DHBW
Duale Hochschule
Baden-Württemberg
Mannheim

DLR

Entwicklung einer wissensbasierten Methodik zur Prozessplanung der Vormontage von Flugzeugkabinenmodulen

Bachelorarbeit

für die Prüfung zum

Bachelor of Science

des Studiengangs Informationstechnik

der Dualen Hochschule Baden-Württemberg Mannheim

von

Pascal Botzum

Abgabe am 29. August 2023

Bearbeitungszeitraum:	6. Juni 2023 — 29. August 2023
Matrikelnummer, Kurs:	3169663, TINF20IT1
Ausbildungsbetrieb:	Deutsches Zentrum für Luft- und Raumfahrt e.V.
Betreuer des Ausbildungsbetriebs:	Prof. Dr. Jörn Biedermann
Gutachter der Dualen Hochschule:	Prof. Dr. Holger Gerhards

Erklärung

Ich versichere hiermit, dass ich meine Bachelorarbeit mit dem

THEMA

Entwicklung einer wissensbasierten Methodik zur Prozessplanung der Vormontage von Flugzeugkabinenmodulen

selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.*

* falls beide Fassungen gefordert sind

P. Bah

Mannheim, den 29. August 2023

Vermerk zu den Abbildungen

Die verwendete Terminologie ergibt sich zum größten Teil aus englischen Begriffen. In dieser Arbeit ist aufgrund von einheitlichen Strukturen zwischen anderen Literaturen in deutscher Sprache mit englischen Fachbegriffen aufgebaut. Sofern möglich, werden diese Begriffe ins Deutsche überführt. Trotzdem wird in den Abbildungen ausschließlich die englische Sprache verwendet, um Konzepte von bereits bestehendem Vokabular anwenden zu können. Des Weiteren wird dadurch ein linguistisches Vermischen vermieden.

Zusammenfassung

Es werden moderne Methoden für die Produktionsplanung von Flugzeugen benötigt. Erst dadurch können Flugzeugmodelle nach Belieben des Kunden effektiv individualisiert und im Zuge der Nachhaltigkeit neue Flugzeugmodelle in die Produktion integriert werden. Das Deutsche Zentrum für Luft- und Raumfahrt (DLR) forscht an Methoden, um dies zu erreichen. Dabei wird an dem digitalen Faden geforscht, der die unterschiedlichen zur Produktion benötigten Schritte miteinander Ende-zu-Ende verbindet. Wissensbasiertes Engineering (KBE) und das semantische Web als Teil von KBE benutzen methodisch wissensbasierte Daten, um ein Verständnis über die Produktionsschritte aufzubauen und diese miteinander zu verbinden. Diese Arbeit beschäftigt sich mit der Forschungsfrage, inwiefern Ontologien und wissensbasierte Methoden aus dem KBE in der Luftfahrt genutzt werden können, um Montageprozesse generieren zu können. Eine Ontologie ist eine Spezifikation einer Konzeptualisierung. Dazu wurde eine Methodik zur Wissenssammlung und -Organisation mittels einer Ontologie aufgestellt. Diese Ontologie wurde auf Basis anderer Ontologien aus der Literatur erstellt, da für diesen Anwendungsbereich benötigte Kriterien in den anderen Ontologien nicht ausreichend vertreten sind. Zwei Anwendungsfälle wurden daraufhin erzeugt, die die verschiedenen Kriterien nachweisen und die Fähigkeiten der aufgestellten Methodik zeigen. Anhand dieser Anwendungsfälle lässt sich dann zeigen, dass Montageprozesse aus der aufgestellten Ontologie mittels Inferenz generiert werden und dass das Expertenwissen genutzt werden kann, um Varianten zu finden und redundante Prozesse auszuschließen. Diese Methodik zeigt das Potenzial, mit anderen Teilen des digitalen Fadens verbunden zu werden.

Abstract

Modern methods are needed for aircraft production planning. Only then can aircraft models be effectively individualized to the customer's liking and new aircraft models integrated into production in the interest of sustainability. The German Aerospace Center (DLR) is researching methods to achieve this. Research is being performed on the digital thread. It connects the various steps required for production with each other end-to-end. Knowledge-based engineering (KBE) and the semantic web as part of KBE are methodically using knowledge-based data to build an understanding about the production steps and connect them together. This thesis addresses the research question in how far ontologies and knowledge-based methods from KBE can be used in aviation to be able to generate assembly processes. For this purpose, a methodology for knowledge acquisition and knowledge organization using an ontology was established. This ontology was created based on other ontologies from the literature, since criteria needed for this application domain are not sufficiently represented in the other ontologies. Two use cases were then generated that demonstrate the different criteria and show the capabilities of the established methodology. Based on these use cases, it can then be shown that assembly processes are generated from the presented ontology by means of inference and that the expert knowledge can be used to find variants and to exclude redundant processes. This methodology shows the potential to be applied in other parts of aircraft production.

Inhaltsverzeichnis

Abbildungsverzeichnis VIII

Tabellenverzeichnis IX

Abkürzungsverzeichnis X

1 Einleitung 1
- 1.1 Ziel der Arbeit . . . 4
- 1.2 Gliederung der Arbeit . . . 5

2 Technische Grundlagen 6
- 2.1 Knowledge-Based-Engineering . . . 6
  - 2.1.1 Datenverarbeitung mithilfe von KBE . . . 8
  - 2.1.2 Konzepte der KBE-Methodik . . . 8
- 2.2 Semantic Web und Ontologien . . . 10
  - 2.2.1 Semantic-Web-Architektur . . . 12
  - 2.2.2 Darstellungsarten im Bereich Semantic Web . . . 14
  - 2.2.3 Wissensablagen mit Ontologien . . . 17
- 2.3 Grundlagen der Montageprozessplanung . . . 20

3 Anforderungsdefinition 22
- 3.1 Aufstellen der Kriterien zum Vergleich verschiedener Ontologien . . . 22
- 3.2 Vorstellen der einzelnen Ontologien . . . 25
- 3.3 Vergleich unterschiedlicher Ontologien aus der Literatur . . . 28

4 Vorstellung der Methodik 31
- 4.1 Methodik-Aufbau . . . 31
- 4.2 Ontologie-Aufbau . . . 33
  - 4.2.1 Aufbau der Meta-Ontologie . . . 33
  - 4.2.2 Analyse der Kriterien für die ideale Ontologie . . . 35
- 4.3 Implementierung der Methodik . . . 36

5 Validierung der Methodik anhand zweier Anwendungsbeispiele 40
5.1 Beschreibung des ersten Anwendungsfalles 41
5.2 Beschreibung des zweiten Anwendungsfalles 42
5.3 Aufbau der Wissensbasis 42
5.3.1 Wissensbasis des ersten Anwendungsfalles 42
5.3.2 Wissensbasis des zweiten Anwendungsfalles 44

6 Ergebnisse und Diskussion 48
6.1 Prozessgenerierung in Tabellen 48
6.2 Evaluation nach Hildebrandt 49
6.2.1 Skalierbarkeit der Wissensbasis 49
6.2.2 Erweiterbarkeit der Meta-Ontologie 51
6.2.3 Multiperspektivität durch die Erweiterbarkeit 53
6.2.4 Inferenzmöglichkeit des Anwendungsfalles 53
6.2.5 Inferenz zum Filtern von Prozessen 54
6.2.6 Validierung einzelner Wissensobjekte mittels Inferenz 57
6.2.7 N3 als Darstellungsformat und Prozessgenerierung 58
6.3 Diskussion 60

7 Zusammenfassung und Ausblick 64

Literatur 67

Abbildungsverzeichnis

2.1 Architektur-Pyramide zum Aufbau des Semantic Web nach Berners-Lee [14] 14
2.2 Beispielhaftes Modell einer Wissensbasis für Materialaufkommen verschiedener Bauteile 15
2.3 Beispielhafte Wissensbasis mit Computer-lesbaren Darstellungsformaten 16
2.4 Beispielhafter Aufbau der Wissensbasis mit Menschen-lesbaren Darstellungsformaten 17
2.5 Aufbau einer Wissensbasis als vergleichbare Definition zum MDA . . 19

4.1 Flussdiagramm zur Methodik zum Aufbau der Wissensbasis 32
4.2 Meta-Ontologie auf Basis des PPR-Modells 35
4.3 Inferenz mehrerer Dokumente mit der EYE Inferenz-Maschine 38

5.1 Wechselwirkungen einzelner Instanzen des Anwendungsfalles 43
5.2 Visualisierung der grundlegenden RDF-Tripel 45
5.3 Erweiterung der Meta-Ontologie auf Multi-KPI-Unterstützung 45
5.4 Wechselwirkungen einzelner Instanzen des zweiten Anwendungsfalles 46
5.5 Subklassenaufbau der Produkte und der Ressourcen innerhalb der Wissensbasis der Instanzen 47

6.1 Prozessgenerierung mit Varianten-Bildung 49
6.2 Prozessgenerierung des zweiten Anwendungsfalles 50
6.3 RDF-Visualisierung der erweiterten Ontologie 52
6.4 Anwendung einer Inferenzregel zum Inferieren der Definitions- und Zielmenge innerhalb der Ontologie 54
6.5 Prozessgenerierung des zweiten Anwendungsfalles mit dem angewendeten Filter 56
6.6 Validierung der Zielmenge 59

Tabellenverzeichnis

3.1 Relevanz-Liste der aufgestellten Kategorien im Kontext der vorgestellten Methode 25
3.2 Vergleich verschiedener Ontologien aus der Literatur 28

4.1 Liste der benutzten Technologien für die Umsetzung der Methodik . . 36

6.1 Liste der angewendeten Kriterien an den Anwendungsfällen 51
6.2 Bewertung der eigen erstellten Ontologie anhand der vorherigen Kriterien aus Kapitel 3 60

Abkürzungsverzeichnis

AI	Künstliche Intelligenz (artificial intelligence)
AOD	UND/ODER-Digraphen (AND/OR-Digraphs)
API	Applikationsprogrammier-Schnittstelle (Application Programming Interface)
CAD	Computergestütztes Design (Computer-Aided Design)
CAO	Kabinenmontage-Ontologie (Cabin assemly ontology)
DELS	Discrete Event Logistics Systems
DLR	Das Deutsche Zentrum für Luft- und Raumfahrt
DSL	Domänenspezifische Sprache (domain-specific language)
EYE	Euler Yet another proof Engine
FOAF	Freund-eines-Freundes-Ontologie (friend of a friend ontology)
HDT	Header Dictionary Triples
HDTQ	Header Dictionary Triples Quads
JSON	JavaScript Objekt-Notation
KB	Wissensbasis (knowledge base)
KBE	Wissensbasiertes Engineering (knowledge-based engineering)
KBS	Wissensbasierte Systeme (knowledge-based systems)
KPI	Performanz-Indikator (key performance indicator)
MAMOT	Modellierungswerkzeug für Fertigung und Montage (Manufacturing and Assembly Modeling Tool)
MASON	Semantik-Ontologie für die industrielle Herstellung (MAnufacturing's Semantics ONtology)
MDA	Rahmen für Modelgetriebene Architektur (Model-Driven-Architecture-Framework)
N3	Notation-3
NR	Namensraum (namespace)
OOP	Objekt-Orientierte Programmierung

OWL	Web-Ontologiensprache (Web Ontology Language)
PPR	Prozesse, Produkte und Ressourcen
PPR-DSL	Domänenspezifische Sprache für Prozess-,Produkt- und Ressourcenmodellierung (Domain-Specific Language for Product-Process-Resource Modeling)
RDF	Ressourcen-Deskriptionsrahmen (Resource Description Framework)
RDFS	Ressourcen-Deskriptionsrahmenschema (Resource Description Framework Schema)
SW	Das semantische Netz (Semantic Web)
SWAP	Die semantische Webapplikations-Plattform (The Semantic Web Application Platform)
SWRL	Regelsprache des Semantischen Webs (Semantic Web Rule Language)
SWT	Technologien des semantischen Netzes (Semantic Web Technologies)
TOVE	Toronto Virtual Enterprise
URI	Bezeichnung für einheitliche Ressourcen (Uniform Resource Identifier)
XML	Erweiterbare Auszeichnungssprache (eXtensible Markup Language)

1 Einleitung

Jede Airline möchte ihre Flugzeuge nach deren Belieben ausgestattet haben. Trotz dessen sind verschiedene Airlines mit häufig gleichen Flugzeugmodellen ausgestattet. Varianten-Fertigung ist in dieser Generation der Luftfahrtindustrie an Priorität gestiegen [1]. Das sorgt dafür, dass Flugzeuge rekonfiguriert werden müssen, damit beispielsweise Kabinenlayout, einzelne Kabinenelemente, Licht oder Unterhaltungsmöglichkeiten für die Individualwünsche der Endkunden angepasst werden [2]. Rekonfiguration heißt hierbei verändern und neu planen von Flugzeugmodellen. Ein Neuplanen von einem Flugzeugdesign sorgt für eine Anpassung von Produktionsschritten in anderen Teilen der Flugzeugproduktion, wie beispielsweise der Prozessplanung, den Simulationen und der Montage. Die verschiedenen Teile der Produktion sind voneinander abhängig, aber Wechselwirkungen untereinander können aufgrund der Komplexität nicht immer betrachtet werden [3].

Laut Analysen von La Rocca [4] befindet sich die Luftfahrtindustrie am Anfang einer Neugestaltung der Luftfahrt durch neue Innovationen und wissenschaftliche Durchbrüche. Durch neue und unkonventionelle Flugzeugmodelle, die schneller, leiser und energieeffizienter sind, wird die zukünftige Luftfahrt beschleunigt und kosteneffizienter. In der steigenden Nachfrage nach diesen neuen Flugzeugmodellen reichen herkömmliche Methoden zum Design und zur Produktion neuer Modelle nicht aus. Auf diese Veränderungen muss sich die ganze Luftfahrtindustrie, so auch die Flugzeugproduktion, anpassen. Es werden neue Methoden gefordert, mit denen alle, die an der Produktion von Flugzeugen beteiligt sind, auf die zukünftige Änderungen und Innovationen der Flugzeugindustrie reagieren können. [4]

Die Forderung nach neuen Methoden zur Flugzeugfertigung ist darauf zurückzuführen, dass mit bisherigen Methoden Rekonfigurationen im Flugzeugdesign erschwert

restliche Produktionsschritte betrachtet werden. Traditionell wird in der Designphase für ein neues Flugzeugmodell ein Design erstellt, wonach ein Produktionsplan erstellt und das kreierte Modell analysiert wird. In einem iterativen Prozess werden dann Modelle neu designt und separat reevaluiert [5]. Diese Schritte sind dabei voneinander getrennt. Wechselwirkungen zwischen den einzelnen Produktionsschritten sind daher eingeschränkt zu betrachten. Die einzelnen Schritte, die für neue Designs von Flugzeugen benötigt werden, sind in ihren Expertisen sehr heterogen und mit logistisch hohem Aufwand verbunden. Die einzelnen Schritte von der ursprünglichen Anforderungsdefinition bis hin zur Produktion des neu erstellten Flugzeugsdesigns kann eine beachtliche Zeit beanspruchen.

Aufgrund der hohen Komplexität eines Flugzeugs werden einige Teile von Flugzeugen schon jetzt in Kooperation mit mehreren Unternehmen an unterschiedlichen Standorten erstellt und zusammengestellt. Dies ist auf die steigende Nachfrage nach Nachhaltigkeit und dem Erstellen von revolutionären Designs zurückzuführen. Dieses sogenannte Outsourcing ist für sehr große Projekte wie die Flugzeugproduktion von Nutzen. Dennoch ist bei so einer globalen Organisation von Produktionsschritten die Verwaltung von Wissenselementen fehleranfällig. Bei komplexen Projekten mit logistischem Aufwand kann ein Verlust von Informationen auftreten. Des Weiteren kann eine Orchestrierung von Prozessen über den ganzen Globus verteilt zu einer echten Herausforderung führen. [4]

Die daraus ergebene Problematik ist ein Verlust von relevanten Informationen. La Rocca [4] beschreibt dies mit dem Konzept des “Brain Drain”: Die direkte Konsequenz an einer globalen Organisation und dem Auslagern der Produktion einiger Bauteile innerhalb eines Flugzeuges ist der Verlust von Wissen. Der englische Begriff steht metaphorisch für Wissen, welches aus einem Gehirn (*englisch:* brain) ausläuft (*englisch:* drain).

> “The first is that, while the risk of losing knowledge when an experienced employee leaves the company is rather evident, the risk associated to professionals’ mobility (within the same company) are often underestimated, although similar.” (La Rocca [4])

Die Aussage hinter diesem Zitat ist, dass Mitarbeiter mit Expertise ein Unternehmen nach einiger Zeit aus unterschiedlichen Gründen verlassen oder das Unternehmen deren Standort wechselt. Dadurch wird Wissen und Expertise verloren. La Rocca schließt daraufhin, dass wissensbasierte Methoden benötigt werden, mit denen Wissen auf längere Zeit erhalten bleibt. Eine dieser Methoden ist Wissensbasiertes Engineering (*englisch*: knowledge-based engineering, KBE). KBE hilft dabei, die unterschiedlichen Schritte in der Flugzeugproduktion miteinander zu kombinieren, sodass schon in der Designphase der Flugzeugmodelle Schlüsse über die Produktionsplanung in die Designphase integriert werden können. Mithilfe der Methodik von KBE werden Wissensobjekte organisiert. Damit können beispielsweise Rekonfigurationen in der Produktionsplanung dynamisch behandelt werden. Das Das semantische Netz (*englisch*: Semantic Web, SW) bietet Techniken, mit denen Wissensobjekte im ganzen Internet ausgetauscht werden können.

Das Deutsche Zentrum für Luft- und Raumfahrt (DLR) hat es sich zur Aufgabe gesetzt, Methoden zu erforschen, mit denen die Luftfahrtindustrie auf die modernen Anforderungen, aufgrund der steigenden Nachfrage zur Nachhaltigkeit, reagieren kann. Eine Vision für die Modernisierung der Luftfahrtindustrie ist der digitale Faden. Der digitale Faden ist eine Methodik, die in der Informationstechnik benutzt wird. Sie beschreibt das Aufstellen von homogenen Datenmodellen und Schnittstellen, mit denen dann verschiedene Stakeholder miteinander kommunizieren können. Das DLR plant das Integrieren des digitalen Fadens in die Flugzeugfertigung [6]. Dafür werden die für das Erstellen der Flugzeugkabine benötigten Schritte Ende-zu-Ende virtuell verbunden, das heißt sie werden miteinander homogen über einheitliche Datenmodelle und Schnittstellen verbunden [7] [2]. Dadurch können während einer Rekonfiguration im Design schon Folgen für beispielsweise der Montageprozesse mit berücksichtigt werden. So können vorzeitige Analysen multidisziplinärer Bereiche durchgeführt werden und neue Wechselwirkungen zwischen den einzelnen Schritten betrachtet werden. Die Flugzeugkabinenproduktion wird dadurch auf eine Massenproduktion mit individuellen Varianten vorbereitet. Mithilfe von einheitlichen virtuellen Modellen wird des Weiteren der Brain Drain minimiert, sodass innovative Designs im Zuge der Nachhaltigkeit, wie beispielsweise die Anwendung von Wasserstoffantrieben, effektiv erforscht und real umgesetzt werden können.

KBE ist eine der Methoden, die bei der Umsetzung des digitalen Fadens helfen. Aufgrund der Möglichkeit, mittels wissensbasierten Daten ein tieferes Verständnis für das Erstellen von Flugzeugen zu erlangen, wird dadurch die Möglichkeit geboten, multidisziplinär zu arbeiten und einzelne Produktionsschritte beim Erstellen eines Flugzeuges zu verbinden. Des Weiteren helfen Technologien des semantischen Netzes (*englisch*: Semantic Web Technologies, SWT), die KBE-Methodik umzusetzen, indem Technologien genutzt werden, welche standardisiert sind und im Internet genutzt werden können.

1.1 Ziel der Arbeit

Diese Arbeit beschäftigt sich hauptsächlich mit der Fragestellung, inwiefern Ontologien und wissensbasierte Methoden in der Luftfahrt genutzt werden können, um Montageprozesse zu generieren. Ferner wird untersucht, wie weit Ontologien benutzt werden, um Prozesse zu generieren und zu planen und inwiefern SWT eingesetzt werden können.

Durch diese Fragestellung implementiert diese Arbeit einen Teil des digitalen Fadens. Sie beschäftigt sich mit dem Erstellen von Schnittstellen zwischen Flugzeugdesign und Montageprozessplanung. Dazu wird in dieser Arbeit eine Methode entwickelt, in der Prozesse in der Vormontage von Flugzeugkabinenmodulen mittels Ontologien und Inferenz geplant werden können. KBE wird angewendet, um wissensbasiert zu arbeiten und Wissen abzubilden. Mit SWT werden standardisierte Technologien genutzt, die auch in anderen wissenschaftlichen Bereichen verwendet werden. Damit wird eine Informationsbasis erstellt, die in Form einer Ontologie abgespeichert wird. Mithilfe von Inferenz wird dann mit dem aufgestellten Kontext Lücken in der Informationsbasis geschlossen, sodass mit diesen Daten gearbeitet werden kann. Daraus lassen sich dann auch Prozesse generieren.

1.2 Gliederung der Arbeit

In Kapitel 2 werden die technischen Grundlagen erklärt, die für die Arbeit benötigt werden. Dazu wird erläutert, was KBE ist und wie Wissen mithilfe von Ontologien und anderen SWT dargestellt und behandelt werden kann. Daraufhin wird die Produktionsplanung erläutert und in Kontext zu KBE gestellt. Kapitel 3 definiert die Anforderungen, die an eine ideale Ontologie gestellt werden und vergleicht verschiedene Ontologien aus der Literatur. Die Ergebnisse des Vergleiches werden benutzt, um später eine eigene Ontologie zu definieren. In Kapitel 4 wird die Methodik vorgestellt und die Implementation gezeigt. Daraufhin werden in Kapitel 5 Anwendungsfälle beschrieben und anhand der aufgestellten Methodik ausgeführt. In Kapitel 6 werden dann die Ergebnisse aus der Anwendung mit den Anwendungsfällen zusammengefasst und diskutiert. Kapitel 7 fasst dann die Arbeit und deren Ergebnisse zusammen, stellt die Ergebnisse im Kontext der Motivation und baut einen Ausblick auf.

2 Technische Grundlagen

Diese Arbeit benutzt zur Umsetzung der Methodik KBE, um eine Wissensbasis aufzubauen, mit der Prozessplanung in der Flugzeugfertigung betrieben werden kann. KBE ist die Methode, um Wissen zu akkumulieren, zu bearbeiten und daraus nutzbare Daten zu generieren. Es stellt sich die Frage, was als Wissen angesehen wird und wie man damit arbeiten kann. Des Weiteren ist das SW ein Teil des Internets. Es benutzt KBE-Methoden, um Wissen darzustellen. Mittels Wissensablagen auf Basis vom SW werden Prozesse geplant. In diesem Kapitel werden Konzepte und Begriffe im Bereich KBE, SW und Prozessplanung definiert und erläutert.

2.1 Knowledge-Based-Engineering

Laut La Rocca [4] ist KBE eine Technologie, die Wissensbasierte Systeme (*englisch*: knowledge-based systems, KBS) benutzt, um Ingenieurswissen abzufangen und wiederzuverwenden. Das führt zu einer Automation von nicht-kreativen, sich wiederholenden Design-Prozessen. Zwei der von La Rocca [4] aufgestellten Unterschiede von KBE zu KBS ist die Unterscheidung einer Regelbasis zu einer Informationsbasis und die zugehörigen Regeln. In einem KBS wird zwischen einer Informationsbasis und einer Regelbasis so unterschieden, dass sie als zwei getrennte Konzepte angesehen werden. Währenddessen sind Regelbasen und Informationsbasen ein gemeinsames Konzept in KBE. Daraus ergibt sich, dass eine Regelbasis und eine Informationsbasis das gleiche Konzept widerspiegeln werden. Des Weiteren wird durch KBE ein multidisziplinäres Optimieren von Prozessen in der Designphase unterstützt. La Rocca bezieht sich dabei sehr auf den Flugzeugentwurf. KBE findet Anwendung in

vielen Bereichen, wie beispielsweise in dem Flugzeugdesign [4], der Flugzeugproduktion [8] und des Brückenentwurfs [9]. Ziel von KBE ist das Erfassen von Techniken und Expertise, um eine Wissensbasis zu kreieren, um daraus nutzbare Informationen zu generieren. Dies geschieht, indem es auf Wissensbasen Informationen erschließen kann und auf neue Informationen anwenden kann [8].

Laut Chapman und Pinfold [8] ist die Methodik des KBE ein evolutionärer Schritt im Bereich der Computergestützten Entwicklung und Planung. Dies liegt daran, dass sie Technologien und Methoden der folgenden Bereiche kombiniert: Objekt-Orientierte Programmierung (OOP), Künstliche Intelligenz (*englisch*: artificial intelligence, AI) und Computergestütztes Design (*englisch*: Computer-Aided Design, CAD) [8]. KBE bietet die Möglichkeit, Modelle und Anforderungen von Bauteilen in digitaler Form, demnach CAD, abzuspeichern. Bauteile und andere Objekte werden zueinander in Relation gestellt, dies ist der Aspekt der OOP. Das geschieht genauer unter Definition von Hierarchien und Prototypen, welche bestimmte Eigenschaften besitzen und an instanziierte Objekte weitergeben. AI ergibt sich aus dem Fakt, dass das dargestellte Wissen erweitert werden kann, indem KBS deduktiv Lücken füllen und Fragen an der Wissensbasis beantworten. Mit AI ist hier kein maschinelles Lernen gemeint, bei dem neuronale Netze aufgebaut werden und neben anderen Methoden das “Deep Learning” betrieben wird. Es handelt sich um den Bereich der künstlichen Intelligenz, welches Expertensysteme aufstellen kann. Ein Expertensystem ist ein Computersystem, das in einen gewissen Kontext Fragen so beantwortet wie ein Experte in diesem Bereich [10]. So kann dieser Part der AI aufgrund vordefinierter Kontexte Fragen mit Expertise beantworten. Aufgrund dieser drei Aspekte, können KBE für Varianten-Bildung und dynamische Prozessbildung eingesetzt werden, was ein Umgestalten und Rekonfiguration jetziger Designs erlaubt. Da ein Expertenwissen innerhalb der Flugzeugproduktion aufgebaut wird, kann mittels der KBE-Methodik Wechselwirkungen betrachtet werden. Laut Chapman und Pinfold [8] benutzen Unternehmen wie Boeing und Textron Aerostuctures schon KBE. Diese Unternehmen waren dadurch in der Lage ihre Produktivität zu steigern.

2.1.1 Datenverarbeitung mithilfe von KBE

Wissensbasiertes Engineering beinhaltet also einen induktiven Prozess. Es wird dabei Wissen in ein Computer-lesbares Format übersetzt und in ein System integriert, mit dem darauf Schlussfolgerungen gezogen werden. Wissen wird akkumuliert, indem Expertenbefragungen durchgeführt werden, wissenschaftliche Texte, technische Dokumente integriert werden oder Verhaltensanalysen durchgeführt werden [11].

In einem Betrieb befinden sich meist Mitarbeiter*innen, die sich im Laufe derer Arbeit stark spezialisieren. Sollte also eine Person ausfallen oder wird ein neuer Mitarbeiter antrainiert, so muss das spezialisierte Wissen übertragen werden. Das Wissen, welches für die unterschiedlichen Produktionsprozesse von Flugzeugen benötigt wird, ist ebenfalls spezialisiert. Das Wissen der einzelnen Stakeholder muss zusammengeführt werden, erst durch diese Zusammenführung kann multidisziplinär gearbeitet werden. [5]

In KBE ist die Prämisse, dass spezialisiertes Wissen formalisiert werden kann. Dadurch können Datenstrukturen aufgebaut werden, mit denen man das spezialisierte Wissen abbildet. Dieses Wissen wird dann benutzt, um zwischen den verschiedenen Teilen der Flugzeugproduktion angewendet zu werden, bei dem auch Wechselwirkungen zwischen den Teilen berücksichtigt werden.

2.1.2 Konzepte der KBE-Methodik

KBE benutzt Wissen als zentrales Objekt, baut damit Wissensbasen auf und erstellt durch eine Inferenz-Maschine neue Wissensobjekte. **Wissen** stellt Konzepte, Gegenstände und Wechselwirkungen zwischen diesen dar. Kiritsis [11] unterscheidet dabei zwischen mehreren Arten von Wissen:

- Objekt-Wissen: Beschreibungen eines Objektes aus der realen Welt. In diesem Fall kann es sich um ein Bauteil sein oder ein Werkzeug handeln.
- Event-Wissen: Ereignisse, die passieren werden oder schon passiert sind. Das können unter anderem Prozesse sein, die in einer Montage benötigt werden.

- Performanz-Wissen: Wissen darüber, wie Wissen angewendet werden kann. Das können Design- und Produktionsschritte oder allgemein Anleitungen sein.
- Metawissen: Wissen über das zuvor erklärte Wissensobjekte. Das können sowohl Relationen zwischen einander sein, oder Wissen darüber, in welchen Kontext Wissen benutzt wird.

Das Erschließen von neuen Wissenselementen auf Basis von bestehenden Wissenselementen bezeichnet man als **Inferenz**. Indem in einem gewissen Kontext neue Wissenselemente deduziert werden, können Lücken in einer Wissensbasis geschlossen werden. Ein Programm, das automatisierte Inferenzen auf eine bestehende Wissensbasis ausführen kann, um so die Wissensbasis zu erweitern, bezeichnet man als Inferenz-Maschine.

Das Konzept des Erschließens von neuem Wissen ist ein Teil der Methodik des KBE und ein zentraler Aspekt des sogenannten semantischen Webs. Zu betonen ist, dass es nicht ausreicht, nur Objekt-Wissen aufzustellen. Mithilfe der anderen Wissensarten wird der Wissensbasis eine Bedeutung gegeben, mit der weitergearbeitet werden kann. Das SW nutzt Inferenz und Verarbeitung von wissensbasierten Daten aus.

Ein KBS besteht nach La Rocca [4] aus mehreren Komponenten: der Wissensbasis (*englisch*: knowledge base, KB), dem Arbeitsplatz, dem Erklärungs-Teilsystem und der Nutzerschnittstelle. Zum einen gibt es die **KB**. Eine KB speichert Expertisen-Wissen in einem dedizierten Container ab. Dieses Wissen ist nicht zwingend anwendungsspezifisch auf eine einzelne Problemstellung. Stattdessen ist verschiedenes Wissen in der ganzen Wissensbasis angehäuft. Die unterschiedlichen Arten von Wissen werden in einer Wissensbasis abgespeichert. Eine Wissensbasis speichert die unterschiedlichen Arten von Wissen zentral und persistent ab. Durch das Abrufen von bereits aufgestellten einzelnen Wissens-Aussagen, sogenannte Wissensobjekte, kann ein tiefes Verständnis aufgebaut werden. Erst durch ein tiefes Verständnis in einem bestimmten Bereich wird die KBE-Methodik wirksam. Denn mit Teilen der einen Wissenskategorie und einigem Wissen aus der anderen Wissensart, kann mithilfe der KBE-Methodik auf neues Wissen geschlossen werden. Durch eine KB wird beispielsweise beschrieben, dass bestimmte Bauteile als Objekt-wissen und Prozesse als Performanz-wissen vorliegen. Daraus kann geschlossen werden, ob der Prozess

mit diesen Bauteilen ausgeführt werden kann oder ob Bauteile dafür fehlen. Diese Informationen können dann später vom Endnutzer der KBE-Methodik verwendet werden.

Ein **Arbeitsplatz** oder "blackboard" definiert einen kurzzeitigen Speicher, der problemspezifische Daten abspeichert. Der Arbeitsplatz wird benutzt, um ein Problem mittels KBE zu lösen.

Ein **Erklärungs-Teilsystem** ist ein weiterer Teil des KBS, das dem Endnutzer eine Erklärungsbasis bietet. Dabei wird erklärt, wie spezifische Lösungen innerhalb des KBS gefunden werden.

Die **Nutzerschnittstelle** gibt dem Endnutzer eine Möglichkeit, dem KBS Wissen hinzuzufügen und Wissen abzufragen. Erst durch eine Nutzerschnittstelle ist es möglich, Lösungen zu einem Problem zu generieren.

2.2 Semantic Web und Ontologien

Das semantische Netz (*englisch*: Semantic Web, SW) wurde konzipiert, um das Internet für den Menschen zugänglicher zu machen [12]. Die Methodik des SW ist es, ein Vokabular zu definieren und eine Semantik zu den definierten Wörtern hinzuzufügen. Dadurch entstehen untereinander verbundene Daten. Dabei ist zu beachten, dass das SW nicht nur einzelne Wissensobjekte, sondern auch ganze Dokumente und Wissensablagen miteinander verbindet [13]. Semantik wird definiert, indem Wissensobjekte eindeutige Bedeutungen gegeben werden und in einem Zusammenhang zu anderen Wissensobjekten gestellt werden. Darüber hinaus stellt Semantik in diesem Kontext eine domänenspezifische Eindeutigkeit auf. So kann bspw. in der Informatik ein "Frame" vieles bedeuten, beispielsweise ein Internet-Datenpaket für die Kommunikation verschiedener Computer oder auch eine Position in einem Robotersysteme. Wenn mithilfe von SWT ein Frame in eine Wissensbasis definiert wird, gibt es verschiedene Möglichkeiten, einem Frame Semantik beizufügen. So wird beispielsweise eine eindeutige Definition verwendet und ein Frame kann dann nur in diesem Kontext benutzt werden. Oder es werden zwei Kontexte definiert, in dem ein Frame

auftreten kann. Mithilfe von Inferenz kann dann aus dem gegebenen Kontext die passende Definition bestimmt werden. Dadurch können KBS diesen Begriff in die dafür vorgesehene Kontexte benutzen.

Durch die Nutzung vom SW, wird eine Wissensbasis schematisch aufgebaut [14]. Es wird ein Vokabular daraus definiert, bei der der Wissensbasis semantische Informationen beigefügt werden. Einzelne Vokabulare definieren eigene Namensräume. Ein Namensraum (*englisch*: namespace, NR) besitzt ein Präfix, wie "log", das an alle Teile des Vokabulars angefügt wird, um es von anderen Vokabularen auseinander halten zu können. Dieses Präfix gehört meistens zu einer Dokumentadresse. Das kann eine lokale Datei als auch eine Internetadresse sein. "log", "math" und weitere sind alle Namensräume, die mit dem SWAP entstanden sind. Die semantische Webapplikations-Plattform (*englisch*: The Semantic Web Application Platform, SWAP) war ein Projekt, dass die Möglichkeiten der Semantic Web-Technologien gezeigt hat. Diese Namensräume werden auch heute noch benutzt von einigen Inferenz-Maschine wie cwm [15] oder EYE [16]. Ein eigenes Vokabular und NR zu definieren, reicht meist nicht aus, um von anderen Bereichen des SW benutzt zu werden. Dafür muss eine gewisse Architektur instandgehalten werden. Erst dadurch können Vokabulare untereinander ausgetauscht werden.

Es gibt unterschiedliche Ansichtsweisen, wie eine Ontologie aufgebaut ist und aus welchen Teilen diese bestehen kann. Eine dieser Perspektiven ist die Einteilung in leicht- und schwerwiegende Ontologien [17]. Eine leichtwiegende Ontologie ist eine Top-Level-Beschreibung einer Wissensbasis. Sie ist meist aufgestellt in intuitiv verständlichen und menschenlesbaren Formaten wie nicht standardisierte, UML- oder SysML-Diagrammen. Daraus lassen sich die Architektur, Wechselwirkungen und ähnliche grobe Eigenschaften einer Wissensbasis ablesen. Auf der anderen Seite befinden sich schwere Ontologien. Diese Ontologien sind reich an Daten, sind meistens schwerer zu verstehen und meistens sind sie in computerlesbaren Sprachen definiert. Eine schwerwiegende Ontologie kann Daten aus der Realität direkt abbilden und kann demnach effizienter als nur Visualisierungen von Computersystemen bearbeitet werden. [17]

Beide Seiten von Ontologie-Arten haben je nach Anwendungsfall ihre Existenzberechtigung, da es manchmal nur wichtig ist, eine grobe Methodik und Architektur zu definieren. Auf der anderen Seite ist es manchmal wichtiger im Anwendungsfall eine Ontologie direkt mit Daten zu füllen und programmatisch abzuarbeiten. Diese Arbeit befindet sich mittig, aber eher auf der Seite der schweren Ontologien. Obwohl in dieser Arbeit Diagramme für die Ontologie erstellt werden, ist die in der Methodik aufgestellte Meta-Ontologie mit technischen Daten zu füllen und programmatisch auszuführen.

2.2.1 Semantic-Web-Architektur

Technologien des semantischen Netzes werden im SW. Um die Vielzahl an Technologien zu sortieren, wurden sie in verschiedene Ebenen eingeteilt, die das ganze SW ausmachen. Die Ebenen sind in einer Pyramide von Berners-Lee [14] angeordnet und sind in Abbildung 2.1 aufgestellt. In der untersten Ebene werden nur Objekte betitelt. Dies geschieht mit einer Bezeichnung für einheitliche Ressourcen (*englisch*: Uniform Resource Identifier, URI). Das heißt, dass die Objekte nicht nur eine einzigartige Identifikation bekommen. Im Kontext des SW gibt die URI zusätzlich einen Standort in Form eines Internetdokumentes an, an dem sich das Wissensobjekt befindet. Für Ordnung wird in der zweiten Ebene gesorgt, indem jedes Objekt einem NR zugeordnet wird. Die Erweiterbare Auszeichnungssprache (*englisch*: eXtensible Markup Language, XML) wird von Berners-Lee hier genannt, damit ein NR definiert werden kann. Andere Darstellungsformate wie JavaScript Objekt-Notation (JSON) oder Header Dictionary Triples (HDT) können ebenfalls benutzt werden. Mit Notation-3 (N3) werden Objekte mit URIs benutzt und jeweils einem NR zugeordnet. Wie üblich zu allen Ressourcen-Deskriptionsrahmen (*englisch*: Resource Description Framework, RDF)-Formaten werden in N3 logische Beziehungen aufgestellt.

Die Daten bekommen in der dritten und vierten Ebene Semantik. Mit RDF werden logische Verknüpfungen erstellt und ein Vokabular definiert, mit dem eine Struktur gegeben wird [18]. Daraus ergibt sich eine Meta-Ontologie, die in ihrem eigenen NR

definiert ist. In RDF werden mehrere Vokabulare aufgestellt, wie das Ressourcen-Deskriptionsrahmenschema (*englisch*: Resource Description Framework Schema, RDFS) [19], welche eine gewisse Semantik definieren. Damit können angelehnt an der OOP Wissensobjekte hierarchisch aufgestellt werden. Diese Semantik wird mit sogenannten Entailments gestützt. Dieses Wort ist ähnlich zur Bedeutung des Wortes "Erfordernis" oder "Konsequenz". Entailments eines Vokabulars definieren logische Regeln, die erfüllt sein müssen, damit die mit dem Vokabular beschriebene Wissensbasis in dem Vokabular angesehen werden kann. Man könnte formulieren, dass innerhalb eines Vokabular diese Regeln erforderlich sind. Oder die Formulierung ist, dass Entailments Inferenzregeln implizieren, die auf eine Wissensbasis angewendet werden und konsequenterweise diese vervollständigen. Da es sich bei Entailments um beides handelt, wird der englische Begriff verwendet, und nicht die Wörter "Konsequenz" oder "Erfordernis".

Die fünfte Ebene, die Logikebene, führt Regeln ein, damit das aufgestellte Vokabular in richtigen Kontexten definiert wird. Diese Ebene benutzt ein Regelsystem und anhand dessen werden dann Daten inferiert. Die Beweis-Ebene (*englisch:* Proof) ergibt sich aus der Logikebene. Die Beweis-Ebene validiert die in der vorigen Ebene definierten Regeln. Mit jedem Regelsystem können Beweise validiert werden [14]. Mithilfe der Möglichkeit, Formeln in N3 darzustellen, können Regeln aufgestellt werden. Der Namensraum RDFS definiert bestimmte Regeln, die neues Wissen inferieren lassen und zusätzlich Aussagen treffen über die Validität des aufgestellten Graphen. Die Vertrauensebene bestimmt die Glaubhaftigkeit/Validität einzelner logischer Aussagen und Regeln. Wenn die Aussagen von einem Mitarbeiter kommen, kann sich herausstellen, dass diese Aussagen falsch oder veraltet sind.

In Abbildung 2.1 wird die digitale Signatur an der rechten Seite genannt. Sie umschließt das Aufbauen der Ontologie bis hin zur Beweis-Ebene. Dies hängt mit der Vertrauensebene zusammen. Dadurch, dass mit einer digitalen Signatur die Identität einer Person nachgewiesen werden kann, können Aussagen über die Vertrauenswürdigkeit einzelner Wissensobjekte getätigt werden. Wenn beispielsweise eine spezifische Aussage in einer Firma von einer mitarbeitenden Person digital getätigt wird, kann dadurch die Aussagekraft hinter dieser Aussage bestimmt werden. Wenn

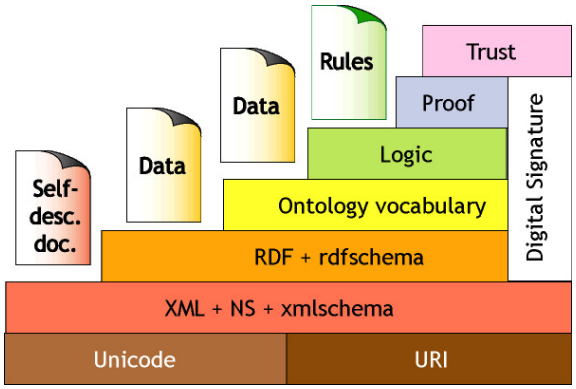


Abbildung 2.1: Architektur-Pyramide zum Aufbau des Semantic Web nach Berners-Lee [14]

der/die Mitarbeiter*in der selben Arbeitsgruppe angehört in der das Wissensobjekt erstellt wurde, ist die Vertrauenswürdigkeit höher, als wenn die Person einer anderen Arbeitsgruppe angehören würde. Diese Arbeit beschäftigt sich nicht mit dieser Ebene. Das ganze Prinzip dieser Pyramide in Abbildung 2.1 ist, dass die Ebenen aufeinander aufbauen. Im Nachfolgendem geht es um die Ebenen, die eine Ontologie, deren Logik und die Validierung aufbauen. Die unterste Ebene ist schon durch URIs gegeben und die oberste Ebene kann als Erweiterung zu dieser Arbeit betrachtet werden.

2.2.2 Darstellungsarten im Bereich Semantic Web

Für RDF gibt es verschiedene Darstellungsarten für verschiedene Anwendungsfälle. Es werden drei Arten von Darstellungsformaten definiert, mit denen im Bereich des SW Wissensbasen aufgestellt werden. Dies sind Computer-lesbare Formate, menschenlesbare Formate und Formate, die für die Massenspeicherung optimiert sind. Letzteres ist ebenfalls ein Computer-lesbares Format, die Massenspeicher-Formate werden aber nicht für die Inferenz benutzt. Grundsätzlich wird für jedes Format ein Graph beschrieben. Ein Graph bezeichnet dabei eine Ansammlung von logischen Aussagen, die miteinander verbunden sind und so einen gerichteten Graphen erstellen. Jede Aussage besteht aus drei logischen Teilen, dem Subjekt, dem Prädikat und

dem Objekt. Bei Subjekten und Objekten handelt es sich um Wissensobjekte, die mit Prädikaten in Beziehung zueinander gestellt werden.

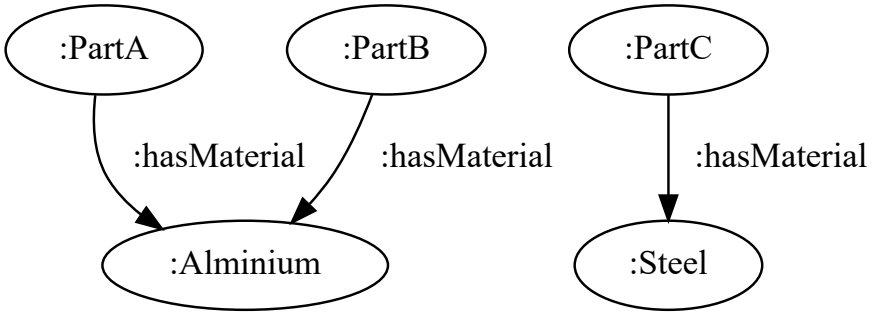


Abbildung 2.2: Beispielhaftes Modell einer Wissensbasis für Materialaufkommen verschiedener Bauteile

RDF-Graphen können visuell beschrieben werden. Abbildung 2.2 zeigt so eine Visualisierung. Mittels Visualisierungswerkzeuge, wie dem "dot"-Programm zum Grafikerstellen [20], können gerichtete Graphen erstellt werden. Dazu werden die Tripel aus den möglichen Darstellungsformaten für RDF-Tripel in das "DOT"-Dateiformat der Softwarebibliothek "graphviz" [21] übersetzt und an das "dot"-Programm übergeben. Innerhalb dieses Graphen werden logische Tripel wie folgt dargestellt: Subjekte und Objekte werden mit Knoten repräsentiert. Prädikate werden dann als Kanten zwischen den Knoten erstellt. Dabei sind die Kanten gerichtet. Der Startpunkt der Kante, der Teil ohne Pfeil, ist innerhalb eines Tripels ein Subjekt. Ein Objekt befindet sich dann auf der anderen Seite am Endpunkt der Kante, dem Teil mit einem Pfeil. In Abbildung 2.2 werden drei Aussagen getätigt: Bauteil A besteht aus Aluminium, genauso wie Bauteil B und Bauteil C besteht aus Stahl. Das Prädikat ":hasMaterial" verbindet dabei das Subjekt mit dem Objekt. Die Subjekte sind in diesem Kontext alle Bauteile und die Objekte sind Materialien.

Für die Computer-lesbaren Formate existieren XML- und JSON-basierte Formate [18]. XML und JavaScript Objekt-Notation sind im Internet stark akzeptierte Darstellungsformates für Daten. Dies ist darauf zurückzuführen, da JSON direkt aus der Programmiersprache JavaScript entstanden ist [22] und für die meisten Programmiersprachen Schnittstellen zur Interaktion mit JSON entwickelt wurden [23]. XML wird von Berners-Lee [14] benutzt als Darstellungsformat für RDF-Daten. In Abbil-

```json
[
  { "@id": "file://...#PartA",
    "file://...#hasMaterial": [{
        "@id": "file://...#Alminium"
      }]
  },
  { "@id": "file://...#PartB",
    "file://...#hasMaterial": [{
        "@id": "file://...#Alminium"
      }]
  },
  { "@id": "file://...#PartC",
    "file://...#hasMaterial": [{
        "@id": "file://...#Steel"
      }]
  }
]
```

(a) JSON-Darstellung

```xml
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF
  xmlns="file://...#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
>
  <rdf:Description rdf:about="file://...#PartA">
    <hasMaterial rdf:resource="file://...#Alminium"/>
  </rdf:Description>
  <rdf:Description rdf:about="file://...#PartC">
    <hasMaterial rdf:resource="file://...#Steel"/>
  </rdf:Description>
  <rdf:Description rdf:about="file://...#PartB">
    <hasMaterial rdf:resource="file://...#Alminium"/>
  </rdf:Description>
</rdf:RDF>
```

(b) XML-Darstellung

Abbildung 2.3: Beispielhafte Wissensbasis mit Computer-lesbaren Darstellungsformaten

dung 2.3 werden zwei Darstellungsarten für RDF-Graphen gezeigt. Abbildung 2.3a zeigt dabei den in Abbildung 2.2 dargestellten Graphen für das Darstellungsformat in JSON und Abbildung 2.3b zeigt die Darstellung in XML. Beide Darstellungsarten führen durch ihren Aufbau eine Hierarchie ein. So sind die Objekte mehr eingerückt, als die Subjekte.

Für die menschenlesbaren Formate existieren unter anderem Turtle und Notation-3. Alle Formate sind dadurch ausgezeichnet, dass einfache logische Aussagen, auch Tripel genannt, in einer simplen Syntax mit “S P O .” dargestellt werden. Dabei steht S, P und O jeweils für Subjekt, Prädikat und Objekt. Turtle bietet zusätzlich die Möglichkeit, mehrere Objekte oder mehrere Prädikate gleichzeitig zu benutzen, um mehrere Aussagen im gleichen Tripel darzustellen. Das, was N3 hervorhebt, ist die Komplexität. N3 unterstützt erweiterte Attribute, wie die Beschreibung von logischen Formeln, damit Regeln definiert werden können. In Abbildung 2.4 werden zwei Darstellungsarten für den RDF-Graphen aus Abbildung 2.2 gezeigt. Abbildung 2.4a zeigt dabei die Darstellung im Turtle-Format und Abbildung 2.4b zeigt die Darstellung in N3. Grundsätzlich sind beide Formate einfacher zu lesen. Dies liegt daran, dass Subjekt, Prädikat und Objekt expliziter aufgeführt wurden und der NR mit einem Präfix einmal definiert wurde. In Abbildung 2.4 sind zusätzlich ein Subgraf definiert, mit dem eigene Aussagen aufstellt wurden. Die Aussage ist, dass nachdem

```
1  @prefix : <file://...#> .
2
3  :PartA :hasMaterial :Alminium .
4  :PartB :hasMaterial :Alminium .
5  :PartC :hasMaterial :Steel .
6
7  _:Formula0 :after :Assembly .
8
```

(a) Darstelung mit dem TURTLE-Format

```
1  @prefix : <file://...#> .
2
3  :PartA :hasMaterial :Alminium .
4  :PartB :hasMaterial :Alminium .
5  :PartC :hasMaterial :Steel .
6
7  {
8      :PartA :wasUsed true .
9  } :after :Assembly .
```

(b) N3-Darstellung

Abbildung 2.4: Beispielhafter Aufbau der Wissensbasis mit Menschen-lesbaren Darstellungsformaten

die Montage abgeschlossen ist, Bauteil A benutzt wurde. Auffällig ist, dass N3 dies darstellen kann, während TURTLE und andere Formate keine Unterstützung dafür haben. In TURTLE wird dieser Subgraf als “Formel null” bezeichnet.

Für die Massenspeicherung eignen sich wiederum andere Formate. HDT ist ein binäres Datenformat. Mithilfe einer Lookup-Tabelle werden logische Tripel in eine Kopfzeile, der Tabelle und binären Tripeln enkodiert und so ein dynamisches Laden von massiven Datenmengen ermöglicht [24]. Header Dictionary Triples Quads (HDTQ) ist eine Erweiterung, mit der auch Subgrafen abgespeichert werden können. Dies geschieht über die zusätzliche Markierung von Graphen. Daraus ergeben sich nicht Tripel, sondern Quadrupel (Quads). Damit kann die Funktionalität der Beschreibung von Formeln nachgestellt werden [25]. Es gibt auch Datenbanksysteme, die in der Lage sind, RDF-Tripel darzustellen. Berkeley DB [26] und InfluxDB [27] sind dafür Beispiele.

2.2.3 Wissensablagen mit Ontologien

Um ein Vokabular zu erstellen, ist es gewöhnlich, eine Ontologie zu definieren. Der Begriff Ontologie stammt ursprünglich aus der Philosophie, bei der die Ontologie die Lehre der Existenz bedeutet. In der philosophischen Ontologie werden Existenz und Sein hinterfragt. Im Kontext der Informationstechnik wurde der Begriff über-

nommen und bezeichnet laut Gruber [28] eine explizite Spezifikation einer Konzeptualisierung. In AI- und Computersystemen ist die Frage, was existiert, durch eine Ontologie beschrieben. In einem definierten Vokabular werden Objekte und deren Beziehungen zueinander beschrieben, damit ein KBS mit wissensbasierten Daten arbeiten kann. Gruber [28] betont dabei, dass für einen Nutzer der Ontologie nicht alle Konzepte einer Domäne beschrieben werden müssen. Es reicht lediglich die für den Nutzer relevanten Informationen zu sehen, während die eigentliche Ontologie eine viel größere Datenmenge besitzen kann. Auch durch die Vision des SW von Berners Lee ist klargestellt worden, dass durch eine Verlinkung von verschiedenen Internetressourcen ein größeres Wissen geschaffen werden kann. Dies zeigt aber auch, dass spezifisches Wissen gefiltert werden kann, sodass der Endnutzer am Ende nur die relevanten Wissensobjekte sieht.

Für diese Arbeit sind zwei Bestandteile einer Ontologie am relevantesten: die Meta-Ontologie und die Informationsbasis. Die Meta-Ontologie ist verantwortlich für die Struktur. Hierbei werden Relationen und allgemeine Regeln definiert. Eine Meta-Ontologie ist vergleichbar zur OOP. Relationen und Regeln werden in der OOP mit Klassen und Schnittstellen definiert. Klassen definieren ein Schema, nach dem Daten strukturiert werden. Schnittstellen definieren Verhalten, das für mehrere Klassen implementiert wird [29]. Klassen definieren auch Unterklassen. Wenn ein Objekt nach dem Schema einer Unterklasse strukturiert wird, werden auch alle Attribute der darüberliegenden Klasse mit berücksichtigt. Dieses Konzept ist ähnlich zu der objektorientierten Programmiersprache Java mit dem ‘interface’-Schlüsselwort [30] oder einer abstrakten Klasse aus der C++-Programmiersprache [31]. Die Informationsbasis ist verantwortlich für den Inhalt. Dabei werden Attribute der realen Objekte und Modelle notiert und spezifische Regeln definiert. Die aufgestellten Objekte entsprechen das von Kiritsis [11] beschriebenen Objekt-Wissen. Regeln und Formeln können dem Performanz-Wissen oder Metawissen zugeordnet werden. Die Meta-Ontologie ist auch dem Metawissen zuzuordnen. Die formalisierten Daten folgen dann streng der Struktur, die in der Meta-Ontologie definiert ist. Auf Grundlage der Struktur können dann weitere Fakten inferiert werden, sodass ein tieferes Wissen gebildet werden kann. Die existierenden und dazugewonnenen Fakten werden dann gemeinsam benutzt, um beispielsweise Probleme der Prozessplanung zu lösen.

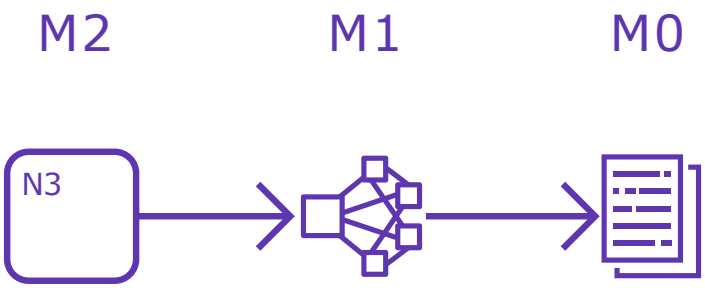


Abbildung 2.5: Aufbau einer Wissensbasis als vergleichbare Definition zum MDA

Dieser Aufbau ist vergleichbar zu der Definition des Rahmen für Modelgetriebene Architektur (*englisch*: Model-Driven-Architecture-Framework, MDA) von der OMG (Object Management Group) [32]. Um Modelle zu beschreiben und anzuwenden, werden verschiedene Ebenen definiert. Die M0-Ebene definiert ein spezifisches Modell mit all ihren Daten. Dies ist vergleichbar zur Informationsbasis, die hier vorgestellt wurde. Die Meta-Ontologie ist vergleichbar zur M1-Ebene des MDA. Denn hier werden Konzepte und Modelle aufgestellt und wie sie miteinander wirken. Die M2-Ebene ist als eine noch höhere Konzeptions-Ebene definiert. Diese Ebene ist eine Beschreibung der Beschreibung von Modellen. Dies ist vergleichbar zur Nutzung des RDF mit der N3-Syntax. Dieser Aufbau ist auch in Abbildung 2.5 gezeigt.

Für den MDA wird ein Modell definiert als eine Vereinfachung eines Systems. Eine Software, dass dieses Modell benutzt, ist in der Lage, Expertenfragen zu beantworten. Es repräsentiert dabei das zugrundeliegende System. Ein Metamodell (eine Meta-Ontologie) spezifiziert die Abstraktion des Systems [32]. Das Metamodell definiert ein Vokabular und Annahmen. Erst durch diese Definition kann ein Modell kreiert werden.

Im SW existieren Meta-Ontologien, die zum Aufbau eigener Ontologien gedacht sind. RDFS ist eines dieser Meta-Ontologien und ist parallel zu RDF vom World Wide Web Consortium definiert. RDFS definiert das Konzept einer Subklasse und einer Subeigenschaft (*englisch:* subproperty). Instanzen von einer Subklasse sind gleichzeitig eine Instanz der Grundklasse. Selbiges Prinzip existiert auch für Subeigenschaften. Die Web-Ontologiensprache (*englisch*: Web Ontology Language, OWL)

bezeichnet eine Meta-Ontologie, die strikter der OOP angelehnt ist [33]. Dabei unterscheidet OWL zwischen verschiedenen Arten von Eigenschaften, Klassen und weiteres, sodass genauere Beschreibungen über die zu erstellende Wissensbasis erzeugt werden kann. Die Freund-eines-Freundes-Ontologie (*englisch*: friend of a friend ontology, FOAF) ist eine Meta-Ontologie, die zwar kein großes Anwendungspotential in der Produktionsplanung findet, aber das Potenzial des SW in der Praxis dargelegt [34]. Das Grundprinzip der FOAF ist es, einzelne Personen an einer Internetadresse mit deren Homepage oder anderer digitaler Medien zu definieren. Des Weiteren werden Bekannte "Freunde" der zuvor genannten Person mit deren Internetadresse deklariert. Aufgrund der Verlinkung anderer Personen, wird ein dezentrales Netzwerk an Personen und deren Internetadressen aufgebaut.

Um die Informationen in einer Wissensbasis zu erlangen, kann SPARQL benutzt werden. SPARQL definiert ein Abfrage-System, das ähnlich zu SQL ist [35]. In einer SPARQL-Abfrage werden in Tripeln nach Elementen gesucht und in tabellarischer Form zurückgegeben. Die Abfrage geschieht mit Platzhaltern. Wenn das Subjekt ein Platzhalter ist, wird nach Subjekten gesucht, die mit dem gleichen Prädikat und Objekt verbunden sind. So ergeben sich Abfrage-Resultate.

2.3 Grundlagen der Montageprozessplanung

Die Montageplanung ist ein Teilgebiet der Prozessplanung. Die Montage ist einer der letzten Schritte der Produktionsprozesse [36]. In der Montageprozessplanung werden Aufgaben geplant, die in einem Montagesystem ausgeführt werden. Dafür ist sowohl wichtig, die Reihenfolge des Ablaufes zu planen als auch alle Anforderungen an so ein System, wie Anweisungen und benötigte Unterlagen, auszuarbeiten [37]. In der Definition nach Hirschbach [36] besteht ein Prozess aus einer Nachfrage nach einem Produkt, mit spezifischer Funktionalität, in einer bestimmten Stückzahl. Dieses Produkt erfüllt Anforderungen, die an die Nachfrage gestellt wurde. Die erstellte Methodik erzeugt Montageprozesse. Diese Prozesse können additiv zur Methodik an einem Optimierungsalgorithmus angehangen werden. Somit können die zur Produktion benötigten Prozesse optimal genutzt werden. Diese Arbeit beschäftigt sich

nicht mit der Prozessoptimierung, da erstmal ein Grundstein gesetzt wird, Prozesse überhaupt zu generieren.

Das Thema der Prozessoptimierung ist ein sehr großes Thema. Es gibt hier einige unterschiedliche Ansätze, um die optimale Prozessreihung zu finden. Blum [38] stellt einen Algorithmus auf, der an das biologische Verhalten von Ameisen gelehnt ist. Ein anderer Ansatz für die Optimierung der Pfade ist der genetische Algorithmus von Konak, Coit und Smith [39]. Dieser Algorithmus ist angelehnt an die Evolutionstheorie aus der Biologie. Eine weitere Optimierungsstrategie bei der Prozessplanung bezieht sich auf die Nutzung von Ressourcen. Markusheska et al. [40] beschreiben einen Allokationsalgorithmus, bei dem unabhängige Prozesse von einzelnen Ressourcen allokiert werden, um so eine maximale Verteilung von Prozessen zu erreichen.

Relevant für die vorzustellende Methodik ist, anhand welcher Metriken die Algorithmen Prozesse optimieren. Es ergibt sich, wie auch schon in der Definition nach Rudolf [37] aufgeführt, dass diese Algorithmen Performanz-Indikatoren benötigen. Ein Performanz-Indikator (*englisch*: key performance indicator, KPI) ist eine Metrik, mit dem Prozesse untereinander verglichen werden können. Einige dieser Indikatoren sind beispielsweise Kosten für die Umsetzung eines Prozesses, Zeitverbrauch der Montage, Energieverbrauch einer Ressource oder Nachhaltigkeit der Prozesse. Des Weiteren ist es von Bedeutung, aufzuführen, welche Prozesse vor anderen ausgeführt werden müssen. Dadurch kann ein Optimierungsalgorithmus korrekte Ergebnisse erzielen.

3 Anforderungsdefinition

Um eindeutige Ergebnisse zu liefern, müssen Kriterien an eine passende Ontologie definiert werden, damit sie in der vorgestellten Methodik genutzt werden kann. Diese Kriterien werden so definiert, damit eine ideale Ontologie beschrieben werden kann. "Ideal" heißt in diesem Kontext, dass die Ontologie die Kontexte Produktionsdesign und Montageprozesse komplett darstellen kann und daraus Prozesse generiert werden können, mit denen dann weiterverfahren werden kann.

Mittels dieser Kategorien können verschiedene Ontologien miteinander verglichen werden. Diese Kriterien werden dann auf die einzelnen Ontologien angewendet, um diese bewerten zu können. Durch die Erkenntnisse, die beim Literaturvergleich erzeugt wurden, kann eine eigene Ontologie entwickelt werden, welche die Problemstellung löst.

3.1 Aufstellen der Kriterien zum Vergleich verschiedener Ontologien

Im Literaturvergleich erhält jede Ontologie für jedes Kriterium jeweils eine Punktevergabe von -1, 0 oder 1. -1 bedeutet, dass eine Ontologie diese Anforderung gar nicht erfüllt, bei 0 wird es teilweise unterstützt und 1 bedeutet, dass die Ontologie eine Anforderung komplett unterstützt. Dies sind die Kriterien, die auf einer idealen Ontologie gestellt werden:

- **Einbinden von neuem Wissen**: Die Meta-Ontologie würde zu groß und komplex werden, wenn jeder mögliche Anwendungsfall betrachtet werden wür-

de. Um die Ontologie dennoch flexibel zu gestalten, muss eine Ontologie skalierbar sein. D. h. eine ideale Ontologie soll beispielsweise sowohl fünf als auch tausende Modelle darstellen können, ohne dabei die Komplexität überproportional zu übersteigen. Unter der Einbindung von neuem Wissen wird hier zwischen zwei Aktionen unterschieden: der **Skalierbarkeit** und der **Erweiterbarkeit**. Bei der Skalierbarkeit heißt eine Punktevergabe von -1, dass die Ontologie nicht ausreichend skalierbar ist und nicht auf die unterschiedlichen Größen flexibel reagieren kann, während dies bei einer Punktevergabe von 1 zutrifft. Bei der Erweiterbarkeit bedeutet eine Punktevergabe von -1, dass Konzepte der Ontologie und deren Kontexte erweitert werden können. Eine Punktevergabe von 1 bedeutet demnach, dass die ideale Ontologie deren erstellte Konzepte erweitern kann und neues Wissen dadurch einfach in die Ontologie integriert werden kann. Beide Aspekte werden im Literaturvergleich als getrennte Kategorien betrachtet, da verschiedene Ontologien in unterschiedlichem Maße Skalierbarkeit und Erweiterbarkeit eingebracht haben.

- **Multiperspektivität**: Um alle Stakeholder anzusprechen, ist es von Bedeutung, dass die ideale Ontologie unterschiedliche Abstraktionsebenen ansprechen kann. Um einzelne Bauteile zu erstellen, werden genaue Informationen über die Geometrie des Bauteils benötigt. Um aber die Architektur zwischen Produktionsschritten in Betracht zu ziehen, werden genaue Informationen über die Geometrie einzelner Objekte nicht beobachtet. Es werden die Wechselwirkungen zwischen verschiedenen Teilen betrachtet. Die Punktevergabe gibt an, wie sehr verschiedene Perspektiven unterstützt werden. Eine Punktevergabe von -1 bedeutet demnach, dass verschiedene Perspektiven gar nicht gebildet werden können, während dies bei einer Punktevergabe von 1 möglich ist.
- **Inferenzmöglichkeit**: Neben der Darstellung von Informationen, besitzt eine ideale Ontologie ebenfalls die Möglichkeit zur Inferenz. Erst durch die Generierung von neuen Fakten aus der expliziten Wissensbasis kann eine Ontologie auch für größere Projekte benutzt werden. Um Wissen aus Kontexten erschließen zu können und um Repetition zu vermeiden, ist dieses Kriterium das Wichtigste. Bei der Punktevergabe heißt -1, dass dies gar nicht unterstützt

ist; 0 heißt, dass es theoretisch unterstützt ist und 1 bedeutet eine komplette Unterstützung dieser Anforderung.

- **Validierungsmöglichkeit**: Es reicht nicht, dass man in einer idealen Ontologie neues Wissen einfügen kann. Überdies besitzt eine ideale Ontologie die Fähigkeit der Validierung. Ähnlich zu einem Programm, dass die Grammatik von Sätzen überprüft, gibt es eine Möglichkeit, dass die Konsistenz der Daten geprüft wird. Validierung ist ein wichtiges Konzept, um die Integrität und Validität der Ontologie zu sicher zu stellen. Die Punktvergabe gibt hier einen Anhalt darüber, inwiefern eine Validierungsmöglichkeit. Das heißt, dass bei -1 keine Validierung betrachtet wurde und schlecht umzusetzen ist, dass bei 0 Ansätze gezeigt wurden oder aufzufinden sind, mit denen Validierung möglich wäre und dass bei 1 eine Validierung möglich ist und auch in der Arbeit betrachtet wurde.

- **Benutzung eines geeigneten Darstellungsformates**: Jede Ontologie wird in einer Sprache formalisiert. In dieser Arbeit wird RDF und N3 als Darstellungsformat benutzt. Dies hat den Grund, dass die Ontologie von anderen Teilen des SW benutzt wird. In der Bewertung werden domänenspezifische Sprachen und Visualisierungs-Sprachen dahingegen schlechter bewertet, da der Aufwand größer ist, in RDF-N3 umgewandelt zu werden. Bei einer Punktzahl von 0 ist das Darstellungsformat nicht ideal im Kontext von SWT, aber es ist ansatzweise übertragbar. Wenn das Darstellungsformat schon in SWT benutzt wird, dann ergibt sich eine Punktzahl von 1.

- **Prozessgenerierung**: Die ideale Ontologie ist nicht nur in der Lage, Informationen darzustellen. Diese Ontologie ist außerdem in der Lage, Prozesse aus den eingegebenen Daten zu generieren. Eine Punktevergabe von -1 bedeutet, dass keine Prozesse aus der Ontologie generiert werden können; 0 bedeutet, dass theoretisch oder nur teilweise Prozesse generiert werden und 1 bedeutet, dass Prozesse komplett generiert werden können.

Für die einzelnen Kategorien ergeben sich individuelle Gewichtungen. Sie sind auf Basis der Relevanz für das Erstellen der eigenen Ontologie aufgebaut. Diese Auflistung wird auch in Tabelle 3.1 gezeigt. An erster Stelle steht die Inferenz, deswegen

hat sie eine Gewichtung von zwei. Nur mit der Inferenzmöglichkeit ergibt sich eine Ontologie, die dynamisch und flexibel eingesetzt werden kann. Die Validierungsfähigkeit und die Prozessgenerierung sind beide Aspekte, die fast genauso relevant sind. Mit der Validierung bleibt die Konsistenz erhalten und mit der Prozessgenerierung ergibt sich ein Nutzen für die Ontologie. Daher bekommen diese beiden Kategorien eine Gewichtung von eineinhalb. Alle anderen Kategorien bekommen eine Gewichtung von eins. Sie sind zwar Anforderungen, die erfüllt werden sollen. Aber die vorher genannten Kategorien haben eine höhere Relevanz in Bezug auf die vorgestellte Methodik.

Kategorie k	Gewichtung ω
Inferenzmöglichkeit	2
Validierungsmöglichkeit	1.5
Prozessgenerierung	1.5
Skalierbarkeit	1
Erweiterbarkeit	1
Multiperspektivität	1
Darstellungsformat	1

Tabelle 3.1: Relevanz-Liste der aufgestellten Kategorien im Kontext der vorgestellten Methode

3.2 Vorstellen der einzelnen Ontologien

Bevor die unterschiedlichen Ontologien miteinander verglichen werden, werden sie im Folgenden aufgelistet und in ihrer Funktion erläutert:

TOVE: Das Projekt des Toronto Virtual Enterprise (TOVE) definiert mehrere Ontologien zur virtuellen Unternehmens- und Prozessplanung. Die Organisations-Ontologie für das "TOVE enterprise model" ist eine dieser Ontologien. Koordination von Prozessen in Organisationen wird mit der Ontologie auf Basis von Prädikatenlogik übernommen [41]. Es werden Aktivitäten und Ermächtigung innerhalb

dieser Ontologie definiert. Die Definition geschieht über der domänenspezifischen Sprache "MODEL". Ähnlich zur bekannteren Sprache Prolog basiert diese Sprache auf logischen Fakten. Prolog ist eine Programmiersprache, dessen Hauptparadigma die logische Programmierung ist. Es werden logische Aussagen getätigt und mittels Prädikatenlogik und einem Lösungsalgorithmus namens "Backtracking" werden in Prolog verfasste Programme ausgeführt. Die EYE Inferenz-Maschine wurde in Prolog implementiert und nutzt den Fakt aus, dass RDF und Prolog auf Prädikatenlogik basiert. Damit werden dann logische Schlüsse in Richtung Arbeitsplanung und -aufteilungen getroffen.

AOD: Richard A. Wysk und Jeffrey S. Smith [42] definieren eine Formalisierung für zeitliche Prozessplanung in Produktionsstätten. Die Formalisierung ist konzeptuell und benutzt sogenannte UND/ODER-Digraphen (*englisch*: AND/OR-Digraphs, AOD). AOD bestehen aus gewichteten Graphen (engl.: "directed graph"), wobei die einzelnen Knoten Prozesse darstellen und die Kanten eine Reihenfolge für die Prozesse angeben. Kanten können dabei eine Auswahl von Möglichkeiten (OR) oder eine Parallelität (AND) von Prozessen ergeben.

OZONE: OZONE ist ein Projekt mit dem Ontologien für die Konstruktion aufgebaut und zeitlich geplant werden. Die in OZONE aufgestellte Ontologie definiert schon eine Vorstufe zum Aufbau: Prozesse, Produkte und Ressourcen (PPR). [43]. Bei diesem Modell werden Prozesse, Produkte und Ressourcen getrennt voneinander betrachtet und beschrieben, dadurch lässt sich eine Produktionsanlage beschreiben.

MASON: Die Semantik-Ontologie für die industrielle Herstellung (*englisch*: MAnufacturing's Semantics ONtology, MASON) ist eine Ontologie, die für die industrielle Herstellung gedacht ist. Diese Ontologie ist mit SysML und OWL-XML aufgebaut worden. Die Funktionalität der Selbst-Validierung und der Inferenz neuer wissensbasierten Daten, trotz der Benutzung von OWL mit vordefinierten Entailments, ist nicht ausgenutzt. [44]

MAMOT: Modellierungswerkzeug für Fertigung und Montage (*englisch*: Manufacturing and Assembly Modeling Tool, MAMOT) stellt nicht nur eine beispielhafte Ontologie auf, sondern beschreibt ein ganzes System. Dieses System ist vorgesehen, um Prozesse zu planen. Dies geschieht aufgrund einfacher Regeln, die angeben, welche Prozesse vor oder nach anderen Prozessen geschaltet werden müssen. Die in MAMOT beschriebene Meta-Ontologie ist sehr spezifisch gehalten für einige Fertigungsprozesse und hat demnach Potenzial abstrakt erweitert zu werden. Auch die Inferenz-Funktionalität kann erweitert werden, indem nicht nur die Prozesse am Ende deduziert werden. Das Verwenden der Regelsprache des Semantischen Webs (*englisch*: Semantic Web Rule Language, SWRL) ist an grafische Tools gebunden und nicht für den Datentransport gedacht. [5]

DELS: DELS kann als Weiterentwicklung oder Folgezustand von OZONE betrachtet werden. Es zeichnet sich durch die Definition von Netzwerken und der Erweiterung des PPR aus. In dieser Arbeit wird die Implementation jedoch bewusst klar und verständlich gehalten, um einen effizienten Nachbau dieser Ontologie zu ermöglichen. [45]

PPR-DSL: Die Domänenspezifische Sprache für Prozess-,Produkt- und Ressourcenmodellierung (*englisch*: Domain-Specific Language for Product-Process-Resource Modeling, PPR-DSL) bringt gute Grundzüge mit sich, bei der eine Domänenspezifische Sprache (*englisch*: domain-specific language, DSL) definiert wird, die für PPR-Prozesse gedacht sind. Dieses Darstellungsformat benutzt jedoch keine im SW standardisierten Technologien. Als einzige Ontologie aus der Literatur definiert diese Ontologie auch eine eigene Syntax, mit der eine Validierung ausgeführt werden kann.[46]

CAO: Die Kabinenmontage-Ontologie (*englisch*: Cabin assemly ontology, CAO) definiert wie MAMOT eine grobe Meta-Ontologie, die auch abstrakter definiert werden kann. Dies liegt daran, dass sich die Arbeit mehr auf die Produktionsplanung und -optimierung konzentriert. Definierte Attribute und Eigenschaften, die dort für

die Prozessplanung gebraucht werden, können für die selbst definierte Ontologie entnommen werden. [40]

3.3 Vergleich unterschiedlicher Ontologien aus der Literatur

Tabelle 3.2 zeigt den Vergleich der unterschiedlichen Ontologien. Diese Ontologien wurden anhand der vorher definierten Anforderungen untereinander verglichen. Dadurch ergeben sich Details, die für das Erstellen der eigenen Ontologie benutzt werden.

Kategorie k	Gew. ω	TOVE [41]	AOD [42]	Ozone [43]	MASON [44]	MAMOT [5]	DELS [45]	PPR-DSL [46]	CAO [40]
Skalierbarkeit	1	-1	0	0	1	1	1	1	1
Erweiterbarkeit	1	1	0	1	1	0	1	-1	1
Multiperspektivität	1	-1	1	-1	0	0	0	-1	-1
Inferenzmöglichkeit	2	1	-1	1	-1	0	1	-1	-1
Validierungsmöglichkeit	1.5	0	-1	-1	-1	0	-1	1	-1
Darstellungsformat	1	-1	-1	0	1	1	0	-1	1
Prozessgenerierung	1.5	0	1	1	0	1	0	-1	1
Summe (max.: 9)	/	0	-2	2	-0.5	3.5	2.5	-4	0

Tabelle 3.2: Vergleich verschiedener Ontologien aus der Literatur
Hier wird für jede Ontologie in jeder Kategorie eine aufsteigende Bewertung zwischen -1 und 1 zugeteilt. Die summierte und gewichtete Punktzahl gibt einen Anhalt für die Relevanz innerhalb dieser Arbeit an.

In Tabelle 3.2 wurden unterschiedliche Quellen aus der Literatur angegeben. In Gleichung (3.1) wird gezeigt, wie die Summe errechnet wird. Jede Quelle i hat zu jeder Kategorie k eine Punktzahl bekommen. Zusammen mit der Gewichtung ω_k erhält jede Punktzahl eine gewichtete Summation σ_i der Punkte. Die ideale Ontologie hat eine ideale Punktzahl von 9. Sollte jede Kategorie erfüllt sein, ist die dargestellte Ontologie ideal. Die einzelnen Werte σ_i geben dann einen Anhalt für die Relevanz

der vorgestellten Ontologie in diesem Kontext. Punktevergaben einzelner Kategorien sind dabei nicht zu vernachlässigen.

$$\sigma_i = \sum_{k \in K} k_i * \omega_k \tag{3.1}$$

Aus dem Vergleich unterschiedlicher Ontologien ergeben sich folgende Schlüsse: Keine der dargestellten Ontologien besitzt die maximale Punktzahl. MAMOT und DELS besitzen aber die Ontologien mit den höchsten Punktzahlen von 3.5 und 2.5. Diese Ontologien sind demnach die relevantesten Ontologien, aufgrund der Punktevergabe lässt sich aber Lücken in der Umsetzung aufweisen, welche in anderen Ontologien vertreten sind. Die beiden Ontologien reichen demnach nicht als eigenständige Ontologie für diesen Anwendungsfall aus. Daher ergeben sich auch Auswertungen für die einzelnen Anforderungen:

Sowohl bei der Skalierbarkeit, als auch bei der Erweiterbarkeit einer Ontologie gibt es mehrere für diesen Anwendungsfall passenden Ontologien. Es lässt sich aus diesen Ontologien entnehmen, dass für die Skalierung der Wissensbasis die Ontologie klar getrennte Meta-Schemata definieren muss. So muss beispielsweise bei TOVE jedes Mal die ganze Wissensbasis auf Konsistenz geprüft werden, bevor neue Wissenselemente hinzugefügt werden können. Bei der Erweiterbarkeit ist es nötig, wie in Ozone, DELS oder PPR-DSL abstrakte Grundklassen zu definieren, auf denen eigene Schemata basieren. Wie auch in diesen Beispielen wird die eigene Ontologie auch auf dem PPR-Modell beruhen. Mithilfe der Definition von spezifischen Montageprozessen auf Basis einer Prozess-Grundklasse zeigen die Ontologien eine ausreichende Erweiterbarkeit.

Auf die Multifidelität wurde nicht oft in den vorgestellten Ontologien eingegangen. Lediglich AOD und MAMOT benutzen diese Konzepte, wenngleich auf unterschiedlicher Herangehensweise. AOD definiert in deren Ontologie verschiedene Abstraktionsebenen, anhand derer Prozesse generiert werden können. MAMOT definiert für einzelne Bauteile die Multifidelität als Variable, nach der eine Genauigkeit von Prozesszeiten und Kosten bestimmt werden.

Die Inferenzmöglichkeit wurde nicht in jeder Ontologie ausgenutzt. So scheinen MAMOT und CAO die Inferenz beim Erstellen der Wissensbasis zu benutzen. TOVE zeigt aber, dass durch logische Schlussfolgerungen iterativ neues Wissen generiert werden kann.

Für die Validierungsmöglichkeit ist nur PPR-DSL relevant. Denn durch eine DSL werden wesentliche Syntaxfehler gefunden. Die Validierung aus dieser Ontologie muss für die eigene Ontologie überarbeitet werden, da es sich um komplett unterschiedliche Darstellungsweisen handelt.

Bei der Darstellungsart lässt sich sehr schlecht die Domäne definiert in TOVE, AOD oder PPR-DSL übernehmen. Dies liegt an der sehr unterschiedlichen Syntax und Paradigmen. Hingegen MAMOT, MASON und CAO definieren mithilfe anderer SWT deren Ontologie.

Für die Prozessgenerierung stellt sich heraus, dass AOD und Ozone für die Prozessplanung gedacht sind, genauso wie MASON und CAO. Obwohl der Algorithmus dafür nicht ganz ersichtlich ist, definiert DELS das Konzept von Flussnetzwerken, mit denen Prozesse abgefragt werden können.

4 Vorstellung der Methodik

Auf Grundlage der vorher erstellten Kriterien und der Grundlagen für KBE und SW wird hier eine eigene Methodik vorgestellt. Kernaspekt dieser Methodik ist das Erstellen einer Ontologie und deren Auswertung. Dabei wird eine Ontologie in zwei Teilen aufgebaut, der strukturierten Meta-Ontologie und der Informationsbasis. Letztendlich kann dadurch mittels Inferenz neues Expertenwissen aufgebaut und programmatisch Prozesse ausgelesen werden.

4.1 Methodik-Aufbau

Der Aufbau der vorgestellten Methodik wird anhand der Grafik in Abbildung 4.1 gezeigt. Angefangen mit den domänenspezifischen Konzepten, welche die Montageprozesse und das Flugzeugdesign erklären, wird ein Metamodell erstellt. Eigenschaften und Anforderungen aus der realen Welt werden zu einer Wissensbasis formuliert. Mithilfe des Metamodells werden die erstellten Daten validiert. Sowohl das Metamodell als auch die Informationsbasis bilden gemeinsam die Ontologie, wie bereits in den vorherigen Kapiteln erläutert wurde. Die resultierende Ontologie wird in eine Datenbank übertragen, damit Daten persistent gespeichert werden können. Damit kann die Wissensbasis erweitert werden. Die Datenbank kann dabei ein Datenbank-Managementsystem, wie Berkeley DB oder InfluxDB sein. Damit ist ein System gemeint, dass die Datenverwaltung einer Datenbank kontrolliert, um Persistenz und korrekten Zugriff zu gewährleisten. Auch ein kompaktes Datenformat wie HDT dient der effizienten Speicherung von RDF-Tripeln. Ein eigenes Datenbank-Managementsystem müsste erstellt werden, um damit auf gleicher Art wie Berkeley DB und InfluxDB zu agieren.

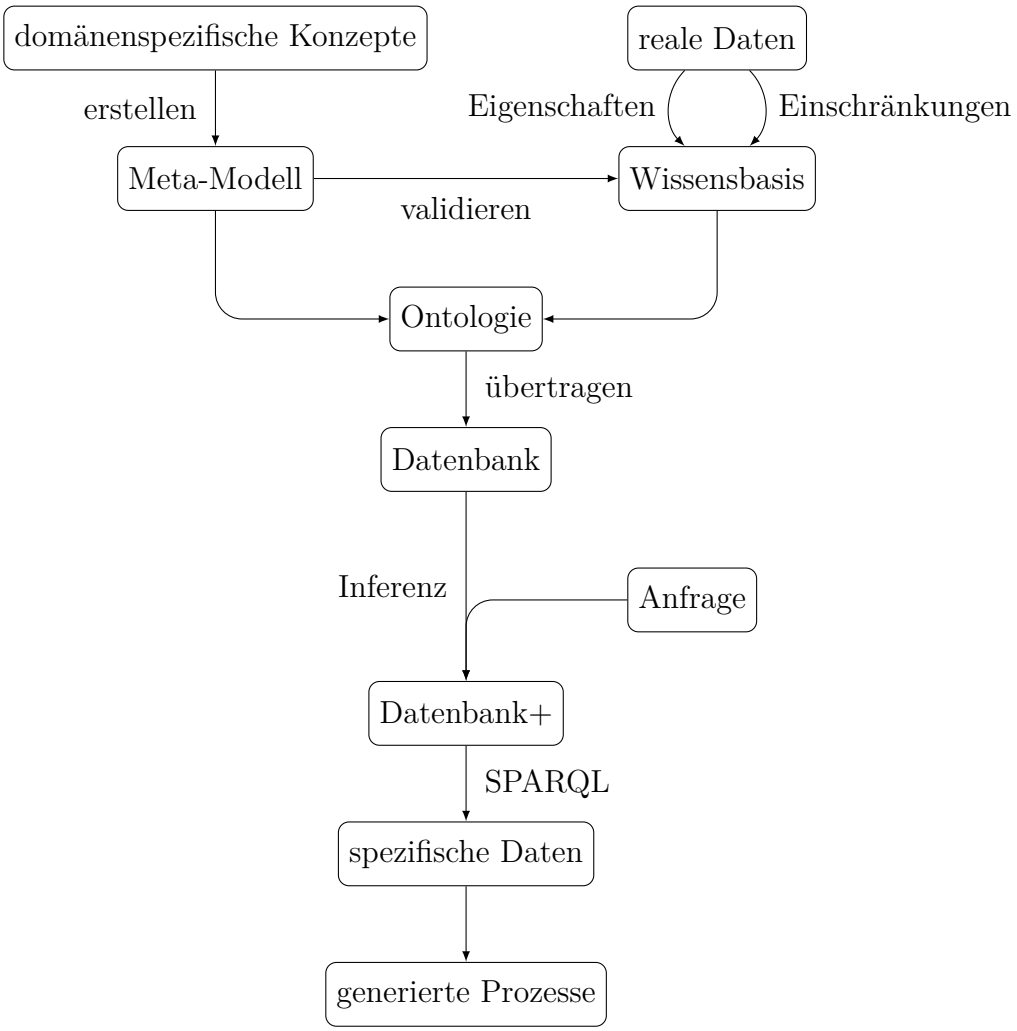


Abbildung 4.1: Flussdiagramm zur Methodik zum Aufbau der Wissensbasis

Um die Ontologie nutzbar zu machen, reicht es nicht aus, einfache Daten abzuspeichern. Wie in Kapitel 2 und Kapitel 3 formuliert, ist die Inferenz ein wichtiger Aspekt der vorgestellten Methodik. Das heißt, dass die Datenbank um inferiertes Wissen erweitert wird. Es ergibt sich eine erweiterte Datenbank oder auch eine "Datenbank+", in der inferiertes Wissen vorliegt. Die inferierten Daten können wiederholt von der Inferenz-Maschine gefunden werden. Da die zusätzlichen Wissensobjekte direkt nach dem Erstellen der erweiterten Datenbank benutzt werden, reicht es, die Datenbank+ für eine kurze Zeit abzuspeichern. Die erweiterte Datenbank muss demnach nicht persistent gespeichert werden. Es reicht lediglich aus, sie nur in einem flüchtigen Speicher zu lagern. Um Prozesse zu generieren, Systemanalysen oder andere Algorithmen anzuwenden, werden externe Anfragen aufgestellt. Die Daten werden wie die Ontologie mit RDF-Fakten dargestellt.

Daher ist es praktisch, SPARQL zu benutzen, um aus der erweiterten Datenbank für das Problem spezifische Daten zu extrahieren. Mit SPARQL werden die Anforderungen zu Abfragen umgewandelt. Diese Abfragen werden auf die erweiterte Datenbank angewandt. Die resultierenden Daten werden in die kommenden Algorithmen eingeführt. Dabei ist der Algorithmus für die Prozessplanung nicht von Bedeutung. Aus den spezifischen Daten ergeben sich die generierten Prozesse.

4.2 Ontologie-Aufbau

Die Erkenntnisse aus dem Literaturvergleich führen zwangsläufig zu einer Definition der eigenen Ontologie. Kritikpunkte und fehlende Aspekte werden hier durchgesetzt, um die vorangestellten Anforderungen zu erfüllen. Die Definition der eigenen Ontologie besteht aus mehreren Aspekten, wobei das wichtigste die Definition der Meta-Ontologie ist.

4.2.1 Aufbau der Meta-Ontologie

Die Meta-Ontologie basiert auf dem PPR-Modell. Abbildung 4.2 zeigt das PPR-Model im Kontext der Meta-Ontologie. Die Modellierung ist vergleichbar zur RDF-

Definition. Es werden Klassen definiert und sie werden untereinander in Kontext zu Prädikaten gestellt. Die Klassen werden mit Rechtecken beschrieben und Prädikate werden mit gerichteten Verbindungen angegeben. Das PPR-Modell ist direkt mit “Product”, “Process” und “Resource” dargestellt. Die Produkt-Klasse steht im Kontext zu anderen Klassen mit drei Verben. Ein Produkt wird kreiert von einem Prozess (“created by”), es wird produziert mit einer Ressource (“produced with”) und es besteht aus anderen Produkten (“consist of”). Dass ein Produkt aus anderen Produkten besteht, sorgt für eine Verkettung von mehreren Produkten. Bestimmte Produkte, wie Grundmaterialien oder vorproduzierten Materialien, bestehen nicht mehr aus anderen Produkten, daher ist dieses Verb optional. Ein Prozess hat eine Eingabe und eine Ausgabe (“hasInput”/“hasOutput”). Eine Ressource stellt Prozesse zur Verfügung und befindet sich in einer Einrichtung (*englisch:* “Facility”). Angelehnt an DELS wurde das PPR-Modell mit der Einrichtung erweitert, die den Standort der zugeordneten Ressource angibt. Wie in OZONE existiert eine Nachfrage (*englisch:* Demand). Eine Nachfrage stellt eine Anfrage auf ein Produkt. Zur Modellierung einer Produktionsstätte ist die Nachfrage nicht notwendig. Das PPR-Model stellt den Bestand einer Werkstätte dar. Mit der Nachfrage werden Möglichkeiten inferiert, mit denen ein angefragtes Produkt über bestimmte Prozesse erstellt werden kann. Die Möglichkeit zur Inferenz ergibt sich aus der Semantik der einzelnen Objekte der Meta-Ontologie.

Die Akteure und Verben sind in einen impliziten Kontext der Montageplanung gestellt, der an dieser Stelle nicht explizit definiert ist. Ein KBS kann nicht mit dieser Ontologie ohne weiteres arbeiten. Daher muss explizites Wissen gegeben werden. Es ergibt sich die Frage, was in Abbildung 4.2 definiert wird. Es werden Meta-Klassen und Meta-Eigenschaften beschrieben. Eine Meta-Klasse ist hier eine Klasse, aus der entweder direkt instantiiert wird oder eine Unterklasse vererbt wird. Eine Meta-Eigenschaft wird in der Informationsbasis an den instanziierten Objekten direkt benutzt und definiert eine Definitions- und eine Zielmenge. In mathematischen Funktionen wird die Definitionsmenge als erlaubte Eingabemenge der Funktion betrachtet. Die Zielmenge repräsentiert die erlaubte Ausgabemenge. Innerhalb von RDF-Tripeln beschreibt die Definitionsmenge die erlaubte Menge der Subjekte. Parallel dazu definiert die Zielmenge die erlaubte Menge der Objekte. Meistens handelt

es sich bei den erlaubten Mengen um Datentypen oder Grundklassen. Diese Mengen-Definitionen ermöglichen eine spätere Validierung. Dadurch, dass in den Instanzen die Meta-Eigenschaften direkt benutzt werden, ergibt sich eine vordefinierte Menge an Tripel-Fragmenten, die in SPARQL abgefragt werden können. Das erlaubt ein programmatisches Abfragen von Informationen in der Ontologie.

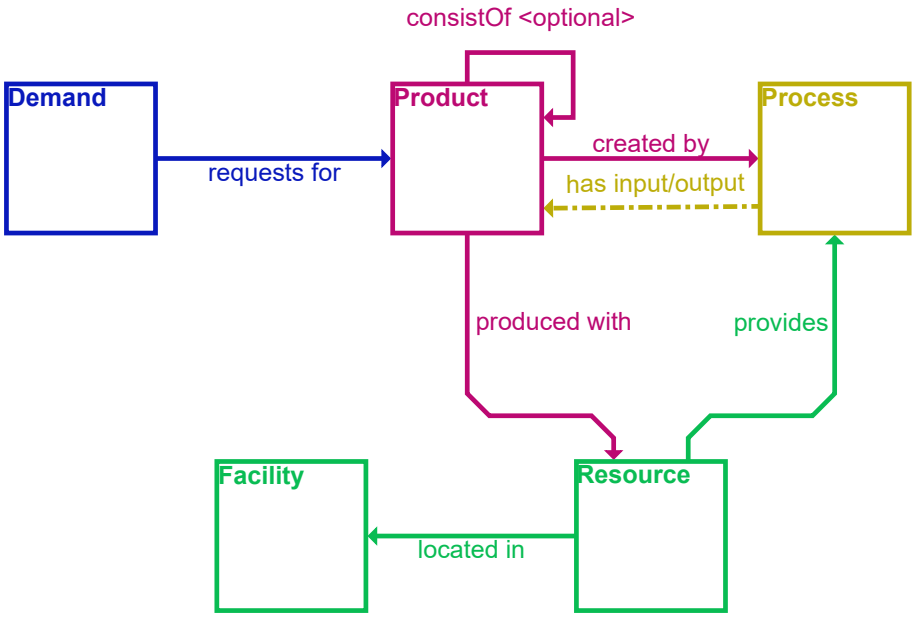


Abbildung 4.2: Meta-Ontologie auf Basis des PPR-Modells

4.2.2 Analyse der Kriterien für die ideale Ontologie

Die aufgestellte Meta-Ontologie implementiert die für die ideale Ontologie aufgestellten Kriterien. Das Kernprinzip der Methodik, die **Inferenz**, ist in der aufgestellten Methodik mittels regelbasierten RDF-Subgraphen definiert. Anhand dessen werden neue Wissensobjekte inferiert. Die Inferenz wird mit einer Inferenz-Maschine durchgeführt, die RDF-Tripel mit der kompletten N3-Syntax einlesen kann. Das heißt, dass sowohl RDF-Tripel als auch Regeln mit Subgrafen definiert werden können. **Validierung** wird ebenfalls mit einer Inferenz-Maschine durchgeführt. Da mittels logischen Regeln eine falsche Anwendung der Ontologie überprüft werden kann, kann direkt von der Inferenz-Maschine eine Inkonsistenz festgestellt werden. Die **Prozess-**

generierung geschieht mittels SPARQL-Abfragen. Die inferierte erweiterte Datenbank besitzt im Falle einer passenden Nachfrage nach der Fertigung eines Produktes alle Daten, die zum Auslesen von möglichen Prozessen benötigt werden. In einer Prozedur wird mit den SPARQL-Abfragen programmatisch die benötigten Daten gefunden und tabellarisch notiert. Wie in Kapitel 5 zu sehen sein wird, ist die tabellarische Auffassung der Prozessdaten ein visuelles Werkzeug, um zu zeigen, dass die Daten digital verarbeitet werden können. Die **Skalierbarkeit** der einzelnen Daten innerhalb der Ontologie ist aufgrund der Inferenz-Maschine möglich, da mit einer ausreichend effizienten Inferenz-Maschine RDF-Tripel unabhängiger Menge bearbeitet werden können. Da die Meta-Ontologie auf simplen Basisklassen basiert, ist die kontextabhängige **Erweiterbarkeit** auch garantiert. Abhängig von der erweiterten Meta-Ontologie ist auch eine **Multiperspektivität** möglich. Und mittels der Nutzung von der N3-Syntax ist schon gezeigt, dass dieses Darstellungsformat in dem Rest des SW einzubinden ist, da schon allein RDF-Tripel ein Teil der SWT sind.

4.3 Implementierung der Methodik

Technologie	Autor	Version
EYE	Jos De Roo	4.10.1
SWI-Prolog	Jan Wielemaker et al.	9.0.4
Python	Guido van Rossum	3.11.4
rdflib	RDFLib Team	7.0.0

Tabelle 4.1: Liste der benutzten Technologien für die Umsetzung der Methodik

Tabelle 4.1 zeigt die benutzten Technologien, die zur Umsetzung der Methodik benutzt wurden. Zur Durchsetzung der Inferenz wird die Euler Yet another proof Engine (EYE) Inferenz-Maschine genutzt, da diese Inferenz-Maschine nach Ergebnissen von Arndt und Mennicke [47] eine höhere Inferenzgeschwindigkeit im Vergleich zu anderen Inferenz-Maschine aufweist. Die EYE Inferenz-Maschine inferiert Wissen auf Grundlage von N3-Wissensobjekten und der N3-Logik [16]. Um EYE als Konsolenprogramm auszuführen, wird Prolog benutzt. Für den Rest der Methodik wird Python verwendet. Eine Applikationsprogrammier-Schnittstelle (*englisch*:

Application Programming Interface, API) zum automatischen Auslesen und Verwalten von RDF-Tripeln ist “rdflib”. Eine API ist eine Ansammlung von Funktionen, Datenstrukturen und Metadaten, die eine Schnittstelle zum Programmieren geben. Mit “rdflib” lassen sich Tripel in den untershciedlichen Darstellungsformaten XML, JSON, TURTLE und letztendlich auch N3 ein- und auslesen. Des Weiteren bietet diese API eine Schnittstelle zur Abfrage von RDF-Daten mittels der SPARQL-Sprache.

Innerhalb der Methodik entstehen mehrere Dateien, die einen Teil der kompletten Ontologie aufbauen. Abbildung 4.3 zeigt diese die Dateien, aus denen die Ontologie besteht. Die Meta-Ontologie liegt in der “ppr.n3”. In dieser Datei sind Klassen, Eigenschaften und Regeln des PPR-Modells definiert. Die realen Daten und die Anfrage auf ein Produkt liegt in “usecase.n3”. Weitere Vokabulare aus dem SW liegen als Internetadressen vor. Für diese Methodik handelt es sich hauptsächlich um das RDFS. Da dieses Dokument nicht in der Methodik, sondern extern definiert ist, ist sie gestrichelt in Abbildung 4.3 markiert. Das RDFS definiert Entailments, diese müssen in dieser Methodik in einer zusätzlichen Datei “entailments.n3” selbst aufgestellt werden. Dies liegt daran, dass das Internetdokument nur aus einfachen Tripeln, d. h. ohne Subgrafen und Regeln, besteht. Das Selbst-Aufstellen der Entailments erweist sich als praktisch, da die RDFS-Entailments das Konzept der Reifikation benutzen. Das heißt, dass für jedes Tripel weitere Tripel erstellt werden, die angeben, was ein Subjekt, Objekt, Prädikat oder Tripel ist. Daraus ergeben sich für jedes Tripel vier weitere Tripel. Daraus entsteht eine unpraktisch hohe Anzahl an Wissensdaten. Diese Wissensdaten sind redundant für diese Methodik, da mit SPARQL diese Fakten abgefragt werden können. Nachdem die Meta-Ontologie erstellt wurde, werden die realen Daten eingeführt. Mittels der EYE Inferenz-Maschine werden die ganzen Dokumente zusammengenommen, um Inferenz anzuwenden. Es ergibt sich eine große N3-Datei, die alle Wissensdaten und das inferierte zusätzliche Wissen in einer Datei abspeichern. Sie wird in der Abbildung mit “inferred.n3” bezeichnet, da es sich um inferiertes Wissen handelt.

Um auf die erweiterte Datenbank zugreifen zu können, wird mittels einer SPARQL-Abfrage überprüft, welche Objekte der Meta-Klasse “Demand” untergeordnet sind. Das heißt, dass Nachfragen gesammelt werden. Die instanziierte “Demand” stellt ei-

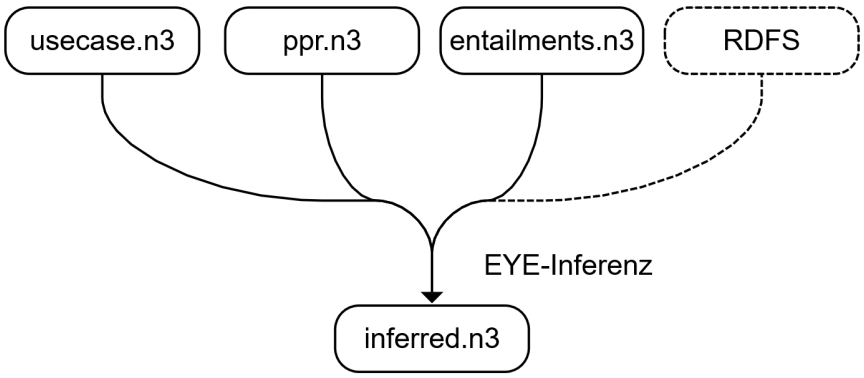


Abbildung 4.3: Inferenz mehrerer Dokumente mit der EYE Inferenz-Maschine

ne Anfrage auf Produkte. Durch alle angefragten Produkte wird aufgestellt, wie die Produkte erstellt werden. Das heißt, dass Grundbestandteile in Form von Produkten gesucht werden; Prozesse, mit denen die resultierenden Produkte produziert werden und welche Ressourcen damit assoziiert werden. Listing 4.1 zeigt eine SPARQL-Abfrage, die benutzt werden kann, um die geforderten Produkte auszulesen. Innerhalb dieses Codes wird nach einer Anfrage gesucht, die nach einem Produkt sucht, wobei das Produkt selber keine Meta-Klasse sein kann.

```
1  SELECT ?Demand ?ReqProd
2  WHERE {
3      ?D a meta:Demand .
4      ?D meta:requestsFor ?ReqProd .
5      FILTER(!EXISTS {
6          ?D a meta:MetaClass .
7      })
8  }
```

Quelltext 4.1: Abfragen von Anfragen vom Endnutzer an die aufgestellte Wissensbasis über SPARQL

Die in Listing 4.2 dargestellte SPARQL-Abfrage zeigt beispielhaft, wie Prozesse samt ihrer zugehörigen Daten abgefragt werden können. Dazu wird in "<REQ>" das benötigte Produkt eingesetzt. Dieses Produkt ist die Ausgabe des gesuchten Prozesses, nach dem die SPARQL-Abfrage sucht. Zusätzlich sucht diese Abfrage nach den eingegebenen Produkten, welche Ressource diesen Prozess generiert und wo sich diese Ressource befindet.

```
1  SELECT ?Proc ?Resource ?Facility ?Input
2  WHERE {
3      ?Proc meta:hasOutput <REQ> .
4      ?Resource meta:provides ?Proc .
5      ?Resource meta:locatedIn ?Facility .
6      ?Proc meta:hasInput ?Input .
7  }
```

Quelltext 4.2: Abfragen der inferierten Prozesse vom Endnutzer an die aufgestellte Wissensbasis über SPARQL

5 Validierung der Methodik anhand zweier Anwendungsbeispiele

Um die aufgezeigte Methodik mit der erstellten Ontologie evaluieren zu können, ist eine qualitative Auswertung nur erschwert möglich. Es hängt vom Anwendungsfall ab, wie die Leistung der aufgestellten Ontologie zu bewerten ist. Um die aufgestellten Kriterien aus dem Literaturvergleich Kapitel 3 mit dieser Methodik zu vergleichen, sind qualitative Werte nicht möglich. Im Falle einer Anwendung von Inferenz ist es abhängig von der Inferenz-Maschine, wie schnell, speichereffizient usw. die Methodik ist. Quantitative Werte für die Inferenz zu benutzen könnte demnach ein verschobenes Bild auf die eigenen oder anderen Ontologien werfen. Um beispielsweise Erweiterung oder Validierung zu evaluieren, ist es abhängig von dem Anwendungsfall, wie viele Werte eine Ontologie beeinflussen. Da zusätzlich die verschiedenen Ontologien unterschiedlich schwergewichtig sind, können konkrete Daten auch ein falsches Bild auf die jeweilige Ontologie werfen. Daher ergibt sich, dass eine qualitative Bewertung der Methodik und Ontologie zu erbringen ist. Eine Aussage über die Qualität der erstellten Ontologie ist vergleichbar zum Literaturvergleich, bei dem verschiedenen Kriterien miteinander verglichen wurden. Durch einen qualitativen Vergleich ergeben sich Referenzpunkte, an dem die aufgestellte Ontologie mit anderen Ontologien verglichen werden kann. Schlussendlich kann eine Aussage über die Validität dieser Methodik getroffen werden, da die aufgestellte Ontologie der Kernbestandteil der Methodik ist.

Diese Art der Evaluation wird nach Hildebrandt [48] gestützt. Dabei wird zur qualitativen Auswertung von erstellten Ontologien eine Evaluationsmethodik aufgestellt. Diese Methodik ist in fünf Schritten aufgebaut:

1. Definition des Anwendungsbeispiels
2. Aufstellen des Projektplanes
3. Anwenden entwickelter Artefakte
4. Analyse und Ergebnisse
5. Dokumentation der Anwendung

Das heißt, das anfänglich ein Anwendungsbeispiel definiert wird sowie deren Zielstellung. Um die nach Hildebrandt [48] aufgestellten Methoden zum Erstellen von Ontologien zu evaluieren, werden Anwendungsbeispiele, oder auch Fallstudien, definiert. Die nach Hildebrandt [48] erstellten Methoden werden als Artefakte bezeichnet. Artefakte sind Teile der ganzen Methodik, die einen Rahmen bieten zum Nachbauen in anderen wissenschaftlichen Arbeiten. Ein Anwendungsfall zeigt mindestens ein Artefakt, was hier gleichgesetzt wird mit einem Anforderungskriterium aus dem Literaturvergleich. Durch diesen Anwendungsfall wird untersucht, inwiefern ein Artefakt in der Ontologie vertreten ist.

5.1 Beschreibung des ersten Anwendungsfalles

Der erste Anwendungsfall beschreibt einen Turm, der mit Klemmbausteinen aufgebaut wird. Es handelt bei der Montage des Turmes nicht um Prozesse, die direkt aus der Flugzeugkabinenfertigung stammen, sondern um Montageprozesse. Daher ist der Anwendungsfall übertragbar in die Luftfahrtindustrie. Zusätzlich ist der Anwendungsfall in einer niedrigen Komplexität aufgebaut, um die nötigsten Bestandteile für die Anforderungen zu evaluieren. Wenn der Grundaufbau gestellt wurde, lassen sich anhand dessen komplexere Anwendungen erstellen, die sich auch in der Luftfahrtindustrie befinden.

Neben der Erfüllung der Anforderungen zeigt dieser Anwendungsfall, dass Produktvarianten betrachtet werden können, um so Varianten-Bildung in der Luftfahrt darzustellen. Die Produkte, die den Turm ausmachen, sind entweder rote oder blaue

Bausteine. Anhand dessen wird gezeigt, dass Prozesse generiert werden können, wobei Varianten bereits berücksichtigt sind.

5.2 Beschreibung des zweiten Anwendungsfalles

Das zweite Anwendungsbeispiel untersucht, ob es möglich ist, verschiedene Prozesse zu betrachten, mit dem das gleiche Produkt kreiert werden soll. Zusätzlich wird untersucht, ob multiple KPIs unterstützt werden. Der Anwendungsfall besteht aus einem Produkt, nach dem eine Anfrage gestellt wird. Dieses Produkt besteht aus zwei Teilen, die zusammen weiterverarbeitet werden. Jedes dieser beiden Teile besteht wiederum aus Simplizitätsgründen aus dem gleichen Material. Da der Anwendungsfall so zu abstrakt ist und um einen Kontext in die Luftfahrt zu geben, wird der Anwendungsfall so beschrieben: Die beiden Teile sind ein Stringer und eine Rumpfhaut. Dewald, Klimmek, Algermissen u. a. [49] beschreiben Stringer und Rumpfhaut als strukturelle Elemente einer Flugzeugkabine, dessen Modelle unter anderem digital erstellt werden können. Wenn beide Teile zusammengeführt und montiert werden, entsteht eine verstärkte Rumpfhaut. Sowohl die Rumpfhaut als auch der Stringer besteht aus dem gleichen Material, beispielsweise einer Aluminiumlegierung.

5.3 Aufbau der Wissensbasis

Nachdem die Anwendungsfälle vorgestellt wurden, müssen sie aufgebaut und durchgeführt werden.

5.3.1 Wissensbasis des ersten Anwendungsfalles

Die Informationsbasis des Anwendungsfalles ist getrennt von der Meta-Ontologie definiert. Das Grundprinzip der Ontologie ist die Erweiterung auf einen spezifischen Anwendungsfall. Daher kann der Anwendungsfall leicht in das bestehende System integriert werden.

Um diesen Anwendungsfall darzustellen und auszutesten, wird anfangs nur das Nötigste formuliert und danach werden komplexere Teile integriert. Anfangs reicht es aus, die fünf Grundklassen zu instanziieren. In der Informationsbasis werden anfänglich die einzelnen Instanzen der Grundklassen in Form von Wechselwirkungen definiert. So wird beschrieben, dass eine Anfrage auf einen Klemmbaustein-Turm existiert. Der Turm besteht aus einzelnen Bausteinen. Es gibt einen Montageprozess, der die Bausteine als Eingabe nimmt und einen Turm erzeugt. Dieser Prozess wird von einem Roboter gestellt, der sich in einer Werkstätte befindet. Dies wird in Abbildung 5.1 visualisiert.

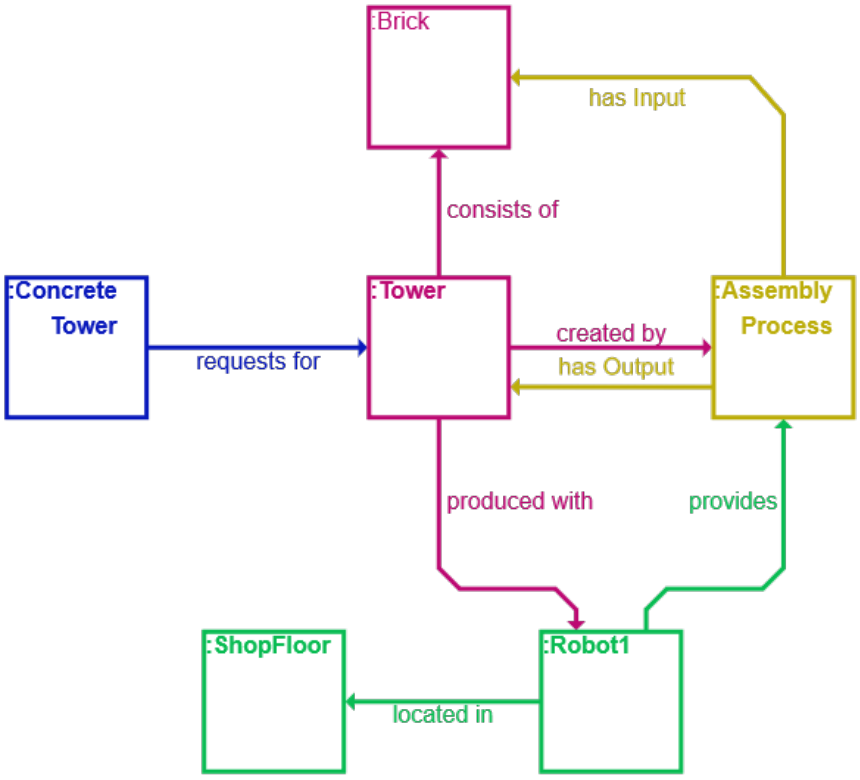


Abbildung 5.1: Wechselwirkungen einzelner Instanzen des Anwendungsfalles

Aufgrund der in der Meta-Ontologie definierten Inferenzregeln, kann darauf geschlossen werden, was schon in Abbildung 5.1 zu beobachten ist. Da die Prädikate in

ihrem Kontext eindeutig definiert wurden, kann für jede Instanz ihre Grundklasse bestimmt werden: Der Turm und die Bausteine sind Produkte, der Montageprozess ist ein Prozess, der Roboter eine Ressource, die Werkstätte ein Ort und ":ConcreteTower" ist eine Nachfrage. Da die Basisklassen aufgrund des vorhandenen Kontextes erschlossen werden kann, sind die Instanzen und die Wechselwirkungen in der gleichen Farbe wie die Grundklassen markiert.

Die in Abbildung 5.1 dargestellten Instanzen sind nur eine Darstellung des Konzeptes dieses Anwendungsfalles. Die eigentlichen RDF-Tripel lassen sich wie in Kapitel 2 in eine grafenbasierte Visualisierung überführen. Dies ist in Abbildung 5.2 gezeigt. Zusätzlich zu den in Abbildung 5.1 aufgestellten Instanzen, wird beschrieben, dass ":ConcreteTower", also die Nachfrage auf einen Turm, von der Nachfragen-Metaklasse instanziiert. ":Robot1" ist eine Instanz von der Klasse der automatisierten Ressourcen. Diese Klasse ist eine Subklasse der Metaklasse für Ressourcen. Eine weitere Subklasse ist die Klasse der menschlichen Ressourcen. Damit sind Mitarbeitende Kräfte gemeint, die an der Werkstätte arbeiten. Auffällig ist, dass diese Klasse keine Instanzen besitzt und sonst auch keine Wechselwirkungen mit anderen Instanzen oder Basisklassen aufweist. Dies ist ein Aspekt von Wissensbasen, welcher schon in Kapitel 2 beschrieben wurde. Die Wissensbasis kann eindeutig größer sein, als der Anwendungsfall es voraussetzt. Die relevanten Daten können mittels SPARQL-Abfragen extrahiert werden.

5.3.2 Wissensbasis des zweiten Anwendungsfalles

Um die multiplen KPIs anzuzeigen, wird ein zusätzliches Element in der Meta-Ontologie gebraucht. Abbildung 5.3 zeigt diese Erweiterung. Dabei wird das mit braun markiertes Element hinzugefügt: ":hasKPI". Dies ist eine Meta-Eigenschaft, die beschreibt, dass eine Meta-Klasse eine KPI besitzt. Die vorgestellte Eigenschaft kann vererbt werden und je nach Anwendungsfall angesetzt werden. Dieser Anwendungsfall besitzt zwei KPIs, Zeit und Kosten. Die Zeit ist ein KPI, die die Ausführungsdauer von Prozessen misst. Mit Kosten ist ein KPI gemeint, der angibt, wie viel die Ausführung eines Prozesses kostet. Zeit wird in Sekunden gemessen und

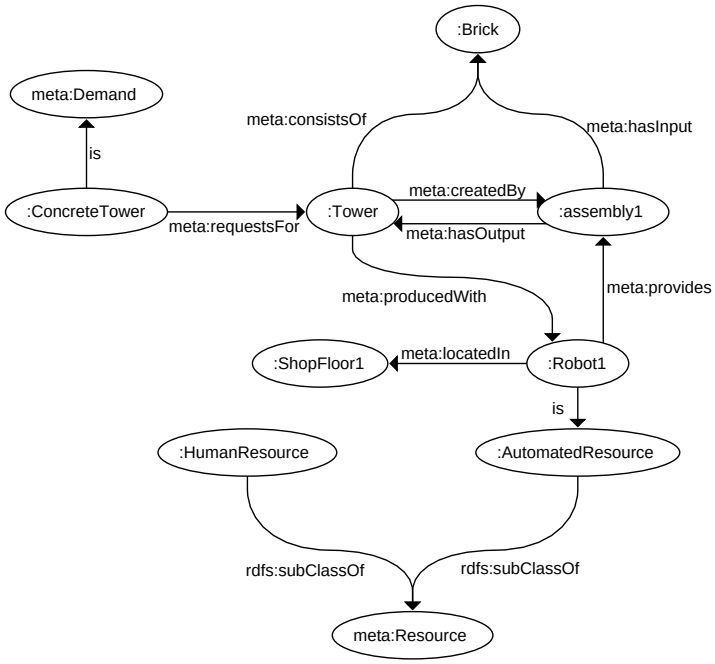


Abbildung 5.2: Visualisierung der grundlegenden RDF-Tripel

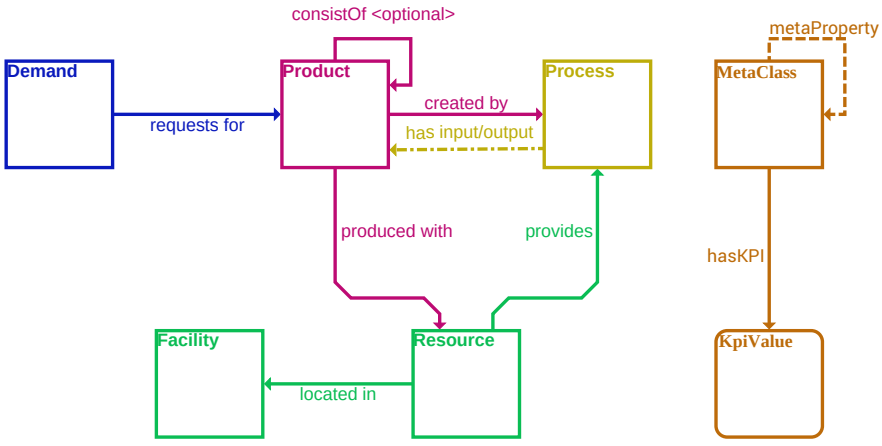


Abbildung 5.3: Erweiterung der Meta-Ontologie auf Multi-KPI-Unterstützung

Kosten werden in Euro angegeben. ":hasCost" und ":takesTime" sind beides Subeigenschaften von der "meta:hasKPI" Eigenschaft und agieren auf der Meta-Klasse der Produkte.

In Abbildung 5.4 wird die Instanziierung der einzelnen Wissensobjekte und ihre Relationen zueinander aufgeführt. Die Instanziierung verhält sich genauso, wie in der Beschreibung des Anwendungsfalles besprochen. Wird eine Anfrage auf ein Objekt gezeigt, in diesem Fall die verstärkte Rumpfhaut. Diese besteht aus zwei Teilen: der Rumpfhaut und der Stringer. Diese Teile werden mit dem ":supply"-Prozess zusammengeführt. Das erste Bauteil kann nur mit einem Prozess erstellt werden. Dieser Prozess dauert 2 Sekunden und kostet 8€. Das zweite Bauteil wird mit zwei Prozessen erstellt. Der erste Prozess benötigt 5 Sekunden und kostet 3€. Der zweite Prozess dauert jedoch 3 Sekunden und kostet 4€. Daher ergibt sich, dass je nach Anfrage andere Prozesskombinationen gewählt werden müssen. Das Grundmaterial ist mit ":atomic1" bezeichnet. Es handelt sich dabei um ein "atomisches" Produkt, d. h. aus Sicht dieses Anwendungsfalles besteht dieses Produkt nicht noch aus weiteren Produkten.

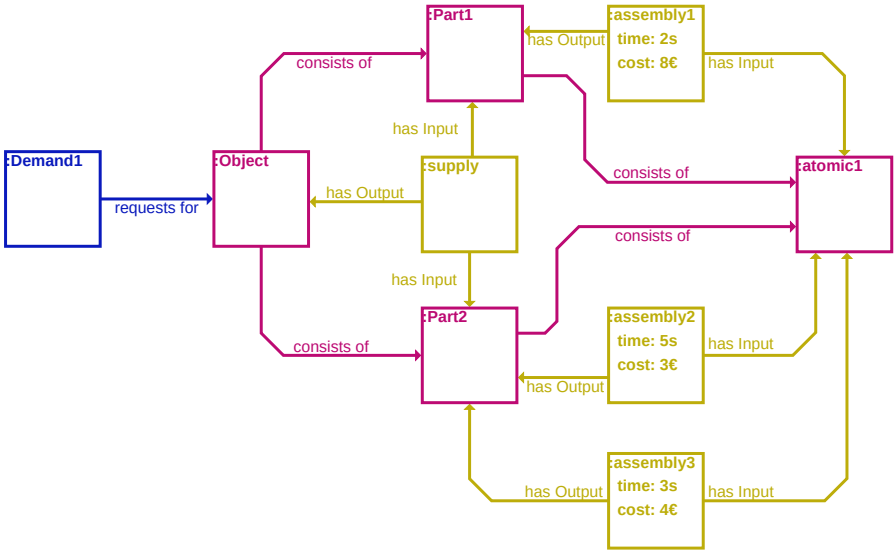


Abbildung 5.4: Wechselwirkungen einzelner Instanzen des zweiten Anwendungsfalles

Diese Wissensobjekte sind dabei nicht direkt von den Basisklassen instanziiert worden. Abbildung 5.5 zeigt, dass die Produkte jeweils in drei Unterklassen aufgeteilt

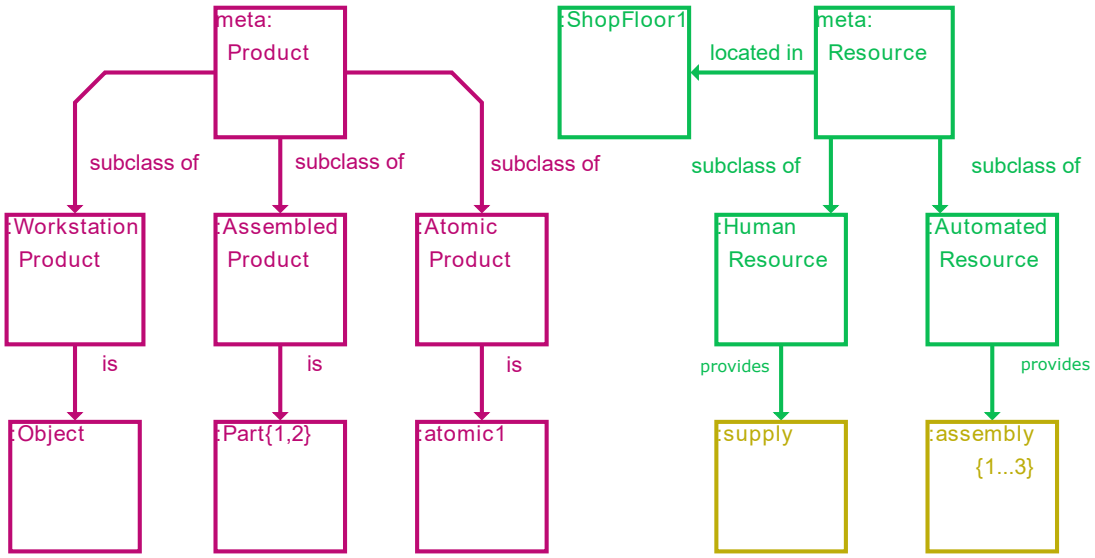


Abbildung 5.5: Subklassenaufbau der Produkte und der Ressourcen innerhalb der Wissensbasis der Instanzen

sind. Ein Produkt kann also entweder ein Produkt der Werkstätte, ein erstelltes Produkt oder ein atomisches Produkt sein. Ein Prozess wird entweder durch eine menschliche Ressource oder eine automatisierte Ressource gestellt. Außerdem befinden sich alle Ressourcen in der gleichen Arbeitstätte.

6 Ergebnisse und Diskussion

Die Anwendungsfälle sind nun aufgestellt. Mit der angewandten Methodik lassen sich, wie es in Kapitel 4 besprochen wurde, die einzelnen Prozesse generieren. Da die Inferenz-Maschine die einzelnen Informationen zu einer logischen Einheit miteinander verbindet und mittels Validierung Inkonsistenzen findet, können mit der erweiterten Datenbank programmatisch die Prozesse abgefragt werden.

6.1 Prozessgenerierung in Tabellen

Der erste Anwendungsfall stellt eine Anfrage auf einen Turm. Nun wird mittels der Inferenz-Maschine die erweiterte Datenbank erstellt und die in Kapitel 4 beschriebenen SPARQL-Anfragen werden ausgeführt. In Abbildung 6.1 wird in der ersten Tabelle das Ergebnis der SPARQL-Abfrage gezeigt. Es stellt sich heraus, dass nach dem Produkt des Turmes gefordert wird. Die dargestellte zweite Tabelle in Abbildung 6.1 zeigt das Ergebnis der Abfrage. Es ist auszulesen, dass zwei Prozesse existieren, mit denen der Turm gebaut werden kann. Beide dieser Prozesse benötigen unterschiedliche Bauteile, daher ergeben sich überhaupt zwei Möglichkeiten.

Der zweite Anwendungsfall stellt eine Anfrage auf eine verstärkte Rumpfhaut. In Abbildung 6.2 wird tabellarisch gezeigt, welche Prozesse erzeugt werden können, wenn die SPARQL-Abfragen aus der vorgestellten Methodik angewandt werden. Dieses Mal wird gezeigt, dass das angefragte Produkt nicht nur aus den drei Bauteilen besteht, sondern dass diese Bauteile auch erstmal erstellt werden müssen. Des Weiteren sieht man, dass das zweite Bauteil mit zwei Prozessen generiert werden kann.

```
#
shape: (1, 2)
```

D	ReqProd
str	str
:ConcreteTower	:Tower

```
--- building :Tower ---
shape: (2, 4)
```

Proc	Resource	Input	Facility
str	str	str	str
:assembly2	:Robot1	:BlueBrick	:ShopFloor1
:assembly1	:Robot1	:RedBrick	:ShopFloor1

```
--- building :BlueBrick ---
nothing to build
--- building :RedBrick ---
nothing to build
```

Abbildung 6.1: Prozessgenerierung mit Varianten-Bildung

6.2 Evaluation nach Hildebrandt

Die aufgestellten Anwendungsfälle werden in diesem Teil genauer auf die aufgestellten Kriterien untersucht. Tabelle 6.1 zeigt auf, welcher Anwendungsfall auf welche Kriterien untersucht wurde. Mit dem ersten Anwendungsfall wird nur die Erweiterbarkeit untersucht. Und nur im zweiten Anwendungsfall wird die Skalierbarkeit und die Multiperspektivität betrachtet. Die anderen Kriterien, also die Inferenzmöglichkeit, die Validierungsmöglichkeit, das Darstellungsformat und die Prozessgenerierung, werden für beide Anwendungsfälle untersucht. Dadurch ergeben sich unterschiedliche Perspektiven für die einzelnen Kriterien.

6.2.1 Skalierbarkeit der Wissensbasis

Der erste Anwendungsfall besteht aus mehr Tripeln als der erste Anwendungsfall. Der Aufbau weist außerdem eine höhere Komplexität auf. Prozesse können immer

```
#
shape: (1, 2)
```

D --- str	ReqProd --- str
:dObject	:Object

```
--- building :Object ---
shape: (2, 4)
```

Proc --- str	Resource --- str	Input --- str	Facility --- str
:supply	:Human1	:Part1	:ShopFloor1
:supply	:Human1	:Part2	:ShopFloor1

```
--- building :Part1 ---
shape: (1, 4)
```

Proc --- str	Resource --- str	Input --- str	Facility --- str
:assembly1	:Robot1	:atomic1	:ShopFloor1

```
--- building :Part2 ---
shape: (2, 4)
```

Proc --- str	Resource --- str	Input --- str	Facility --- str
:assembly2	:Robot1	:atomic1	:ShopFloor1
:assembly3	:Robot1	:atomic1	:ShopFloor1

```
--- building :atomic1 ---
nothing to build
```

Abbildung 6.2: Prozessgenerierung des zweiten Anwendungsfalles

Kategorie	AF1	AF2
Skalierbarkeit		x
Erweiterbarkeit	x	
Multiperspektivität		x
Inferenzmöglichkeit	x	x
Validierungsmöglichkeit	x	x
Darstellungsformat	x	x
Prozessgenerierung	x	x

Tabelle 6.1: Liste der angewendeten Kriterien an den Anwendungsfällen

noch produziert werden. Die EYE Inferenz-Maschine kann weiterhin Prozesse generieren. Da weiterhin Prozesse ausgelesen werden, ist abzulesen, dass eine höhere Komplexität immer noch bearbeitet werden kann. Aber auch für die Inferenz-Maschine bedeutet das mehr Aufwand, da sie mehr berechnen muss und dadurch länger braucht. Die Ontologie ist zwar mit EYE bearbeitbar, aber eine zu hohe Komplexität sorgt für Fehlverhalten der Inferenz-Maschine. Dabei werden Inferenz-Sicherungen manchmal geschlossen, aber das Programm bricht nicht ab. Dieser "leise Fehler" kann verheerende Folgen haben, so können daraus von der SPARQL-Prozedur falsche Daten ausgelesen werden. Da die Wissensbasis daher weitestgehend skalierbar ist, ist dieses Kriterium erfüllt.

6.2.2 Erweiterbarkeit der Meta-Ontologie

Um die Meta-Ontologie im Kontext verschiedener Anwendungsfälle beschreiben zu können, ist die Meta-Ontologie so erstellt worden, dass alle Grundklassen erweitert werden können. So können einzelne Instanzen der gleichen Grundklasse kategorisiert und unterschieden werden. Um jedoch Erweiterbarkeit unterstützen zu können, müssen die Verhältnisse einer erweiterten Basisklasse mit anderen Basisklassen auch in der Unterklasse vertreten sein. Dazu wird Inferenz benötigt.

In dem ersten Anwendungsbeispiel werden verschiedene Arten von Bausteinen benutzt, rote und blaue. Im Kontext der Flugzeugfertigung bedeutet dies entweder

verschieden konstruierte und austauschbare Varianten oder Bauteile mit verschiedenen Materialien. Die Basisklasse, die erweitert wird, ist die Basisklasse der Produkte. Fortan gibt es zwei Bausteine: rote und blaue. Dies sind die Instanzen. Das Objekt ":Brick" ist fortan eine Unterklasse der Basisklasse der Produkte und die rote und blaue Instanz instanziiert von ":Brick". Abbildung 6.3 zeigt die Erweiterung der Ontologie als RDF-Tripel-Visualisierung. Die Aussage, dass ein Turm aus Bausteinen besteht, bleibt weiterhin bei. Dadurch ergibt sich, dass ein Turm aus zwei Varianten bestehen kann. Zusätzlich werden beiden Bausteinen verschiedene Montageprozessen zugeordnet.

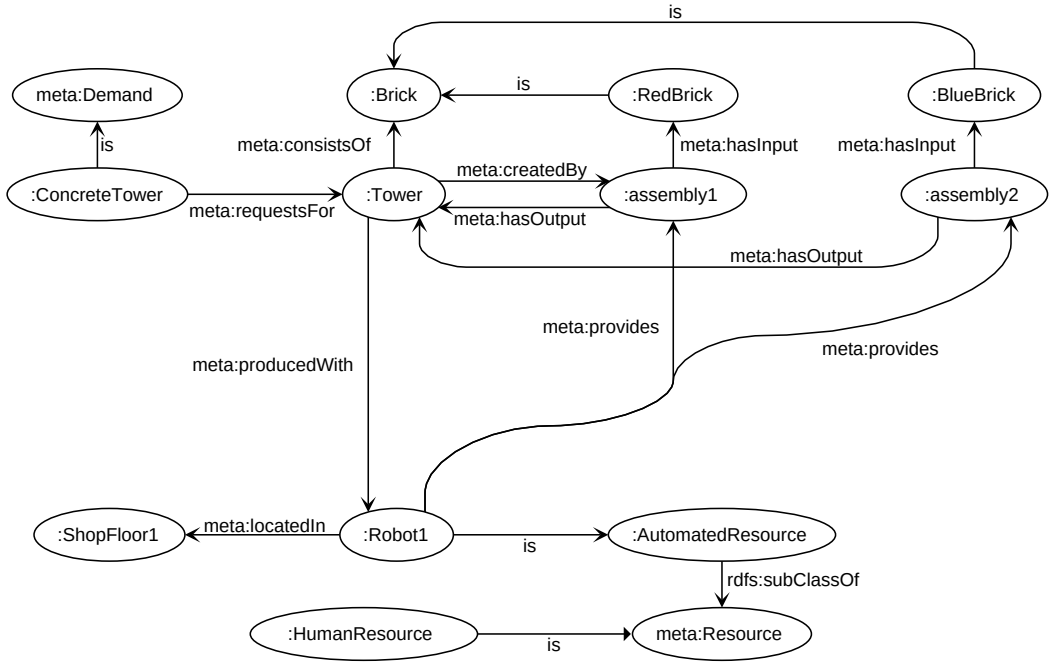


Abbildung 6.3: RDF-Visualisierung der erweiterten Ontologie

Mit der Erweiterung der Meta-Ontologie ist zu beobachten, dass weiterhin Prozese generiert und ausgelesen werden. Daher ist in Abbildung 6.1 zu sehen, dass zwei Prozesse mit jeweils unterschiedlichem Bausteinen aufgeführt werden.

6.2.3 Multiperspektivität durch die Erweiterbarkeit

Aufgrund der Erweiterung der Basisklassen im zweiten Anwendungsfalles ist eine Multiperspektivität ermöglicht. Denn in Kapitel 5 wurde für den zweiten Anwendungsfall beschrieben, wie die Klassen aufgebaut werden. Da anhand dessen in **??** nachvollzogen werden kann, dass daraus Prozesse generiert werden können, können die erweiterten Klassen benutzt werden. Mittels spezifischer SPARQL-Abfragen lassen sich beispielsweise nur montierte Produkte extrahieren. Wie in AOD beschrieben, kann für die Multiperspektivität mehrere Ebenen betrachtet werden. Der Nachteil ergibt sich aus der Flexibilität der Meta-Ontologie. Da die Meta-Ontologie und die Wissensbasis nach dem Anwendungsfall modelliert werden, ist eine Multiperspektivität nicht verpflichtend. So können die Produkte, Ressourcen oder Prozesse im ersten Anwendungsfall nicht in mehreren Betrachtungsebenen aufgeführt werden. Daher ist das Kriterium der Multiperspektivität nicht immer unterstützt und es ergibt sich eine Punktzahl von 0.

6.2.4 Inferenzmöglichkeit des Anwendungsfalles

Abbildung 6.4 zeigt eine angewandte Inferenzregel. Diese Inferenzregel wurde auf der Ebene der Meta-Ontologie definiert und vervollständigt implizite Aussagen über Definitions- und Zielmengen Der Graph $\lambda 0$ findet alle Prädikate, die eine Meta-Eigenschaft sind und sucht nach einem Tripel, welches das Prädikat mit zwei Meta-Klassen benutzt. Dieser Subgraph impliziert Graph $\lambda 1$. Es ist dadurch schon impliziert, dass diese Meta-Eigenschaft als Definitionsmenge das Subjekt als Meta-Klasse hat und dass die Zielmenge das Objekt des Tripels repräsentiert. Diese beiden Aussagen werden von der Inferenz-Maschine der resultierenden Tripel-Menge und demnach auch der erweiterten Datenbank hinzugefügt. Auf der rechten Seite von Abbildung 6.4 ist ein Ausschnitt der Meta-Ontologie aus Abbildung 5.2 gezeigt. Genauso wie es in $\lambda 0$ dargestellt ist, wird ":provides" als Meta-Eigenschaft bestimmt und eine Ressource ":provides" einen Prozess. Daher werden wie in $\lambda 1$ gezeigt zwei neue Tripel hinzugefügt, die die Definitionsmenge auf Ressourcen und die Zielmenge auf

Prozesse bestimmen. Diese beiden Tripel sind mit Grün markiert worden, da sie hinzugefügte Tripel/Verbindungen sind.

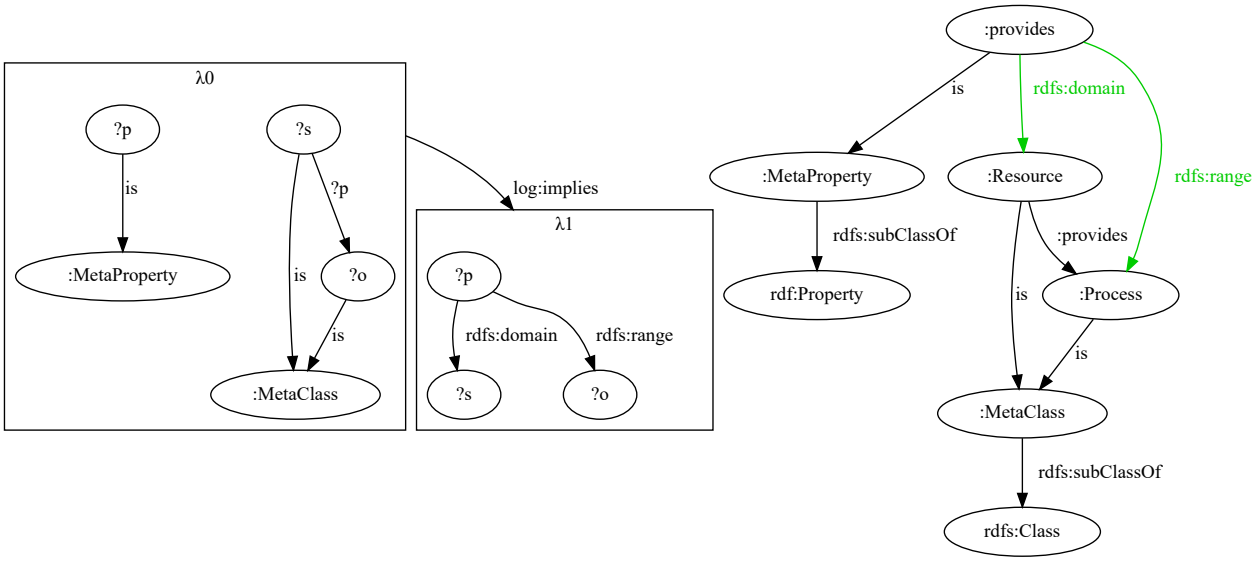


Abbildung 6.4: Anwendung einer Inferenzregel zum Inferieren der Definitions- und Zielmenge innerhalb der Ontologie

Es fällt auf, dass dieser Ausschnitt aus der Ontologie vor dem Hinzufügen der neuen Tripel aus zwei getrennten Teilen bzw. Fragmenten bestand. Zwar wird ":provides" zweimal benutzt, einmal als Subjekt und einmal als Prädikat. Grafisch sind diese Fragmente aber nicht miteinander verbunden. Durch die Inferenz und das Hinzufügen der neuen Tripel werden beide Fragmente zu einem Graphen verbunden. Dies lässt sich dahingegen deuten, dass die Ontologie eine deutlichere Semantik aufweist. Durch eine deutlichere Semantik ergeben sich verschiedene Vorteile. Eines dieser Vorteile ist eine striktere Validierungsmöglichkeit.

6.2.5 Inferenz zum Filtern von Prozessen

Angenommen, die Nachfrage ändert sich, sodass das zweite Bauteil nur mit einem kurzzeitigen Prozess produziert werden kann, unabhängig wie viel der Prozess dauert. Die Methodik sollte in der Lage sein, die relevanten Prozesse anzuzeigen und Prozesse zu ignorieren, die für die Anfrage irrelevant sind. Listing 6.1 zeigt, dass

schon in der Anfrage der Informationsbasis irrelevante Prozesse ausgefiltert werden können. Dabei wird mit der angegebenen N3-Formel Prozesse gesucht, die das zweite Bauteil erzeugen und die länger als 4 Sekunden dauern. Ein zusätzliches Prädikat ":disqualifiedWith" wird dann deduziert. Dieses Prädikat gibt dann an, dass der gefundene Prozess mit der KPI für die Zeit disqualifiziert wird.

```
1 {
2     ?p a meta:Process .
3     ?p meta:hasOutput :Part2 .
4     ?p :takesTime [ math:greaterThan 4 ] .
5 } => {
6     ?p :disqualifiedWith :takesTime .
7 } .
```

Quelltext 6.1: Ausfiltern spezifischer Prozesse mittels N3-Formeln

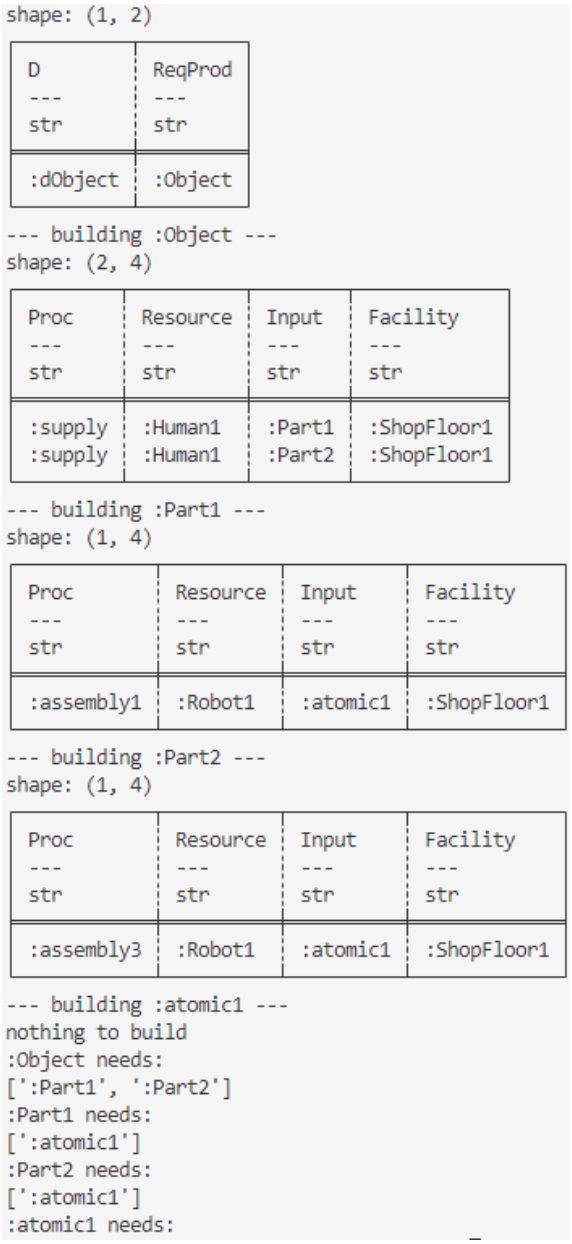
In Listing 6.2 wird die gleiche SPARQL-Abfrage gestellt, wie im ersten Anwendungsfall. Darauf aufbauend wird ein Filter benutzt, um die Ergebnismenge der Resultate zu limitieren. So werden alle Ergebnisse gefiltert, dessen Prozesse disqualifiziert wurden.

```
1  SELECT ?Proc ?Resource ?Facility ?Input
2  WHERE {
3      ?Proc meta:hasOutput {req} .
4      ?Resource meta:provides ?Proc .
5      ?Resource meta:locatedIn ?Facility .
6      ?Proc meta:hasInput ?Input .
7      FILTER(!EXISTS {
8          ?Proc :disqualifiedWith _:x .
9      })
10 }
```

Quelltext 6.2: Abfragen der inferierten Prozesse mit hinzugefügtem Filter

Abbildung 6.5 zeigt die Ausgabe, wenn die Disqualifizierung umgesetzt wurde. Der zweite Fertigungsprozess entfällt hier komplett.

Das am meisten gewichtetet Kriterium, die Inferenzmöglichkeit, ist komplett vertreten. Sowohl in der Meta-Ontologie als auch in der Ebene der Wissensbasis kann Inferenz angewendet werden. Damit kann implizites Wissen aufgeführt werden und



```
shape: (1, 2)
```

D --- str	ReqProd --- str
:dObject	:Object

```
--- building :Object ---
shape: (2, 4)
```

Proc --- str	Resource --- str	Input --- str	Facility --- str
:supply	:Human1	:Part1	:ShopFloor1
:supply	:Human1	:Part2	:ShopFloor1

```
--- building :Part1 ---
shape: (1, 4)
```

Proc --- str	Resource --- str	Input --- str	Facility --- str
:assembly1	:Robot1	:atomic1	:ShopFloor1

```
--- building :Part2 ---
shape: (1, 4)
```

Proc --- str	Resource --- str	Input --- str	Facility --- str
:assembly3	:Robot1	:atomic1	:ShopFloor1

```
--- building :atomic1 ---
nothing to build
:Object needs:
[':Part1', ':Part2']
:Part1 needs:
[':atomic1']
:Part2 needs:
[':atomic1']
:atomic1 needs:
```

Abbildung 6.5: Prozessgenerierung des zweiten Anwendungsfalles mit dem angewendeten Filter

Wissensobjekte werden gefiltert, um eine angepasste Prozessgeneration zu ermöglichen. Daraus ergibt sich, dass die Inferenz unterstützt ist. Dieses Kriterium bekommt daher eine Punktzahl von 1.

6.2.6 Validierung einzelner Wissensobjekte mittels Inferenz

In Abbildung 6.6 wird wieder ein Ausschnitt gezeigt, mit denen die Validierungsmöglichkeit durch die Definition der Meta-Ontologie gezeigt wird. Die Kernaussage in diesem Ausschnitt ist es, dass ein Roboter fehlerhafterweise einen Turm als Prozess zur Verfügung stellt. Ein Turm (*englisch:* Tower) und ein Roboter (*englisch:* Robot) werden von dem Endnutzer instantiiert. Daher sind sie auch in dem Namensraum "inst:" und nicht in dem gleichen Namensraum wie die Meta-Ontologie. Wie im vorigen Beispiel in Abbildung 6.4 steht ":provides" in dem Kontext, dass dieses Prädikat als Definitionsmenge eine Ressource nimmt und als Zielmenge einen Prozess gibt. Diese Informationen sind anders als im vorigen Beispiel schon in der Ontologie mit aufgeführt. Zusätzlich wird die Information mit eingetragen, dass ein Turm ein Produkt ist. Die Inferenz-Maschine startet also mit den schwarzen und der roten Verknüpfung.

Die rechte Formel $\lambda 0$ sucht nach dem Muster. Zum einen wird ein Prädikat gesucht, dass eine Meta-Eigenschaft ist. Zudem werden das Subjekt und das Objekt verbunden, die mit diesem Prädikat verbunden sind und eine Meta-Klasse sind. Daraufhin wird die zweite Formel $\lambda 1$ impliziert. In dieser Formel wird hinzugefügt, dass das Subjekt der Definitionsmenge angehört und dass das Objekt der aus Zielmenge kommt. Wenn diese Formel angewendet wird, ergibt sich die grüne Verbindung, d. h. dass der Turm ein Prozess ist.

Dies stimmt mit der Meta-Ontologie überein, da ":provides" auf Prozesse ausgeübt wird. Daraus entsteht aber eine Kontradiktion. Der Turm ist zugleich ein Prozess und ein Produkt. Das heißt, dass mit der jetzigen Informationsbasis die Möglichkeit bestünde, eine Ressource zu definieren, die einen Turm erstellt, indem der Anwendungsfall durchgeführt wird. Um diesen Widerspruch zu verhindern, ist in der Meta-

Ontologie definiert, dass eine Instanz zugleich nur von einer Meta-Klasse instanziiert werden kann.

Die linke Regel definiert diese Validierung im Kontext der Zielmenge. Wenn ein Tripel mit einer Meta-Eigenschaft als Prädikat gegeben ist, dieses Prädikat eine Zielmenge hat und das Objekt des anfänglichen Tripels von einer Meta-Klasse ungleich der vorherigen instanziiert worden ist, dann ist die Ontologie inkonsistent. Die EYE Inferenz-Maschine benutzt das Konzept einer Inferenz-Sicherung. In einem elektrischen System unterbindet eine Sicherung den Stromkreislauf, um damit verbundene elektrische Abnehmer zu schützen. Ähnlich funktioniert eine Inferenz-Sicherung. Im Falle eines Widerspruches oder einer Inkonsistenz unterbindet diese Sicherung das Finden von weiteren Tripeln mittels Inferenz. Dieser Abbruch verhindert die Weitergabe der Daten an eine Prozedur, die Prozesse über SPARQL und ggf. andere Daten-Entnehmer ausliest.

Die Validierungsmöglichkeit ergibt sich dadurch für die aufgestellte Methodik. Dies liegt unmittelbar an der Inferenz. Daher ergibt sich ebenfalls eine Punktzahl von 1, da dieses Kriterium ebenfalls komplett unterstützt ist.

6.2.7 N3 als Darstellungsformat und Prozessgenerierung

Beide Anwendungsfälle benutzen das N3-Darstellungsformat für RDF-Tripel. Mittels einer Inferenz-Maschine kann über Prolog inferiert werden und über Python aufgerufen und abgefragt werden. Daher ist ein automatisiertes Auslesen der Daten möglich. Prozesse können demnach generiert werden, was auch in Abschnitt 6.1 gezeigt wird. Daher sind beide Kriterien ebenfalls erfüllt. Sie werden beide mit der Punktzahl von 1 bewertet.

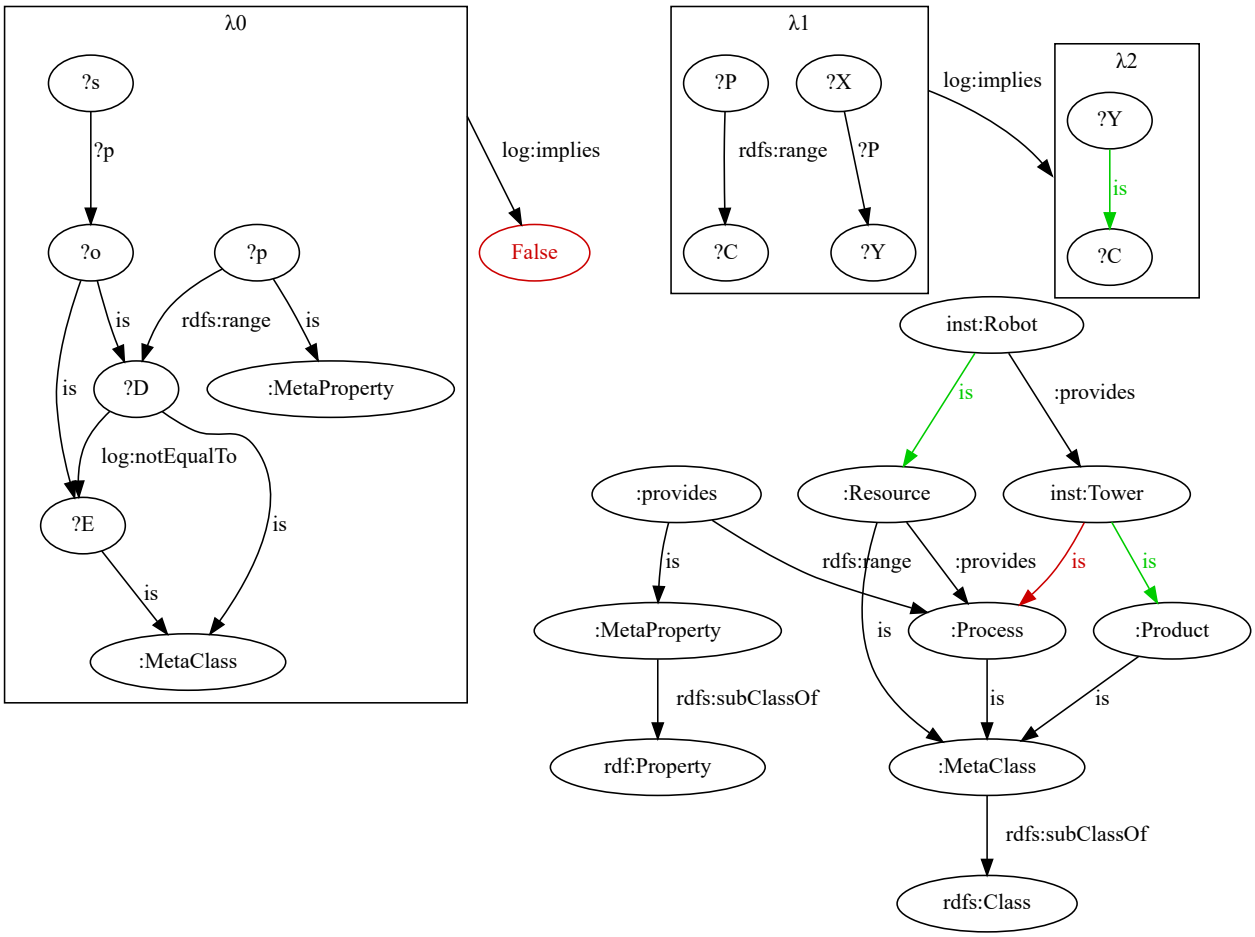


Abbildung 6.6: Validierung der Zielmenge

6.3 Diskussion

In Tabelle 6.2 wird die Punktevergabe zu der erstellten Methodik angezeigt. Jede Kategorie hat eine Punktzahl von 1 bekommen, außer die Multiperspektivität. Diese hat eine Punktzahl von 0 aus den vorher genannten Gründen, dass Multiperspektivität nicht verpflichtend ist und abhängig vom Anwendungsfall unterstützt ist. Daraus ergibt sich eine Summe von 8 Punkten. Dies bedeutet, dass diese Ontologie besser auf den vorgestellte Anwendungsfällen ausgelegt ist als Ontologien aus der Literatur. Da dennoch keine Punktzahl von 9 erreicht wurde, handelt es sich aber nicht um eine ideale Lösung .

Kategorie	Gewichtung	eigene Ontologie
Skalierbarkeit	1	1
Erweiterbarkeit	1	1
Multiperspektivität	1	0
Inferenzmöglichkeit	2	1
Validierungsmöglichkeit	1.5	1
Darstellungsformat	1	1
Prozessgenerierung	1.5	1
Summe (max.: 9)	/	8

Tabelle 6.2: Bewertung der eigen erstellten Ontologie anhand der vorherigen Kriterien aus Kapitel 3

Genauso wie es in [12] beschrieben wurde, liegt die Gefahr beim Beschreiben von Wissensabbildungen in der Komplexität. In einer Ontologie kann die Beschreibung von Wissensobjekten so komplex werden, dass die Verflochtenheit der Wechselwirkungen von Tripeln außer Kontrolle gerät. Wenn sich besonders konzentriert wird auf Inferenz, wie sie in der in Kapitel 2 beschrieben ist, dann kann so eine hohe Komplexität für unvorhersehbare Fehler sorgen. Aus diesem Grund ist die Meta-Ontologie minimal definiert. So können Erweiterungen nach Belieben hinzugefügt werden und verringern ein Risiko auf einer zu großen Komplexität. Durch die Erweiterbarkeit ist also die Inferenzmöglichkeit stärker gesichert.

Nachdem der Anwendungsfall mittels der Methodik angewendet wurde, können Er-

gebnisse ausgelesen werden. Genauso wie CAO, DELS und weitere kann eine skalierbare Ontologie aufgebaut werden. Eine Limitierung besteht nicht, wie viele Instanzen hinzugefügt werden können. Da die Grundklassen und ihre Wechselwirkungen klar definiert sind, ist die Komplexität beim Hinzufügen neuer Instanzen nicht zu hoch. Dies bietet schon jetzt Vorteile gegenüber TOVE.

Genauso wie die anderen genannten Ontologien, besteht die Möglichkeit der Erweiterung. Dazu kann mit dem "rdfs:subClassOf" Prädikat und weiteren eine Grundklasse erweitert werden. Dies erlaubt dem Endnutzer, anwendungsspezifische Subklassen zu definieren und verschiedene Klassen für komplexere Anwendungen zu unterscheiden.

Neben OZONE und DELS, die eine theoretische Inferenzmöglichkeit durch die erstellte Meta-Ontologie bieten, nutzt nur TOVE das Potenzial der Inferenz aus. Genauso wie TOVE, benutzt die vorgestellte Methodik die Prädikatenlogik, um Inferenzen durchzuführen. Dies geschieht über Subgrafen, die in RDF-Tripeln miteinander zu logischen Schlussfolgerungen verbunden sind. Durch zusätzliche Regeln, die im RDFS als Entailments definiert sind, lässt sich implizites Wissen schließen und es werden Informationen geschlossen, die für die Prozessgenerierung gebraucht wird.

In der Literatur besitzt nur die PPR-DSL durch eine DSL die Möglichkeit zur Validierung der aufgestellten Ontologien. Aufgrund der Inferenzmöglichkeit hat sich ergeben, dass die selbsterstellte Ontologie auch in der Lage ist, Inkonsistenzen innerhalb der Ontologie zu finden. Als Beispiel wurde hier eine Instanz in der falschen Basisklasse definiert und die Inferenz-Maschine hat eine Inferenz-Sicherung geschlossen, sodass die Inferenz-Maschine abgebrochen ist. Diese Eigenschaft macht die Ontologie nutzerfreundlicher, da logische Fehler gefunden werden und dem Endnutzer mitgeteilt werden können. Während des Erstellens der Ontologie kann es vorkommen, dass eine Wissensbasis falsch übertragen wird. Mit der Validierungsfähigkeit können vorzeitige Fehler erkannt und dem Endnutzer übergeben werden. Dies gewährleistet eine Benutzerfreundlichkeit, die bis auf bei MAMOT und PPR-DSL nicht sonderlich angesprochen wurden.

Für Darstellungsformat wurde auf die N3-Syntax gewählt. Dies bietet einen Vorteil gegenüber TOVE und PPR-DSL, da ein Format benutzt wird, dass im SW Anwendung findet. Gegenüber AOD, DELS und weiteren bietet es den Vorteil, nicht nur

grafisch zu sein. Daher können auch Computer dieses Format lesen und berechnen. Ein weiterer Vorteil entsteht durch die zusätzliche Nutzung von SPARQL. Die Syntax für einzelne Tripel, Subgrafen und Platzhalter in SPARQL und N3 sind identisch. Dadurch, dass für die gleiche Methodik nicht zwei komplett verschiedene Syntaxen erlernt werden müssen, steigt die Benutzerfreundlichkeit.

Dadurch dass Prozesse generiert werden können, können sie direkt eingesetzt werden in weitere Algorithmen, die Prozesse nehmen. Dazu gehört CAO, denn mit den in CAO vorgestellten Allokationsalgorithmus können die generierten Prozesse optimal auf verschiedene Ressourcen aufgeteilt werden.

Das Kriterium der Multiperspektivität wurde in den verglichenen Ontologien wenig bis ungenügend unterstützt, außer bei AOD. Daher wurde die Ontologie nach AOD im Kriterium der Multiperspektivität modelliert. Die Multiperspektivität ergibt sich aus der Inferenz und Erweiterung der Basisklassen, wodurch unterschiedliche Blickwinkel gebaut werden. Dadurch dass in diesem Anwendungsfall Produkte und Prozesse in unterschiedlichen Klassen zugeordnet werden, kann mit SPARQL-Abfragen nach Teilen der Ontologie gefragt werden, die für die unterschiedlichen Stakeholder von Nutzen sind. So kann für einen Roboterprogrammierer nur wichtig sein, welche montierten Produkte existieren und aus welchen Materialien sie bestehen. Die Nachfrage nach Endprodukten der Arbeitsstationen kann relevant sein, für eine Person, die sich mit der Orchestrierung mehrerer zusammengeführter Produkte oder Architekturen beschäftigt. Daher kann dieses Kriterium für unterschiedliche Anwendungsfälle auch flexibel implementiert werden.

Ferner zeigt der zweite Anwendungsfall, dass es möglich ist, mehrere KPIs an Prozessen anzuhängen. Dies ist aufgrund der Basisklassen-Architektur auch mit anderen Grundklassen der Ontologie möglich. Anhand von KPIs können im vorne herein schon Entscheidungen getroffen werden über relevante Prozesse, Produkte oder Ressourcen. Ein Optimierungsalgorithmus wie dem Allokationsalgorithmus oder andere Algorithmen können dann als Eingabe nicht nur Prozesse und deren Varianten erhalten, sondern auch die unterschiedlichen anwendungsspezifischen KPIs. Dadurch werden dann optimale Prozesskombinationen gefunden, die dann für die eigentliche Fertigung gebraucht werden.

Im Vergleich zu anderen Umgehensweisen zum Aufstellen automatischer Produktionslösungen mittels Wissensbasen, wie in der Arbeit von Hildebrandt, Torsleff, Caesar u. a. [17] oder in DELS nach Sprock, Thiers, McGinnis u. a. [45], bietet diese Arbeit eine praktischere Lösung. Gegenüber der anderen Seite, wie CAO nach Markusheska et al. [40], wird eine eher schwergewichtige Ontologie anstelle einer leichtgewichtigen Ontologie definiert.

7 Zusammenfassung und Ausblick

Diese Arbeit hat untersucht, inwiefern Ontologien und das Semantic Web genutzt werden können, um mittels Wissensbasiertes Engineering Prozesse zu generieren. Dazu wurden als erstes Kriterien aufgestellt, die eine ideale Ontologie im Kontext der Montageprozessplanung bietet, bevor Ontologien aus der Literatur mittels dieser Kriterien miteinander verglichen wurden. Diese Ergebnisse wurden benutzt, um einen Anhalt darüber zu geben, was zum Konstruieren einer eigenen Ontologie benötigt wird und was aus der Literatur genutzt werden kann. So fehlte beispielsweise eine kohärente Inferenz mittels Notation-3-Formeln und eine Validierungsmöglichkeit. Danach wurde eine Methodik zur Prozessgenerierung aufgestellt, die eine Wissensbasis aufbaut und mit einer Inferenz-Maschine und SPARQL-Abfragen die Prozesse schließt und ausliest. Daraufhin wurde eine eigene Meta-Ontologie definiert, welche die Struktur der ganzen Wissensbasis bestimmt. Diese Meta-Ontologie basiert auf dem “Prozesse, Produkte und Ressourcen”-Modell.

Anhand zweier Anwendungsfälle wurde methodisch evaluiert, dass die unterschiedlichen Kriterien in der selbsterstellten Ontologie implementiert sind und mit den anderen Ontologien verglichen wurden. Beide Anwendungsfälle zeigen auf, dass die Kriterien, welche für den Ontologie-Vergleich benutzt wurden, in der selbst erstellten Ontologie vertreten sind. Innerhalb der vorgestellten Methodik erfüllt die Meta-Ontologie die aufgestellten Kriterien, um eine Wissensbasis aufzustellen und innerhalb der KBE-Methodik und der SWT Prozesse aus der Wissensbasis generieren zu können. Sehr schwerwiegende Ontologien wie DELS bieten eine sehr detaillierte Ausführung der Ontologien-Konzeption. TOVE bietet sehr logisch konsistente Aussagen, anhand dessen Prozesse erstellt werden.

Beide dieser Ontologien sind sehr komplex in der Ausführung und benötigen genaue Eingaben mit einer kleinen Fehlertoleranz. Auf der anderen Seite sind Ontologien zu leichtgewichtig. AOD und OZONE bieten unter anderem grafische Visualisierungen von Konzepten, die für die Praxis zu unspezifisch sind. Um aber einen Kontext in die Luftfahrt zu geben und ein Mittelmaß zu finden zwischen schwerwiegenden und leichtgewichtigen Ontologien, ist die erstellte Meta-Ontologie der vorgestellten Methodik eine Lösung. Durch die erstellte Ontologie wird eine Wissensbasis dargestellt, die sogar in der Lage ist, relevante Prozesse auf Basis der aufgestellten Nachfrage nach Produkten zu filtern. Dies ist eine Anwendung um Brain Drain entgegenzuwirken, und Expertenwissen in einer gesammelten Datenbank abzuspeichern.

Die vorgestellte Methodik zeigt, dass es möglich ist, Montageplanung mit Kabinenmodellen zu vereinen, indem Ontologien benutzt werden, um Prozesse zu generieren. Damit ist dieser Teil des digitalen Fadens umgesetzt. Der digitale Faden verbindet aber alle Schritte, die in der Produktion benötigt werden. Daher ist eine Erweiterung der vorgestellten Methode, andere Teile der Produktion zu integrieren. Die in den Grundlagen vorgestellten Algorithmen zur Prozessoptimierung können demnach direkt an der Methodik angehangen werden.

Da Prozesse generiert werden, die für automatisierte Ressourcen geplant sind, ist eine weitere Erweiterung, zu zeigen, inwiefern Robotersysteme mit den Ergebnissen aus der Ontologien-Abfrage automatisiert werden können. Die Prozessdaten, die extrahiert werden können, werden sie direkt in Roboter-spezifische Anweisungen übersetzt oder mittels Inferenz erschlossen. Um aber auf Ereignisse in einer Werkstätte reagieren zu können, muss das aufgebaute KBS dynamisch auf Ereignisse der Werkstätte reagieren können. Dazu muss analysiert werden, inwiefern die EYE Inferenz-Maschine in der Lage ist, dynamisch auf Veränderung in der Wissensbasis reagieren zu können. Das hat den Hintergrund, dass die für die Evaluation aufgestellten Anwendungsfälle statische Wissensbasen behandelt haben. Auf der anderen Seite stellt sich die Frage, ob sich auch automatisch generierte Modelle in die Wissensbasis integrieren lassen. Strukturen der Flugzeugkabine lassen sich schon digital generieren [49]. Daher besteht auch die Möglichkeit diese Modelle in die Ontologie mitzuintegrieren.

Für das Erstellen der Meta-Ontologie wurde RDFS benutzt. Damit wurde die Grundstruktur mittels OOP beschrieben. OWL erfüllt dieselbe Aufgabe. Daher lässt sich OWL auch als Grundpfeiler der Meta-Ontologie benutzen. Die Meta-Ontologie könnte dadurch strikter definiert werden und stärkere Validierungs-Regeln könnten durchgesetzt werden. Dies würde die Nutzerfreundlichkeit verbessern.

Um die Nutzerfreundlichkeit weiterhin zu verbessern, ist es wichtig, die darunterliegenden N3-Formeln zugänglicher zu machen, besonders wenn Subgrafen benutzt werden. Visualisierungswerkzeuge, wie die “dot”-Visualisierung mit gerichteten Grafen ist der erste Schritt in diese Richtung. Eine Nutzerschnittstelle erweist sich in diesem Kontext als hilfreich, um besonders nicht-Experten mit der Ontologie arbeiten zu lassen. MAMOT [5] bietet eine Nutzerschnittstelle als Webanwendung. Solange die Verbindung zum semantischem Netz bleibt, ist die Benutzung einer domänenspezifischen Sprache ebenfalls hilfreich.

Literatur

[1] A. Wenzel, P. Nyhuis, H. Nabizada, L. Beers, A. Fay und M. Röhrig, „Neue Produktionsstrukturen für die Flugzeugfertigung der Zukunft“, in *Beiträge der Helmut-Schmidt-Universität / Universität der Bundeswehr Hamburg: Forschungsaktivitäten im Zentrum für Digitalisierungs- und Technologieforschung der Bundeswehr*, Bd. 1, 2022, S. 196–201. DOI: 10.24405/14551.

[2] M. Fuchs, Y. Ghanjaoui, J. Biedermann und B. Nagel, „An Approach for Linking Heterogenous and Domain-Specific Models to Investigate Cabin System Variants“, in *33rd Annual INCOSE International Symposium*, 2023. Adresse: https://elib.dlr.de/196218/.

[3] L. Beers, M. Weigand, A. Fay und A. Chahine, „Entwicklung eines generischen Modells für die standardisierte Beschreibung von Ressourcen in der Luftfahrtproduktion“, in *Beiträge der Helmut-Schmidt-Universität / Universität der Bundeswehr Hamburg: Forschungsaktivitäten im Zentrum für Digitalisierungs- und Technologieforschung der Bundeswehr*, Bd. 1, 2022, S. 202–208. DOI: 10.24405/14552.

[4] G. La Rocca, „Knowledge based engineering techniques to support aircraft design and optimization“, ISBN: 9789090260693, Dissertation, Technische Universiteit Delft, [S.l.], 0020, 339 S.

[5] J. Page Risueno und B. Nagel, „Development of a Knowledge-Based Engineering Framework for Modeling Aircraft Production“, in *AIAA Aviation 2019 Forum*, (Dallas, Texas), Reston, Virginia: American Institute of Aeronautics and Astronautics, 2019. DOI: 10.2514/6.2019-2889.

[6] DLR e.V. „Programm und Strategie“. (2023), Adresse: https://www.dlr.de/de/forschung-und-transfer/luftfahrt/programm-und-strategie (besucht am 23. 08. 2023).

[7] DLR e.V. „Institut für Systemarchitekturen in der Luftfahrt - DLR - Institut für Systemarchitekturen in der Luftfahrt - Home“. (2023), Adresse: https://www.dlr.de/sl/de/desktopdefault.aspx/tabid-12343/21557_read-49524/ (besucht am 23. 08. 2023).

[8] C. B. Chapman und M. Pinfold, „The application of a knowledge based engineering approach to the rapid design and analysis of an automotive structure“, *Advances in Engineering Software*, Jg. 32, Nr. 12, S. 903–912, 2001, PII: S0965997801000412. DOI: 10.1016/S0965-9978(01)00041-2.

[9] Dominic Singer, „Entwicklung eines Prototyps für den Einsatz von Knowledge-based Engineering in frühen Phasen des Brückenentwurfs“, Masterthesis, Technische Universität München, München, 2014, 121 S.

[10] J. Liebowitz, „Expert systems: A short introduction“, *Engineering Fracture Mechanics*, Jg. 50, Nr. 5-6, S. 601–607, 1995, PII: 0013794494E0047K. DOI: 10.1016/0013-7944(94)E0047-K.

[11] D. Kiritsis, „A Review of Knowledge-Based Expert Systems for Process Planning. Methods and Problems“, in *The International Journal of Advanced Manufacturing Technology*, Bd. 10, Springer-Verlag London Limited, 1995, S. 240–262.

[12] G. Kück, „Tim Berners-Lee's Semantic Web“, *South African Journal of information management*, Jg. 6, Nr. 1, 2004.

[13] C. Bizer, T. Heath und T. Berners-Lee, „Linked data: Principles and state of the art“, in *World wide web conference*, Citeseer, Bd. 1, 2008, S. 40.

[14] T. Berners-Lee. „Semantic Web on XML“. (2000), Adresse: http://www.w3.org/2000/talks/1206-xml2k-tbl/slide1-0. (besucht am 19. 07. 2023).

[15] T. Berners-Lee. „Homepage: Semantic Web Application Platform“. (2019), Adresse: https://www.w3.org/2000/10/swap/ (besucht am 08. 10. 2023).

[16] B. de Meester, D. Arndt, P. Bonte u. a., „Event-Driven Rule-Based Reasoning using EYE“, in *SSN-TC/OrdRing@ ISWC*, 2015, S. 75–86.

[17] C. Hildebrandt, S. Torsleff, B. Caesar und A. Fay, „Ontology Building for Cyber-Physical Systems: A domain expert-centric approach“, in *2018 IEEE 14th International Conference on Automation Science and Engineering, 20-24 August 2018, Munich, Germany*, (Munich, Germany), Institute of Electrical and Electronics Engineers, Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, 2018, S. 1079–1086. DOI: 10.1109/COASE.2018.8560465.

[18] N. Shadbolt, T. Berners-Lee und W. Hall, „The Semantic Web Revisited“, *IEEE Intelligent Systems*, Jg. 21, Nr. 3, S. 96–101, 2006. DOI: 10.1109/MIS.2006.62.

[19] World Wide Web Consortium. „Internetdokumentation: RDF Semantics“. (2004), Adresse: https://www.w3.org/TR/rdf-mt/#RDFSRules (besucht am 12. 07. 2023).

[20] E. R. Gansner, E. Koutsofios, S. C. North und K.-P. Vo, „A technique for drawing directed graphs“, *IEEE Transactions on Software Engineering*, Jg. 19, Nr. 3, S. 214–230, 1993. DOI: 10.1109/32.221135.

[21] E. R. Gansner und S. C. North, „An open graph visualization system and its applications to software engineering“, *Software: Practice and Experience*, Jg. 30, Nr. 11, S. 1203–1233, 2000. DOI: 10.1002/1097-024X(200009)30:11<1203::AID-SPE338>3.0.CO;2-N.

[22] Ecma International. „ECMA-404, The JSON data interchange syntax“. (2017), Adresse: https://www.ecma-international.org/publications-and-standards/standards/ecma-404/ (besucht am 08. 08. 2023).

[23] „Einführung in JSON“. (2023), Adresse: https://www.json.org/json-de.html (besucht am 08. 08. 2023).

[24] J. D. Fernández, M. A. Martínez-Prieto, C. Gutiérrez, A. Polleres und M. Arias, „Binary RDF representation for publication and exchange (HDT)“, *Journal of Web Semantics*, Jg. 19, S. 22–41, 2013, PII: S1570826813000036. DOI: 10.1016/j.websem.2013.01.002.

[25] J. D. Fernández, M. A. Martínez-Prieto, A. Polleres und J. Reindorf, „HDTQ: Managing RDF Datasets in Compressed Space“, in *The Semantic Web*, Ser. Lecture Notes in Computer Science, A. Gangemi, R. Navigli, M.-E. Vidal u. a., Hrsg., Bd. 10843, Cham: Springer International Publishing, 2018, S. 191–208. DOI: 10.1007/978-3-319-93417-4_13.

[26] A. Matono, T. Amagasa, M. Yoshikawa und S. Uemura, „A path-based relational RDF database“, in *Proceedings of the 16th Australasian database conference-Volume 39*, 2005, S. 95–103.

[27] H. Hossayni, I. Khan und C. E. Kaed, „Embedded Semantic Engine for Numerical Time Series Data“, in *2018 Global Internet of Things Summit, 4-7 June 2018, Bilbao, Spain*, (Bilbao), IEEE Communications Society, Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, 2018, S. 1–6. DOI: 10.1109/GIOTS.2018.8534580.

[28] T. R. Gruber, „Toward principles for the design of ontologies used for knowledge sharing?“, *International journal of human-computer studies*, Jg. 43, Nr. 5-6, S. 907–928, 1995.

[29] J. A. Robinson, *Software design for engineers and scientists*. Amsterdam: Newnes und Elsevier Science & Technology, 2004. DOI: 10.1016/B978-0-7506-6080-8.X5000-2.

[30] Oracle. „Interfaces, Java Documentation“. (2023), Adresse: https://docs.oracle.com/javase/tutorial/java/IandI/createinterface.html (besucht am 08. 08. 2023).

[31] cppreference.com. „Abstract class“. (2023), Adresse: https://en.cppreference.com/w/cpp/language/abstract_class (besucht am 08. 08. 2023).

[32] J. Bezivin und O. Gerbe, „Towards a precise definition of the OMG/MDA framework“, in *Proceedings / 16th Annual International Conference on Automated Software Engineering (ASE 2001), 26 - 29 November 2001, Loew's Coronado Bay Resort, Coronado Island, San Diego, California*, (San Diego, CA, USA), Los Alamitos, Calif. u.a.: IEEE Computer Society, 2001, S. 273–280. DOI: 10.1109/ASE.2001.989813.

[33] World Wide Web Consortium. „OWL 2 Web Ontology Language Document Overview (Second Edition)“. (2012), Adresse: https://www.w3.org/TR/owl2-overview/ (besucht am 23. 08. 2023).

[34] L. M. Dan Brickley. „FOAF Vocabulary Specification“. (2004), Adresse: http://xmlns.com/foaf/0.1/ (besucht am 23. 08. 2023).

[35] World Wide Web Consortium. „SPARQL 1.1 Overview“. (2013), Adresse: https://www.w3.org/TR/2013/REC-sparql11-overview-20130321/ (besucht am 23. 08. 2023).

[36] O. Hirschbach, „Rechnerunterstützte Montageplanerstellung“, Zugl.: Stuttgart, Univ., Diss., 1978, Mainz, 1978. Adresse: https://permalink.obvsg.at/AC00316733.

[37] H. Rudolf, „Wissensbasierte Montageplanung in der digitalen Fabrik am Beispiel der Automobilindustrie“, Dissertation, Technische Universität München, München, 2006.

[38] C. Blum, „Beam-ACO—hybridizing ant colony optimization with beam search: an application to open shop scheduling“, *Computers & Operations Research*, Jg. 32, Nr. 6, S. 1565–1591, 2005, PII: S0305054803003599. DOI: 10.1016/j.cor.2003.11.018.

[39] A. Konak, D. W. Coit und A. E. Smith, „Multi-objective optimization using genetic algorithms: A tutorial“, *Reliability Engineering & System Safety*, Jg. 91, Nr. 9, S. 992–1007, 2006, PII: S0951832005002012. DOI: 10.1016/j.ress.2005.11.018.

[40] N. Markusheska, V. Srinivasan, J.-N. Walther u. a., „Implementing a system architecture model for automated aircraft cabin assembly processes“, *CEAS Aeronautical Journal*, Jg. 13, Nr. 3, S. 689–703, 2022, PII: 582. DOI: 10.1007/s13272-022-00582-6.

[41] M. S. Fox, M. Barbuceanu und M. Gruninger, „An organisation ontology for enterprise modelling: preliminary concepts for linking structure and behaviour“, in *Proceedings 4th IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE '95)*, (Berkeley Springs, WV, USA), IEEE Comput. Soc. Press, 1995, S. 71–81. DOI: 10.1109/ENABL.1995.484550.

[42] Richard A. Wysk und Jeffrey S. Smith, „A formal functional characterization of shop floor control“, *Computers & Industrial Engineering*, Jg. 28, Nr. 3, S. 631–643, 1995. DOI: 10.1016/0360-8352(94)00216-A.

[43] S. F. Smith und M. A. Becker, „An Ontology for Constructing Scheduling Systems“, *Working Notes of 1997 AAAI Symposium on Ontological Engineering*, S. 120–127, 1997.

[44] S. Lemaignan, A. Siadat, J.-Y. Dantan und A. Semenenko, „MASON: A Proposal For An Ontology Of Manufacturing Domain“, in *DIS 2006 : IEEE Workshop on Distributed Intelligent Systems, Collective Intelligence and its Applications : proceedings : June 15-16, 2006, Prague, Czech Republic*, (Prague, Czech Republic), V. Maérâik und V. Marík, Hrsg., IEEE Systems, Man, and Cybernetics Society und IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and its Applications, Place of publication not identified: IEEE Computer Society, 2006, S. 195–200. DOI: 10.1109/DIS.2006.48.

[45] T. Sprock, G. Thiers, L. F. McGinnis und C. Bock, *Theory of Discrete Event Logistics Systems (DELS) specification*, Gaithersburg, MD, 2020. DOI: 10.6028/NIST.IR.8262.

[46] K. Meixner, F. Rinker, H. Marcher, J. Decker und S. Biffl, „A Domain-Specific Language for Product-Process-Resource Modeling“, in *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, (Vasteras, Sweden), IEEE, 2021, S. 1–8. DOI: 10.1109/ETFA45728.2021.9613674.

[47] D. Arndt und S. Mennicke, *Notation3 as an Existential Rule Language*, 2023. DOI: 10.48550/arXiv.2308.07332. Adresse: http://arxiv.org/pdf/2308.07332v1.

[48] C. Hildebrandt, „Engineering ontologischer Modelle in der Automatisierungstechnik“, Dissertation, Universitätsbibliothek der HSU/UniBwH, 2021.

[49] R. D. Dewald, T. Klimmek, S. Algermissen, C. Hesse und R. Winter, „Angepasste Modellierungsvorschriften für vibroakustische Untersuchungen von Flugzeugrümpfen“, in *49. Jahrestagung für Akustik DAGA 2023*, April. Adresse: https://elib.dlr.de/194901/.