eigenschaft verbindet eine Instanz einer beliebigen Klasse mit einer Instanz der MissionProfileParameter-Klasse.

hasRootFailure Diese Eigenschaft verbindet eine Instanz der Failure-Klasse mit einer Instanz der Failure-Klasse.

ofFailure Diese Eigenschaft verbindet eine Instanz einer beliebigen Klasse mit einer Instanz der Failure-Klasse.

ofSupplyChainLevel Diese Eigenschaft verbindet eine Instanz einer beliebigen Klasse mit einer Instanz der SupplyChainLevel-Klasse.

### OWL-Datentyp-Eigenschaften

hasEquation Diese Eigenschaft verbindet eine Instanz einer beliebigen Klasse, mit einer Zeichenkette, welche eine Gleichung ausdrückt.

hasName Diese Eigenschaft verbindet eine Instanz einer beliebigen Klasse mit einer Zeichenkette, welche einen Namen ausdrückt.

hasUnit Diese Eigenschaft verbindet eine Instanz einer beliebigen Klasse mit einer Zeichenkette, welche eine Einheit ausdrückt.

hasValue Diese Eigenschaft verbindet eine Instanz der Parameter-Klasse, die diese Eigenschaft enthält, mit einer Zeichenkette, welche eine Einheit ausdrückt.

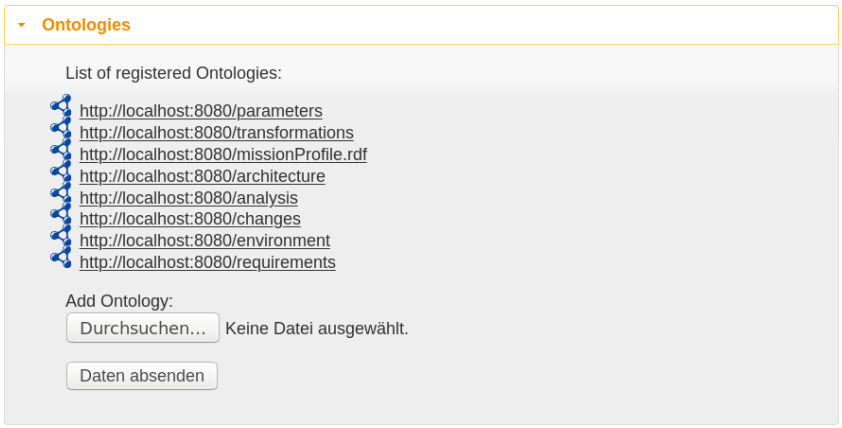
## OWL-Instanzen

BEOL Instanz der Klasse Layer die das *Back End of Line* repräsentiert.

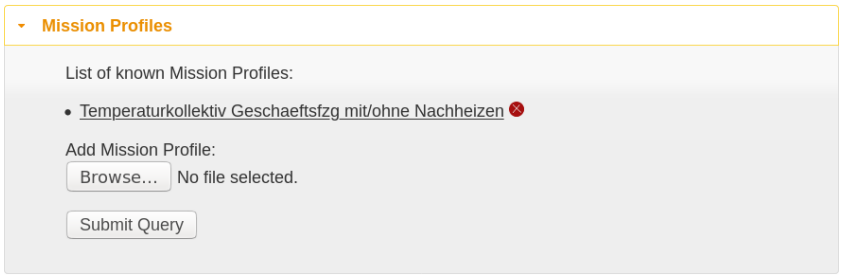
FEOL Instanz der Klasse Layer die das *Front End of Line* repräsentiert.

OEMLevel Instanz der Klasse SupplyChainLevel die die OEM-Ebene repräsentiert.

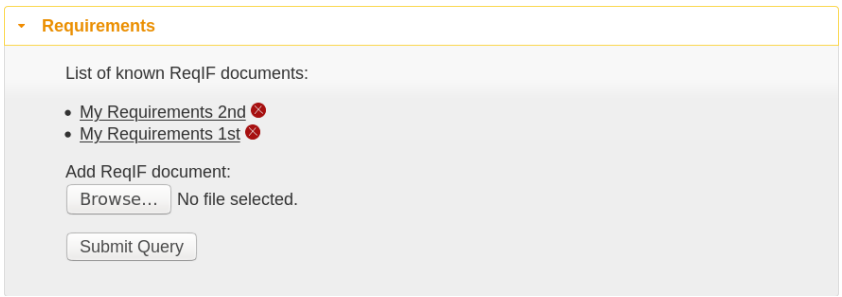
# Anhang C Benutzerschnittstelle




(a) Liste der Back-End Ontologien mit der Möglichkeit zum Hinzufügen weiterer Ontologien




(b) Liste transformierter MPs mit den Möglichkeiten zur Transformation eines MPs nach dem MPFO, sowie zum Herunterladen und Entfernen




(c) Liste transformierter Anforderungsontologien mit den Möglichkeiten zur Transformation eines ReqIF Dokuments, zum Herunterladen und zum Entfernen

**Abbildung C.1** – Ausschnitte der Web-GUI des Prototypen der Plattform

# Literaturverzeichnis

## Eigene Veröffentlichungen


[1] J. Novacek, A. Viehl, O. Bringmann und W. Rosenstiel, „Reasoning-supported Robustness Validation of Automotive E/E Components“, *IEEE 11th International Conference on Semantic Computing (ICSC)*, Jan. 2017. Adresse: https://doi.org/10.1109/ICSC.2017.28.

[2] J. Novacek, A. Viehl, O. Bringmann und W. Rosenstiel, „Reasoning-supported Robustness Validation of Automotive E/E Components“, *International Journal of Semantic Computing*, Jg. 11, Nr. 4, Dez. 2017. Adresse: http://dx.doi.org/10.1142/S1793351X17400190.

[3] J. Novacek, A. Ahari, A. Cornaglia, F. Haxel, A. Viehl, O. Bringmann und W. Rosenstiel, „Ontology-supported Design Parameter Management for Change Impact Analysis“, *Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Aug. 2018. Adresse: https://doi.org/10.1109/SEAA.2018.00011.

[4] J. Novacek, A. Viehl, O. Bringmann und W. Rosenstiel, „Ontology-based Requirements Transformation“, in *2019 International Symposium on Systems Engineering (ISSE) (IEEE ISSE 2019)*, Edinburgh, United Kingdom (Great Britain), Okt. 2019.


## Beteiligte Veröffentlichungen


[5] A. Ahari, J. Novacek, A. Viehl, O. Bringmann und W. Rosenstiel, „CTEF: Collaborative Technology Evaluation Framework“, in *2018 IEEE International Systems Engineering Symposium (ISSE)*, Okt. 2018, S. 1–7. Adresse: https://doi.org/10.1109/SysEng.2018.8544418.


## Literaturquellen


[6] M. Aeberhard, S. Rauch, M. Bahram, G. Tanzmeister, J. Thomas, Y. Pilat, F. Homm, W. Huber und N. Kaempchen, „Experience, results and lessons

learned from automated driving on Germany's highways“, *IEEE Intelligent transportation systems magazine*, Jg. 7, Nr. 1, S. 42–57, 2015.

[7] R. Hoeger, A. Amditis, M. Kunert, A. Hoess, F. Flemisch, H.-P. Krueger, A. Bartels, A. Beutner und K. Pagle, „Highly automated vehicles for intelligent transport: HAVEit approach“, in *ITS World Congress, NY, USA*, 2008.

[8] M. Hummel, M. Kühn, J. Bende und A. Lang, „Fahrerassistenzsysteme. Ermittlung des Sicherheitspotenzials auf Basis des Schadengeschehens der Deutschen Versicherer“, *Forschungsbericht/Unfallforschung der Versicherer (UDV), Reihe Fahrzeugsicherheit*, Nr. 03, 2011.

[9] G. Eisele, M. Kauth, C. Steffens und P. Glusk, „Automotive megatrends and their impact on NVH“, in *19. Internationales Stuttgarter Symposium*, M. Bargende, H.-C. Reuss, A. Wagner und J. Wiedemann, Hrsg., Wiesbaden: Springer Fachmedien Wiesbaden, 2019, S. 523–539.

[10] C. Byrne und Zentralverband Elektrotechnik und Elektronikindustrie, *Handbook for robustness validation of automotive electrical/electronic modules*. ZVEI, 2008.

[11] Springer India-New Delhi, „Emerging Trends, Technologies in the Automotive Sector“, *Auto Tech Review*, Jg. 4, Nr. 11, S. 18–23, Nov. 2015. Adresse: `https://doi.org/10.1365/s40112-015-1025-3`.

[12] J. D'Ambrosio und G. Soremekun, „Systems engineering challenges and MBSE opportunities for automotive system design“, in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Okt. 2017, S. 2075–2080.

[13] L. Wang, M. Izygon, S. Okon, H. Wagner und L. Garner, „Effort to Accelerate MBSE Adoption and Usage at JSC“, in *AIAA SPACE 2016*, 2016, S. 5542.

[14] O. Chourabi, Y. Pollet und M. B. Ahmed, „Ontology based knowledge modeling for System Engineering projects“, in *Second International Conference on Research Challenges in Information Science, 2008. RCIS 2008.*, IEEE, 2008, S. 453–458.

[15] U. Abelein, H. Lochner, D. Hahn und S. Straube, „Complexity, quality and robustness - the challenges of tomorrow's automotive electronics“, in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2012*, IEEE, 2012, S. 870–871.

[16] G. Jerke und A. B. Kahng, „Mission profile aware IC design: a case study“, in *Proceedings of the conference on Design, Automation & Test in Europe*, European Design und Automation Association, 2014, S. 64.

[17] T. Nirmaier, A. Burger, G. Harrant, A. Viehl, O. Bringmann, W. Rosenstiel und G. Pelz, „Mission profile aware robustness assessment of automotive power devices“, in *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*, 2014, S. 1–6.

[18] A. J. Gerber, A. Barnard und A. J. Van der Merwe, „Towards a semantic web layered architecture“, 2007.

[19] A. Gerber, A. Van der Merwe und A. Barnard, „A functional semantic web architecture“, in *European Semantic Web Conference*, Springer, 2008, S. 273–287.

[20] I. Horrocks, B. Parsia, P. Patel-Schneider und J. Hendler, „Semantic web architecture: Stack or two towers?“, in *International Workshop on Principles and Practice of Semantic Web Reasoning*, Springer, 2005, S. 37–41.

[21] T. Berners-Lee u. a., *Semantic web road map*, 1998.

[22] N. Bikakis, C. Tsinaraki, N. Gioldasis, I. Stavrakantonakis und S. Christodoulakis, „The XML and semantic web worlds: technologies, interoperability and integration: a survey of the state of the art“, in *Semantic Hyper/Multimedia Adaptation*, Springer, 2013, S. 319–360.

[23] W. W. W. Consortium u. a., „Extensible markup language (XML) 1.1“, 2006.

[24] D. Beckett und B. McBride, „RDF/XML syntax specification (revised)“, *W3C recommendation*, Jg. 10, Nr. 2.3, 2004.

[25] G. Klyne und J. J. Carroll, „Resource description framework (RDF): Concepts and abstract syntax“, 2006.

[26] P. J. Hayes und P. F. Patel-Schneider, „RDF 1.1 Semantics“, *W3C recommendation*, Jg. 25, S. 7–13, 2014.

[27] F. Manola, E. Miller, B. McBride u. a., „RDF primer“, *W3C recommendation*, Jg. 10, Nr. 1-107, S. 6, 2004.

[28] S. Weibel, J. Kunze, C. Lagoze und M. Wolf, „Dublin core metadata for resource discovery“, Techn. Ber., 1998.

[29] D. Brickley, „RDF vocabulary description language 1.0: RDF schema“, *http://www. w3. org/TR/rdf-schema/*, 2004.

[30] B. Beckert, *Formale Systeme - Prädikatenlogik: Syntax*, Unveröffentlichte Vorlesungsmaterialien, Karlsruher Institut für Technologie (KIT) - Institut für Theoretische Informatik, 2010.

[31] R. R. Smullyan, *First-order logic*. Springer Science & Business Media, 2012, Bd. 43.

[32] H. Enderton und H. B. Enderton, *A mathematical introduction to logic*. Elsevier, 2001.

[33] M. Fitting, *First-order logic and automated theorem proving*. Springer Science & Business Media, 2012.

[34] M. R. Quillian, „Word concepts: A theory and simulation of some basic semantic capabilities“, *Behavioral science*, Jg. 12, Nr. 5, S. 410–430, 1967.

[35] R. J. Brachman und J. G. Schmolze, „An overview of the KL-ONE knowledge representation system“, in *Readings in artificial intelligence and databases*, Elsevier, 1989, S. 207–230.

[36] M. Schmidt-Schauß und G. Smolka, „Attributive concept descriptions with complements“, *Artificial intelligence*, Jg. 48, Nr. 1, S. 1–26, 1991.

[37] F. Baader, D. Calvanese, D. McGuinness, P. Patel-Schneider und D. Nardi, *The description logic handbook: Theory, implementation and applications*. Cambridge university press, 2003.

[38] J. Wissmann, T. Weyde und D. Conklin, „Chord sequence patterns in owl“, in *Proceedings of SMC Conference*, Citeseer, 2010.

[39] P. Hitzler, M. Krötzsch, S. Rudolph und Y. Sure, *Semantic Web: Grundlagen*. Springer-Verlag, 2007.

[40] I. Horrocks, O. Kutz und U. Sattler, „The Even More Irresistible SROIQ.“, *Kr*, Jg. 6, S. 57–67, 2006.

[41] G. Antoniou und F. Van Harmelen, „Web ontology language: Owl“, in *Handbook on ontologies*, Springer, 2004, S. 67–92.

[42] M. Schmidt-Schauß und D. Sabel, „Einführung in die Methoden der künstlichen Intelligenz“, *Institut für Informatik, Goethe-Universität Frankfurt am Main*, 2013.

[43] B. Motik, B. C. Grau, I. Horrocks, Z. Wu, A. Fokoue, C. Lutz u. a., „OWL 2 web ontology language profiles“, *W3C recommendation*, Jg. 27, S. 61, 2009.

[44] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean u. a., „SWRL: A Semantic Web rule language combining OWL and RuleML“, *W3C Member submission*, Jg. 21, S. 79, 2004.

[45] D. Carral, M. Knorr und A. Krishnadhi, „OWL + Rules = .. ?“, *10th Extended Semantic Web Conference (ESWC)*, 2013.

[46] E. Prud'Hommeaux, A. Seaborne u. a., „SPARQL query language for RDF“, *W3C recommendation*, Jg. 15, 2008.

[47] P. Haase, J. Broekstra, A. Eberhart und R. Volz, „A comparison of RDF query languages“, in *International Semantic Web Conference*, Springer, 2004, S. 502–517.

[48] D. Beckett, T. Berners-Lee, E. Prud'hommeaux und G. Carothers, „RDF 1.1 Turtle“, *World Wide Web Consortium*, 2014.

[49] D. Brickley und L. Miller, *FOAF vocabulary specification 0.91*, 2007.

[50] C. Beierle und G. Kern-Isberner, *Methoden wissensbasierter Systeme: Grundlagen, Algorithmen, Anwendungen*. Springer-Verlag, 2014.

[51] F. Hayes-Roth, D. Waterman und D. Lenat, „Building expert systems“, Jan. 1983.

[52] F. Puppe, *Problemlösungsmethoden in Expertensystemen*. Springer-Verlag, 2013.

[53] R. Brachman und H. Levesque, *Knowledge Representation and Reasoning*. Elsevier, 2004.

[54] M. Richter, *Prinzipien der künstlichen Intelligenz: Wissensrepräsentation, Inferenz und Expertensysteme*. Springer-Verlag, 2013.

[55] W. Bibel, S. Hölldobler und T. Schaub, *Wissensrepräsentation und Inferenz: eine grundlegende Einführung*. Springer-Verlag, 2013.

[56] T. Berners-Lee, *Linked data*, 2006.

[57] C. Bizer, T. Heath und T. Berners-Lee, „Linked Data - The Story So Far“, *Semantic Services, Interoperability and Web Applications: Emerging Concepts*, S. 205–227, 2009.

[58] T. Heath und C. Bizer, „Linked data: Evolving the web into a global data space“, *Synthesis lectures on the semantic web: theory and technology*, Jg. 1, Nr. 1, S. 1–136, 2011.

[59] B. Jäger und Zentralverband Elektrotechnik und Elektronikindustrie, *Robustness Validation for MEMS*. ZVEI, 2009.

[60] Automotive Electronics Council, „Failure mechanism based stress test qualification for integrated circuits“, 2014.

[61] C. Ebert und M. Jastram, „ReqIF: Seamless requirements interchange format between business partners“, *IEEE software*, Jg. 29, Nr. 5, S. 82–87, 2012.

[62] H. Stachowiak, „Allgemeine Modelltheorie“, 1973.

[63] M. Brambilla, J. Cabot und M. Wimmer, „Model-driven software engineering in practice“, *Synthesis Lectures on Software Engineering*, Jg. 3, Nr. 1, S. 1–207, 2017.

[64] O. Kautz, A. Roth und B. Rumpe, *Achievements, Failures, and the Future of Model-Based Software Engineering.* 2018.

[65] C. Atkinson, M. Gutheil und K. Kiko, „On the Relationship of Ontologies and Models.“, *WoMM*, Jg. 96, S. 47–60, 2006.

[66] D. C. Schmidt, „Model-driven engineering“, *COMPUTER-IEEE COMPUTER SOCIETY-*, Jg. 39, Nr. 2, S. 25, 2006.

[67] A. Vallecillo, „On the industrial adoption of model driven engineering. Is your company ready for MDE?“, *International Journal of Information Systems and Software Engineering for Big Companies (IJISEBC)*, Jg. 1, Nr. 1, S. 52–68, 2015.

[68] A. R. Da Silva, „Model-driven engineering: A survey supported by the unified conceptual model“, *Computer Languages, Systems & Structures*, Jg. 43, S. 139–155, 2015.

[69] G. Mussbacher, D. Amyot, R. Breu, J.-M. Bruel, B. H. Cheng, P. Collet, B. Combemale, R. B. France, R. Heldal, J. Hill u. a., „The relevance of model-driven engineering thirty years from now“, in *International Conference on Model Driven Engineering Languages and Systems*, Springer, 2014, S. 183–200.

[70] J. Miller und J. Mukerji, Hrsg., *Model Driven Architecture (MDA): MDA Guide rev 2.0*, OMG: Object Management Group, Juni 2014.

[71] F. Vares, M. J. Amiri und S. Parsa, „Towards a model-driven development of enterprise systems“, in *2017 International Symposium on Computer Science and Software Engineering Conference (CSSE)*, Okt. 2017, S. 42–48.

[72] C. Atkinson und T. Kuhne, „Model-driven development: a metamodeling foundation“, *IEEE software*, Jg. 20, Nr. 5, S. 36–41, 2003.

[73] D. Ameller, „Considering Non-Functional Requirements in Model-Driven Engineering“, Magisterarb., Llenguatges i Sistemes Informàtics (LSI), Juni 2009. Adresse: `http://upcommons.upc.edu/pfc/handle/2099.1/7192`.

[74] J. Cabot, „Clarifying concepts: MBE vs MDE vs MDD vs MDA“, 2015. Adresse: `http://modeling-languages.com/clarifying-concepts-mbe-vs-mde-vs-mdd-vs-mda/`.

[75] R. Acerbis, A. Bongio, M. Brambilla, M. Tisi, S. Ceri und E. Tosetti, „Developing eBusiness Solutions with a Model Driven Approach: The Case of Acer EMEA“, in *Web Engineering*, L. Baresi, P. Fraternali und G.-J. Houben, Hrsg., Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, S. 539–544.

[76] J. Hutchinson, M. Rouncefield und J. Whittle, „Model-driven engineering practices in industry“, in *Proceedings of the 33rd International Conference on Software Engineering*, ACM, 2011, S. 633–642.

[77] J. Hutchinson, J. Whittle, M. Rouncefield und S. Kristoffersen, „Empirical assessment of MDE in industry“, in *Proceedings of the 33rd international conference on software engineering*, ACM, 2011, S. 471–480.

[78] P. Mohagheghi und V. Dehlen, „Where is the proof?-A review of experiences from applying MDE in industry“, in *European Conference on Model Driven Architecture-Foundations and Applications*, Springer, 2008, S. 432–443.

[79] J. Whittle, J. Hutchinson und M. Rouncefield, „The state of practice in model-driven engineering“, *IEEE software*, Jg. 31, Nr. 3, S. 79–85, 2014.

[80] J.-P. Tolvanen und S. Kelly, „Model-driven development challenges and solutions: experiences with domain-specific modelling in industry“, in *2016 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*, IEEE, 2016, S. 711–719.

[81] J.-M. Jézéquel, „Model-driven engineering for software product lines“, *ISRN Software Engineering*, Jg. 2012, 2012.

[82] P. Liggesmeyer und M. Trapp, „Trends in embedded software engineering“, *IEEE software*, Jg. 26, Nr. 3, S. 19–25, 2009.

[83] K. Pohl, H. Hönninger, R. Achatz und M. Broy, *Model-Based Engineering of Embedded Systems: The SPES 2020 Methodology*. Springer Science & Business Media, 2012.

[84] T. M. Shortell, *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. John Wiley & Sons, 2015.

[85] S. Friedenthal, R. Griego und M. Sampson, „INCOSE model based systems engineering (MBSE) initiative“, in *INCOSE 2007 Symposium*, Bd. 11, 2007.

[86] S. Sendall und W. Kozaczynski, „Model transformation: The heart and soul of model-driven software development“, *IEEE software*, Jg. 20, Nr. 5, S. 42–45, 2003.

[87] K. Czarnecki und S. Helsen, „Classification of model transformation approaches“, in *Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture*, USA, Bd. 45, 2003, S. 1–17.

[88] K. Czarnecki und S. Helsen, „Feature-based survey of model transformation approaches“, *IBM Systems Journal*, Jg. 45, Nr. 3, S. 621–645, 2006.

[89] N. Kahani, M. Bagherzadeh, J. R. Cordy, J. Dingel und D. Varró, „Survey and classification of model transformation tools“, *Software & Systems Modeling*, S. 1–37, 2018.

[90] E. Guerra, J. de Lara, M. Wimmer, G. Kappel, A. Kusel, W. Retschitzegger, J. Schönböck und W. Schwinger, „Automated verification of model transformations based on visual contracts“, *Automated Software Engineering*, Jg. 20, Nr. 1, S. 5–46, 2013.

[91] T. Mens und P. Van Gorp, „A taxonomy of model transformation“, *Electronic Notes in Theoretical Computer Science*, Jg. 152, S. 125–142, 2006.

[92] M. Tisi, F. Jouault, P. Fraternali, S. Ceri und J. Bézivin, „On the use of higher-order model transformations“, in *European Conference on Model Driven Architecture-Foundations and Applications*, Springer, 2009, S. 18–33.

[93] E. Visser, „A survey of strategies in rule-based program transformation systems“, *Journal of symbolic computation*, Jg. 40, Nr. 1, S. 831–873, 2005.

[94] L. Lúcio, M. Amrani, J. Dingel, L. Lambers, R. Salay, G. M. Selim, E. Syriani und M. Wimmer, „Model transformation intents and their properties“, *Software & systems modeling*, Jg. 15, Nr. 3, S. 647–684, 2016.

[95] T. Gardner, C. Griffin, J. Koehler und R. Hauser, „A review of OMG MOF 2.0 Query/Views/Transformations Submissions and Recommendations towards the final Standard“, in *MetaModelling for MDA Workshop*, Bd. 13, 2003, S. 41.

[96] D. H. Akehurst, B. Bordbar, M. J. Evans, W. G. J. Howells und K. D. McDonald-Maier, „SiTra: Simple transformations in java“, in *International Conference on Model Driven Engineering Languages and Systems*, Springer, 2006, S. 351–364.

[97] M. Lehman, „On understanding laws, evolution, and conservation in the large-program life cycle“, *Journal of Systems and Software*, Jg. 1, S. 213–221, 1979. Adresse: `http://www.sciencedirect.com/science/article/pii/0164121279900220`.

[98] I. ISO, „26262: Road vehicles-Functional safety“, *International Standard ISO/FDIS*, Jg. 26262, 2011.

[99] M. Borg, K. Wnuk, B. Regnell und P. Runeson, „Supporting Change Impact Analysis Using a Recommendation System: An Industrial Case Study in a Safety-Critical Context“, *IEEE Transactions on Software Engineering*, Jg. 43, Nr. 7, S. 675–700, Juli 2017.

[100] S. A. Bohner und R. S. Arnold, *Software change impact analysis*. IEEE Computer Society Press Los Alamitos, 1996, Bd. 6.

[101] S. L. Pfleeger und J. M. Atlee, *Software engineering: theory and practice*. Pearson Education India, 1998.

[102] S. Lehnert, „A review of software change impact analysis“, *Ilmenau University of Technology, Tech. Rep*, 2011.

[103] M. S. Kilpinen u. a., „The emergence of change at the systems engineering and software design interface“, Diss., University of Cambridge, 2008.

[104] S. Lehnert, „A taxonomy for software change impact analysis“, in *Proceedings of the 12th International Workshop on Principles of Software Evolution and the 7th annual ERCIM Workshop on Software Evolution*, ACM, 2011, S. 41–50.

[105] B. Li, X. Sun, H. Leung und S. Zhang, „A survey of code-based change impact analysis techniques“, *Software Testing, Verification and Reliability*, Jg. 23, Nr. 8, S. 613–646, 2013.

[106] P. Rovegård, L. Angelis und C. Wohlin, „An empirical study on views of importance of change impact analysis issues“, *IEEE Transactions on Software Engineering*, Jg. 34, Nr. 4, S. 516–530, 2008.

[107] S. El Kadiri, W. Terkaj, E. N. Urwin, C. Palmer, D. Kiritsis und R. Young, „Ontology in engineering applications“, in *International Workshop Formal Ontologies Meet Industries*, Springer, 2015, S. 126–137.

[108] H.-J. Happel und S. Seedorf, „Applications of ontologies in software engineering“, in *Proc. of Workshop on Sematic Web Enabled Software Engineering"(SWESE) on the ISWC*, Citeseer, 2006, S. 5–9.

[109] H. Knublauch, „Ontology-driven software development in the context of the semantic web: An example scenario with Protege/OWL“, in *1st International workshop on the model-driven semantic web (MDSW2004)*, Monterey, California, USA.[WWW document] http://www. knublauch. com . . ., 2004, S. 381–401.

[110] A. Ankolekar, „Towards a semantic web of community, content and interactions“, Carnegie-Mellon University Pittsburgh PA School Of Computer Science, Techn. Ber., 2005.

[111] M. V. Bossche, P. Ross, I. MacLarty, B. Van Nuffelen und N. Pelov, „Ontology driven software engineering for real life applications“, in *Proc. 3rd Intl. Workshop on Semantic Web Enabled Software Engineering*, Citeseer, 2007.

[112] R. Jasper, M. Uschold u. a., „A framework for understanding and classifying ontology applications“, in *Proceedings 12th Int. Workshop on Knowledge Acquisition, Modelling, and Management KAW*, Bd. 99, 1999, S. 16–21.

[113] V. Castañeda, L. Ballejos, M. L. Caliusco und M. R. Galli, „The Use of Ontologies in Requirements Engineering“, *Global journal of researches in engineering*, Jg. 10, Nr. 6, S. 2–8, 2010.

[114] D. Dermeval, J. Vilela, I. I. Bittencourt, J. Castro, S. Isotani, P. Brito und A. Silva, „Applications of ontologies in requirements engineering: a systematic review of the literature“, *Requirements Engineering*, Jg. 21, Nr. 4, S. 405–437, 2016.

[115] K. Siegemund, E. J. Thomas, Y. Zhao, J. Pan und U. Assmann, „Towards ontology-driven requirements engineering“, in *Workshop semantic web enabled software engineering at 10th international semantic web conference (ISWC), Bonn*, 2011.

[116] Y. Ait-Ameur, M. Baron, L. Bellatreche, S. Jean und E. Sardet, „Ontologies in engineering: the OntoDB/OntoQL platform“, *Soft Computing*, Jg. 21, Nr. 2, S. 369–389, 2017.

[117] J. Morbach, A. Wiesner und W. Marquardt, „OntoCAPE—A (re) usable ontology for computer-aided process engineering“, *Computers & Chemical Engineering*, Jg. 33, Nr. 10, S. 1546–1556, 2009.

[118] L. van Ruijven, „Ontology for systems engineering“, *Procedia Computer Science*, Jg. 16, S. 383–392, 2013.

[119] M. B. Sarder und S. Ferreira, „Developing systems engineering ontologies“, in *System of Systems Engineering, 2007. SoSE'07. IEEE International Conference on*, IEEE, 2007, S. 1–6.

[120] C. Hennig, H. Eisenmann, A. Viehl und O. Bringmann, „On languages for conceptual data modeling in multi-disciplinary space systems engineering“, in *Model-Driven Engineering and Software Development (MODELSWARD), 2015 3rd International Conference on*, IEEE, 2015, S. 384–393.

[121] C. Hennig, A. Viehl, B. Kämpgen und H. Eisenmann, „Ontology-Based Design of Space Systems“, in *International Semantic Web Conference*, Springer, 2016, S. 308–324.

[122] D. Ernadote, „An ontology mindset for system engineering“, in *Systems Engineering (ISSE), 2015 IEEE International Symposium on*, IEEE, 2015, S. 454–460.

[123] T. Tudorache, „Employing ontologies for an improved development process in collaborative engineering“, *Technical University of Berlin*, 2006.

[124] T. Hagedorn, „Supporting Engineering Design of Additively Manufactured Medical Devices with Knowledge Management Through Ontologies“, 2018.

[125] M. Reke, „Modellbasierte Entwicklung automobiler Steuerungssysteme in kleinen und mittelständischen Unternehmen“, Diss., RWTH Aachen University, 2013.

[126] S. Kugele, „Model-Based Development of Software-intensive Automotive Systems“, Diss., Technische Universität München, 2012.

[127] D. Firesmith, „Common Requirements Problems, Their Negative Consequences, and the Industry Best Practices to Help Solve Them.“, *Journal of Object Technology*, Jg. 6, Nr. 1, S. 17–33, 2007.

[128] S. Farfeleder, T. Moser, A. Krall, T. Stålhane, H. Zojer und C. Panis, „DODT: Increasing requirements formalism using domain ontologies for improved embedded systems development“, in *14th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems*, IEEE, 2011, S. 271–274.

[129] N. Mahmud, C. Seceleanu und O. Ljungkrantz, „ReSA: An ontology-based requirement specification language tailored to automotive systems“, in *10th IEEE International Symposium on Industrial Embedded Systems (SIES)*, IEEE, 2015, S. 1–10.

[130] N. Mahmud, C. Seceleanu und O. Ljungkrantz, „ReSA tool: Structured requirements specification and SAT-based consistency-checking“, in *2016 Federated Conference on Computer Science and Information Systems (FedCSIS)*, IEEE, 2016, S. 1737–1746.

[131] S. Wagner, D. M. Fernández, M. Felderer und M. Kalinowski, „Requirements Engineering Practice and Problems in Agile Projects: Results from an International Survey“, *CoRR*, Jg. abs/1703.08360, 2017. arXiv: `1703.08360`. Adresse: `http://arxiv.org/abs/1703.08360`.

[132] M. A. Musen u. a., „The protégé project: a look back and a look forward“, *AI matters*, Jg. 1, Nr. 4, S. 4, 2015.

[133] S. Zander und R. Awad, „Expressing and reasoning on features of robot-centric workplaces using ontological semantics“, in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2015, S. 2889–2896.

[134] Y. Zhang und X. Luo, „An ontology-based knowledge model for engineering material selections“, in *Proceedings of the 2014 International Conference on Innovative Design and Manufacturing (ICIDM)*, IEEE, 2014, S. 53–58.

[135] M. Ring, J. Stoppe, C. Luth und R. Drechsler, „Change impact analysis for hardware designs from natural language to system level“, in *Specification and Design Languages (FDL), 2016 Forum on*, IEEE, 2016, S. 1–7.

[136] A. Goknil, I. Kurtev und K. v. d. Berg, „A rule-based change impact analysis approach in software architecture for requirements changes“, *arXiv preprint arXiv:1608.02757*, 2016.

[137] S. Roser und B. Bauer, „Ontology-based model transformation“, in *International Conference on Model Driven Engineering Languages and Systems*, Springer, 2005, S. 355–356.

[138] Q. Omg, „Meta object facility (mof) 2.0 query/view/transformation specification“, *Final Adopted Specification (November 2005)*, 2008.

[139] F. Jouault, F. Allilaire, J. Bézivin, I. Kurtev und P. Valduriez, „ATL: a QVT-like transformation language“, in *Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*, ACM, 2006, S. 719–720.

[140] L. Wouters und M.-P. Gervais, „Ontology transformations“, in *2012 IEEE 16th International Enterprise Distributed Object Computing Conference*, IEEE, 2012, S. 71–80.

[141] K. Geihs, P. Baer, R. Reichle und J. Wollenhaupt, „Ontology-based automatic model transformations“, in *2008 Sixth IEEE International Conference on Software Engineering and Formal Methods*, IEEE, 2008, S. 387–391.

[142] O. Šváb-Zamazal, V. Svátek und F. Scharffe, „Pattern-based ontology transformation service“, in *Proc. 1st IK3C international conference on knowledge engineering and ontology development (KEOD)*, No commercial editor., 2009, S. 210–223.

[143] O. Šváb-Zamazal, V. Svátek und L. Iannone, „Pattern-based ontology transformation service exploiting OPPL and OWL-API“, in *International Conference on Knowledge Engineering and Knowledge Management*, Springer, 2010, S. 105–119.

[144] S. Rudolph, „Foundations of description logics“, in *Reasoning Web. Semantic Technologies for the Web of Data*, Springer, 2011, S. 76–136.

[145] Y. Kazakov, „An Extension of Complex Role Inclusion Axioms in the Description Logic $\mathcal{SROIQ}$“, in *Automated Reasoning*, Springer, 2010, S. 472–486.

[146] LV124, Volkswagen, *VW 80000 2013-06*, 2013.

[147] U. Abelein, O. Bringmann, A. Burger u. a., „Robuster Entwurf von neuen Elektronikkomponenten für Anwendungen im Bereich Elektromobilität (RESCAR)“, 2014.

[148] M. J. O'Connor, R. D. Shankar, M. A. Musen, A. K. Das und C. Nyulas, „The SWRLAPI: A Development Environment for Working with SWRL Rules.“, in *OWLED*, 2008.

[149] M. O'connor, H. Knublauch, S. Tu, B. Grosof, M. Dean, W. Grosso und M. Musen, „Supporting rule system interoperability on the semantic web with SWRL“, in *International semantic web conference*, Springer, 2005, S. 974–986.

[150] E. Maricau und G. Gielen, „CMOS reliability overview“, in *Analog IC Reliability in Nanometer CMOS*, Springer, 2013, S. 15–35.

[151] M. Odersky, P. Altherr, V. Cremet, B. Emir, S. Maneth, S. Micheloud, N. Mihaylov, M. Schinz, E. Stenman und M. Zenger, „An overview of the Scala programming language“, Techn. Ber., 2004.

[152] R. E. Johnson und B. Foote, „Designing reusable classes“, *Journal of object-oriented programming*, Jg. 1, Nr. 2, S. 22–35, 1988.

[153] H. Paulheim, D. Oberle, R. Plendl und F. Probst, „An architecture for information exchange based on reference models“, in *International Conference on Software Language Engineering*, Springer, 2011, S. 160–179.

[154] D. S. Kolovos, R. F. Paige und F. A. Polack, „The epsilon transformation language“, in *International Conference on Theory and Practice of Model Transformations*, Springer, 2008, S. 46–60.

[155] C. Puleston, B. Parsia, J. Cunningham und A. Rector, „Integrating object-oriented and ontological representations: A case study in Java and OWL“, in *International Semantic Web Conference*, Springer, 2008, S. 130–145.

[156] S. Bechhofer, R. Volz und P. Lord, „Cooking the Semantic Web with the OWL API“, in *International Semantic Web Conference*, Springer, 2003, S. 659–675.

[157] M. Horridge und S. Bechhofer, „The OWL API: a Java API for working with OWL 2 ontologies“, in *Proceedings of the 6th International Conference on OWL: Experiences and Directions-Volume 529*, CEUR-WS. org, 2009, S. 49–58.

[158] M. Horridge und S. Bechhofer, „The owl api: A java api for owl ontologies“, *Semantic Web*, Jg. 2, Nr. 1, S. 11–21, 2011.

[159] J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne und K. Wilkinson, „Jena: implementing the semantic web recommendations“, in *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, ACM, 2004, S. 74–83.

[160] D. Steinberg, F. Budinsky, E. Merks und M. Paternostro, *EMF: eclipse modeling framework*. Pearson Education, 2008.

[161] G. A. Miller, „WordNet: a lexical database for English“, *Communications of the ACM*, Jg. 38, Nr. 11, S. 39–41, 1995.

[162] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard und D. McClosky, „The stanford CoreNLP natural language processing toolkit“, in *ACL (System Demonstrations)*, 2014, S. 55–60.

[163] M. Vacca, G. Turvani, F. Riente, M. Graziano, D. Demarchi und G. Piccinini, „TAMTAMS: An open tool to understand nanoelectronics“, in *Nanotechnology (IEEE-NANO), 2012 12th IEEE Conference on*, IEEE, 2012, S. 1–5.

[164] T. Hogenmüller und H. Zinner, *Tutorial for Lifetime Requirements and Physical Testing of Automotive Electronic Control Units (ECUs)*, Juli 2012.

[165] A. Kusel, J. Schönböck, M. Wimmer, G. Kappel, W. Retschitzegger und W. Schwinger, „Reuse in model-to-model transformation languages: are we there yet?“, *Software & Systems Modeling*, Jg. 14, Nr. 2, S. 537–572, 2015.

[166] L. A. Rahim und J. Whittle, „A survey of approaches for verifying model transformations“, *Software & Systems Modeling*, Jg. 14, Nr. 2, S. 1003–1028, 2015.

[167] D. Wagelaar, R. Van Der Straeten und D. Deridder, „Module superimposition: a composition technique for rule-based model transformation languages“, *Software & Systems Modeling*, Jg. 9, Nr. 3, S. 285–309, 2010.

[168] L. Abele, C. Legat, S. Grimm und A. W. Müller, „Ontology-based validation of plant models“, in *Industrial Informatics (INDIN), 2013 11th IEEE International Conference on*, IEEE, 2013, S. 236–241.

[169] D. Oberle, „How ontologies benefit enterprise applications“, *Semantic Web*, Jg. 5, Nr. 6, S. 473–491, 2014.

[170] J. Henss, J. Kleb, S. Grimm und I. Fraunhofer, „A Protégé 4 backend for native owl persistence“, in *11th Intl. Protégé Conference*, Citeseer, 2009, S. 1–4.

[171] J. Henss, J. Kleb, S. Grimm und J. Bock, „A Database Backend for OWL“, in *OWLED*, Bd. 529, 2009.

[172] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne u. a., „OWL-S: Semantic markup for web services“, *W3C member submission*, Jg. 22, Nr. 4, 2004.

[173] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne u. a., „OWL-S: Semantic markup for web services“, *W3C member submission*, Jg. 22, Nr. 4, 2004.

[174] A. M. Davis, *Software requirements: analysis and specification*. Prentice Hall Press, 1990.

# Tabellenverzeichnis

# Abbildungsverzeichnis

# Quelltexteverzeichnis

# Abkürzungsverzeichnis

| | |
|---|---|
| A | Automotive |
| ABox | Assertionaler Formalismus |
| ADAS | Advanced Driver Assistance Systems |
| AEC | Automotive Electronics Council |
| AÜG | Abhängigkeits-Übereinstimmungs-Graph |
| API | Application Programming Interface |
| ATL | Atlas Transformation Language |
| BEOL | Back-end-of-line |
| CAPE | Computer-Aided Production Engineering |
| CIA | Change Impact Analysis |
| CMOS | Complementary Metal-Oxide-Semiconductor |
| CTEF | Collaborative Technology Evaluation Framework |
| DL | Description Logic |
| DfA | Design for Automotive |
| DSL | Domain-specific Language |
| DSM | Domain-Specific Modelling |
| DSP | Direct Semantic Programming |
| DMG | Dependency Matching Graph |
| EAES | Entwurf und Architektur eingebetteter Systeme |
| ECU | Electronic Control Unit |

| | |
|---|---|
| EDA | Electronic Design Automation |
| E/E | Electrical/Electronic |
| EM | Electromigration |
| EMF | Eclipse Modeling Framework |
| EMI | Electromagnetic Interference |
| ES | Expertensystem |
| ETL | Epsilon Transformation Language |
| FAS | Fahrerassistenzsystem |
| FEOL | Front-end-of-line |
| FOAF | Friend of a Friend |
| GUI | Graphical User Interface |
| GUID | Globally Unique Identifier |
| HCI | Hot-Carrier Injection |
| HTML | Hypertext Markup Language |
| HTTP | Hypertext Transfer Protocol |
| IC | Integrated Circuit |
| IDE | Integrated Development Environment |
| INCOSE | International Council on Systems Engineering |
| IP | Intellectual Property |
| ISP | Indirect Semantic Programming |
| JRE | Java Runtime Environment |
| JSON | JavaScript Object Notation |
| KMU | Kleine und mittlere Unternehmen |
| LD | Linked Data |
| LOD | Linked Open Data |

| | |
|---|---|
| M2M | Model-to-Model |
| M2T | Model-to-Text |
| MB | Maschinenbau |
| MBSE | Modell-based Systems Engineering |
| MDA | Model-Driven Archicture |
| MDD | Model-Driven Development |
| MDE | Model-Driven Engineering |
| MDSD | Model-Driven Software Development |
| MDSE | Model-Driven Software Engineering |
| MMT | Model-to-Model Transformation |
| MOF | Meta Object Facility |
| MP | Mission Profile |
| MPO | Mission Profile Ontologie |
| MT | Medizintechnik |
| MVC | Model-View-Controller |
| MOF | Meta Object Facility |
| MPAD | Mission Profile Aware Design |
| MPF | Mission Profile Framework |
| MPFO | Mission Profile Format |
| ODD | Ontology-driven Development |
| OED | Ontology-enabled Development |
| OBA | Ontology-based Architectures |
| OEA | Ontology-enabled Architectures |
| OEM | Original Equipment Manufacturer |
| OMG | Object Management Group |

| | |
|---|---|
| OOP | Objektorientierte Programmierung |
| OWA | Open World Assumption |
| OWL | Web Ontology Language |
| OWL-S | Web Ontology Language for Web Services |
| PKW | Personenkraftwagen |
| QVT | Query/View/Transformation |
| RBox | Rollen Axiome |
| RDF | Resource Description Framework |
| RDFS | Resource Description Framework Schema |
| ReqIF | Requirements Interchange Format |
| REST | Representational State Transfer |
| RIF | Robustness Indicator Figure |
| RV | Robustness Validation |
| SE | Systems Engineering |
| SGML | Standard Generalized Markup Language |
| SiM | Systementwurf in der Mikroelektronik |
| SKOS | Simple Knowledge Organization System |
| SOAP | Simple Object Access Protocol |
| SQL | Structured Query Language |
| SWRL | Semantic Web Rule Language |
| SWT | Semantic Web Technologies |
| SPARQL | SPARQL Protocol and RDF Query Language |
| T2M | Text-to-Model |
| TBox | Terminologischer Formalismus |
| TDDB | Time-Dependent Dielectric Breakdown |

| | |
|---|---|
| Turtle | Terse RDF Triple Language |
| UML | Unified Modeling Language |
| URI | Uniform Resource Identifier |
| URL | Uniform Resource Locator |
| VOWL | Visual Notation for OWL Ontologies |
| W3C | World Wide Web Consortium |
| WSDL | Web Services Description Language |
| WWW | World Wide Web |
| XML | Extensible Markup Language |
| XSD | XML Schema Definition |
| XSL | Extensible Stylesheet Language |
| XSLT | XSL Transformation |