Ähnlichkeitsmaße für ontologie-basierte Produktkonfigurationen

B A C H E L O R A R B E I T

zur Erlangung des akademischen Grades

Bachelor of Science (B. Sc.)

im Studiengang Angewandte Informatik

FRIEDRICH-SCHILLER-UNIVERSITÄT JENA
Fakultät für Mathematik und Informatik

eingereicht von Niklas Baumbach

geb. am 08.02.1998 in Rudolstadt

Universitäre Betreuerin: Prof. Dr. Birgitta König-Ries
Betrieblicher Betreuer: Lars Wagner (Jena Optronik GmbH)
Betriebliche Betreuerin: Diana Peters (DLR e.V.)

Kurzfassung

In der Raumfahrt sind Instrumente und Sensoren je nach Mission sehr unterschiedlichen Bedingungen ausgesetzt. Daher bieten die Hersteller häufig konfigurierbare Produkte an, um den Anforderungen ihrer Kunden gerecht werden zu können. Ein Beispiel für so ein Produkt ist der Sternsensor ASTRO APS der Jena-Optronik GmbH (JOP). Eine stark vereinfachte Version dieses Geräts wird in der Arbeit betrachtet. Die Ermittlung von passenden bestehenden Konfigurationen zu einer Kundenanfrage erfolgt meist per Hand durch die Mitarbeiter und sollte durch ein automatisiertes Verfahren deutlich zu vereinfachen sein.

Das Ziel der Arbeit ist ein praktisch nutzbarer Ansatz zur Bestimmung der Ähnlichkeit solcher Produktkonfigurationen. Dafür wird eine formale Beschreibung des Produkts benötigt. Diese erfolgt durch eine Ontologie in der Sprache OWL, welche durch Konstrukte aus der Validierungssprache SHACL um technische Einschränkungen erweitert wird. Zum anderen muss ein passendes Ähnlichkeitsmaß ausgewählt werden, um zu Kundenanfragen die bestmöglichen Konfigurationen zu finden. Dazu werden ein mengen-basierter, ein probabilistischer und ein graphen-basierter Ansatz in der Programmiersprache Java implementiert. Letzterer wird dabei unter Rücksprache mit den Experten von JOP selbst entworfen.

Die Eignung der Ontologie wird durch die Beantwortung von vorher definierten Kompetenzfragen geklärt. Für die Auswahl eines passenden Ähnlichkeitsmaßes werden interessante Beispielanfragen ermittelt und die Ergebnisse der einzelnen Methoden zu diesen durch die Experten von JOP bewertet.

Dabei stellt sich heraus, dass das selbst entwickelte, graphen-basierte Ähnlichkeitsmaß die beste Bewertung erhält und einen guten Ausgangspunkt für die praktische Anwendung darstellt. Weiterführende Forschung könnte sich mit der vollständigen Beschreibung eines Produkts/einer Produktart nach internationalen technischen Standards oder der Entwicklung eines Anwendungsprogramms zur Bestimmung von passenden Produktkonfigurationen auf Herstellerseite beschäftigen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Auflistungsverzeichnis

1 Einleitung

1.1 Motivation

In vielen Industriezweigen hängen die Anforderungen an ein Produkt vom Einsatzzweck ab. Insbesondere im Bereich der Raumfahrt sind die technischen und physikalischen Anforderungen an die verschiedenen Bauteile eines Satelliten stark von den Gegebenheiten der jeweiligen Mission abhängig. Daher müssen die Hersteller in diesem Bereich vielseitig konfigurierbare Produkte anbieten, um den spezifischen Anfragen ihrer Kunden möglichst gut nachkommen zu können.
Die Jena-Optronik GmbH (JOP) ist ein Thüringer Raumfahrtunternehmen, welches sich auf die Entwicklung und Fertigung unterschiedlicher optischer Instrumente und Sensoren spezialisiert hat. Zu den erfolgreichsten und weltweit eingesetzten Produkten zählen Lageregelungssensoren, die zur präzisen Ausrichtung von Sonden und Satelliten benötigt werden. Eine Art dieser Sensoren sind die sogenannten Sternsensoren, welche die Orientierung durch den Abgleich von Bilddaten mit integrierten Sternkatalogen bestimmen. In diesem Bereich bietet JOP sehr unterschiedliche Modelle an und entwickelt weitere. [1, 2].
Da die Anforderungen an die Sensoren sehr von den Gegebenheiten der jeweiligen Mission (Strahlungsanforderung, Lebensdauer, Temperaturen, etc.) und dem verwendeten Satellitenbus (Kommunikationsschnittstelle, Spannungsversorgung, etc.) abhängen, können die Sensoren unterschiedlich konfiguriert werden. Der Kunden stellt daher eine Anfrage mit den geforderten Spezifikationen an JOP. Beispielsweise muss ein Sternsensor im erdnahen Orbit weniger strahlungsfest sein, als ein anderer in einem höheren. Allerdings braucht er wahrscheinlich eine höhere Updaterate, da seine Geschwindigkeit größer ist und deshalb häufiger Aktualisierungen vorgenommen werden müssen. Zusätzlich sollte das Gerät natürlich möglichst leicht und kosteneffizient sein. Deshalb stehen bei der Konfiguration eines Sternsensors verschiedene Varianten einer Komponenten zu Verfügung, aus denen das Gerät aufgebaut ist. So existieren verschiedene Detektoren, Spannungswandler und Hauptplatinen (Prozessierung und Kommunikation). Allerdings sind nicht alle Bauteile untereinander kompatibel oder

die Kombination würde großen zeitlichen und finanziellen Aufwand erfordern.
Aufgrund der enormen Anforderungen an die Geräte müssen neue Konfigurationen umfangreiche Testkampagnen mit Schock, Vibration, Thermalvakuumkammer, EMC/ESD und weiteren Funktionstests durchlaufen. Zusätzlich sind umfangreiche Analysen notwendig, die beispielsweise die Strahlungseigenschaften untersuchen. Grundlage dieser Analysen sind häufig spezielle Strahlungstests auf Bauteilebene. Bei diesen wird die kosmische Strahlung, welche um ein Vielfaches stärker ist als auf der Erde, simuliert. Da diese Tests sehr zeitaufwändig und teuer sind, ist die Produktion einer bereits qualifizierten Konfiguration deutlich günstiger. In manchen Fällen könnte es daher sinnvoller sein, dem Kunden ein bereits qualifiziertes Produkt mit ähnlichen Spezifikationen anzubieten, welches trotzdem den Anforderungen gerecht werden kann.
Aktuell erfolgt die Auswahl von solchen potentiell ähnlichen Konfigurationen durch die Mitarbeiter von JOP anhand von verschiedenen Quellen wie Dokumenten, Excel-Tabellen und ihren persönlichen Erfahrungen. Dieser Prozess ist relativ ineffizient und sollte durch eine geeignete formale Beschreibung der Konfigurationen zusammen mit dem nötigen Expertenwissen automatisiert werden können.

1.2 Aufgabe

Im Rahmen dieser Bachelorarbeit soll eine Ontologie zur Beschreibung von Konfigurationsmöglichkeiten eines Produkts erstellt werden. Dabei sollen insbesondere Einschränkungen, welche bestimmte Konfigurationsoptionen auf die zulässigen Möglichkeiten anderer Eigenschaften haben, abgebildet und validiert werden können. Für zulässige Konfigurationen soll außerdem eine weitere Unterteilung in technisch mögliche, existierende oder bereits qualifizierte möglich sein. Konkret implementiert werden soll ein Modell für die Konfigurationen des Sternsensors ASTRO APS[3] von JOP.
Anschließend werden drei verschiedene Ähnlichkeitsmaße ausgewählt und implementiert, mit deren Hilfe möglichst ähnliche Konfigurationen des Sternsensors automatisch gefunden werden sollen. Bei der Auswahl der Funktionen wurde darauf geachtet, dass sie verschiedene Ansätze nutzen und unterschiedliche Komplexität besitzen, um

einen möglichst guten Überblick zu erhalten. Die Auswahl der Testdaten und die Evaluation der Ergebnisse erfolgt durch die Experten von JOP. Am Ende soll ein praktisch verwertbarer Ansatz für die Bestimmung von ähnlichen Konfigurationen vorliegen.
Unterstützt wird die Arbeit vom Institut für Datenwissenschaften Jena des Deutschen Zentrums für Luft- und Raumfahrt e.V., welches unter anderem an Technologien des Semantic Web forscht, und der Jena-Optronik GmbH, die fachspezifische Informationen bereitstellt und die Evaluation unterstützt.

2 Stand der Forschung

Dieses Kapitel gibt einen Überblick über die zentralen Themenfelder der Arbeit. Das sind zum einen Ontologien, um das Konfigurationswissen formal darzustellen, und zum anderen Ähnlichkeitsmaße, welche für die automatische Bestimmung von passenden Konfigurationen zu Kundenanfragen benötigt werden. Dabei wird der Bezug zu relevanten Veröffentlichungen hergestellt. Im letzten Abschnitt werden die bei der Implementierung verwendeten Technologien und Spezifikationen aufgeführt und kurz erläutert.

2.1 Ontologien

In der Informatik bezeichnet eine Ontologie ein explizites, formales Modell eines bestimmten Sachverhaltes [4]. Dieses besteht aus relevanten Begriffen oder Konzepten und Relationen zwischen diesen, welche sich aus der Beobachtung der Thematik ergeben. Die Konzepte sind dabei meist in einer Abstraktions- bzw. Verallgemeinerungshierarchie angeordnet, sie bilden also eine Taxonomie. Eine Ontologie über Universitäten könnte beispielsweise das allgemeine Konzept Person beinhalten, welches sich in Studenten, Dozenten und Verwaltungsangestellte unterteilen lässt. Studenten könnten dann wiederum durch Bachelor- und Masterstudenten spezialisiert werden. Der Vorteil von Ontologien liegt in ihrer großen Ausdrucksmächtigkeit und strikten Formalisierung, welche es erlaubt das spezifizierte „Wissen“ in digitalisierter Form zu speichern, zu verarbeiten und auszutauschen. Mit Hilfe von sogenannten Inferenz- und Integritätsregeln können außerdem neue Zusammenhänge aus den bereits bestehenden Informationen abgeleitet werden. Die dafür verwendeten Programme nennt man Reasoner [5].
Von diesen steht bereits eine Vielzahl zur Verfügung, zu deren bekanntesten Pellet[6], FaCT++[7] und HermiT[8] zählen. Zur Darstellung von Ontologien existieren unterschiedliche Sprachen, wie F-Logic[9], RDF-Schema (RDFS) [10], Simple Knowledge Organisation System (SKOS) [11] und die Web Ontology Language (OWL) [12]. Als einer der bekanntesten Editoren hat sich das Programm „Protégé“[13] der Stanford University etabliert.

Mit dem Problem der Darstellung von konfigurierbaren Produkten durch Ontologien haben sich schon mehrere Forschungsarbeiten befasst. Bereits 1998 stellten Soininen et al. in ihrem Paper „Towards a general ontology of configuration“ eine Ontologie zur Abbildung von Wissen über Konfigurationen vor. Sie unterscheiden Wissen über Konfigurationen dort in Konfigurationsmodellwissen (configuration model knowledge) und Konfigurationslösungswissen (configuration solution knowledge). Ersteres spezifiziert die Produkte, ihre Eigenschaften und die Möglichkeiten wie diese kombiniert werden können, während das Zweite explizite Instanzen enthält, welche in der realen Welt existieren können. Da in der Arbeit keine formale Sprache zur Beschreibung der Ontologie genutzt wird, werden auch alle Einschränkungen durch selbst definierte Klassen dargestellt.[14]
Nach der zunehmenden Standardisierung der Semantic Web Technologien durch die Spezifikationen des W3C und der daraus resultierenden Verbreitung von OWL und anderer formaler Sprachen zur Wissensrepräsentation, hat sich ein Teil der Forschung mit der Verwendung und Erweiterung dieser Standards zur Beschreibung von Konfigurationsoptionen befasst. Ein großes Problem ist dabei die „Open World Assumption“ (OWA), welche OWL zugrunde liegt, und die Formulierung von Integritätsbeschränkungen erschwert. Deshalb beschäftigten sich bereits eine Vielzahl von Veröffentlichungen [15–18] damit, wie man OWL um Integritätsbeschränkungen erweitern kann oder schlugen auf OWL basierende Sprachen vor, welche diese unterstützen [19].
In ihrer Publikation „Methodology for an Ontology-Driven Product Configuration Process“ [20] zeigen Bergner, Bartelt, Bergner und Rausch einen Ansatz zur Erstellung von Ontologien für Produktkonfigurationen, welcher auf der Verwendung von SPIN [21] basiert. Bei SPIN handelt es sich um die sogenannte SPARQL Inferencing Notation der Firma TopQuadrant, welche die Erstellung von Integritätsbeschränkungen durch SPARQL-Abfragen ermöglicht. Basierend auf dieser Sprache wurde 2017 die Shapes Constraint Language (SHACL) [22] als W3C-Standard veröffentlicht, die in dieser Arbeit als Grundlage zur Formulierung von Integritätsbedingungen dient.

2.2 Ähnlichkeitsmaße

Bereits seit vielen Jahrhunderten beschäftigen sich sowohl die Mathematik als auch die Philosophie mit dem Begriff der Ähnlichkeit. Während sich die antiken Philosophen mit der Ähnlichkeit von geometrischen Formen befassten, ist der 1902 von Paul Jaccard formulierte Jaccard-Koeffizient eines der ersten Maße[1] für die Bestimmung der Ähnlichkeit zweier Mengen, welches sich bis heute etabliert hat [23]. Zur Berechnung teilt man die Anzahl der gemeinsamen Elemente durch die Anzahl der Gesamtelemente [24].

Ähnlichkeitsmaße werden heute im Bereich der multivariaten Statistik untersucht und vor allem in der Datenanalyse (Clustering, Duplikaterkennung), der Informationsverarbeitung (Information Retrieval, Schema Matching), in der Bioinformatik (Sequencealignment) und in der Psychologie (Vergleich empirischer Phänomene) verwendet. Auch wenn, soweit ich weiß, keine exakte Definition existiert, handelt es sich um eine reellwertige Funktion, deren Funktionswert umso größer ist, je ähnlicher sich die Objekte sind. Oftmals wird der Wertebereich auf das Intervall [0;1] eingegrenzt, indem für gleiche Objekte der Wert 1 und für solche ohne jegliche Ähnlichkeit der Wert 0 gefordert wird. Da einige Ähnlichkeitsmaße nur zwischen gleich (Wert 1) und ungleich (0) unterscheiden, können diese auch über eine boolesche Funktion beschrieben werden. Außerdem stehen Ähnlichkeitsmaße für metrisch skalierte Werte in engem Zusammenhang mit Distanzmaßen, da sie sich umgekehrt zu diesen verhalten.[23, 25]

Um die Ähnlichkeit von gemischten numerischen und kategorischen Datenmengen vergleichen zu können, haben sich viele Arbeiten vor allem im Bereich der Datenanalyse und des Clusterings in den letzten 20 Jahren mit Distanz- und Ähnlichkeitsmaßen für diese beschäftigt. Aufbauend auf der K-Modes[26] genannten Anpassung des K-Means Clustering-Algorithmus für kategorische Daten, sind in den darauffolgenden Jahren weitere Ansätze oder Erweiterungen zum Clustern von gemischten Datensätzen erschienen, beispielsweise [27–30].

Abschließend muss noch auf semantische Ähnlichkeitsmaße eingegangen werden.

[1] Hierbei handelt es sich nicht um mein Maß im streng mathematischen Sinn der Maßtheorie. Im Kontext von Ähnlichkeitsmaßen bezieht sich der Begriff lediglich auf die Messung einer Größe.

Diese können in zwei Gruppen unterteilt werden. Dabei handelt es sich zum einen um verteilungsbasierte semantische Ähnlichkeitsmaße ("distributional semantic measures"), welche unter der Annahme, dass semantisch ähnliche Wörter oft gemeinsam auftauchen, und Mitteln der Textanalyse zum Vergleich von Begriffen und Dokumenten eingesetzt wird. Andererseits versteht man darunter auch wissensbasierte semantische Ähnlichkeitsmaße ("knowledge-based semantic measures"). Diese nutzen die Struktur der Taxonomie und der Relationen einer Ontologie oder eines anderen Wissensrepräsentationssystems, um die Ähnlichkeit von Klassen oder Instanzen zu bestimmen [31]. Einige Beispiel für solche Verfahren sind [32, 33]. Auch das dritte in dieser Arbeit betrachtete Ähnlichkeitsmaß (siehe 5.3) kann im weiteren Sinn dazu gezählt werden.

2.3 Verwendete Technologien und Spezifikationen

Dieser Abschnitt gibt einen Überblick über die bei der Implementierung relevanten Technologien und Spezifikationen und erläutert jeweils kurz, warum gerade diese verwendet wurden.

2.3.1 OWL

Die Web Ontology Language (OWL) ist eine formale Sprache zur Erstellung von Ontologien, welche 2004 als Spezifikation des W3C veröffentlicht wurde und mittlerweile in der zweiten Version vorliegt. Außerdem ist sie Teil des Semantic Web und dient dort zur Definition des Vokabulars[34]. OWL basiert auf RDF-Schema[10] (Entitäten werden durch eine URI eindeutig identifiziert und alle Aussagen werden in Form von Tripeln aus Subjekt - Prädikat - Objekt gemacht) und erweitert dieses um viele weitere Konstrukte, welche die Ausdrucksmächtigkeit erhöhen.
Dabei wird zwischen Klassen, Instanzen und Eigenschaften unterschieden. Klassen repräsentieren abstrakte Konzepte und können eine Vererbungshierarchie bilden und/oder durch Mengenoperationen wie Vereinigung und Durchschnitt gebildet werden. Instanzen sind Individuen von einer oder mehreren Klassen. Eigenschaften sind binäre Relationen, die den Instanzen entweder Literale („DatatypeProperty“) oder andere Instanzen („ObjectProperty“) zuordnen. Besonders hervorzuheben sind

außerdem die sogenannten „BNodes“ (blank nodes), bei denen es sich um Entitäten handelt, welche keine URI haben und somit nach ihrer Definition nicht wieder referenziert werden können. Sie finden vor allem Verwendung bei der Darstellung von Restriktionen, Listen oder anderen Konstrukten, die keinen expliziten Begriff beschreiben.
OWL wird in drei verschiedene Sprachebenen unterteilt: Lite, DL und Full. OWL Full enthält alle Sprachkonstrukte und ist damit am ausdrucksstärksten, allerdings sind die Ontologien hier unentscheidbar und können deshalb nicht von einem Reasoner verarbeitet werden. Bei OWL DL wurden verschiedene Einschränkungen gemacht, um eine Abbildung auf die Beschreibungslogik SHOIN(D) und damit Entscheidbarkeit zu ermöglichen. OWL Lite ist eine vereinfachte Untermenge von OWL DL, die vor allem zur Erstellung von Taxonomien und einfachen Ontologien gedacht ist.[12]
Da bereits Erfahrung mit OWL Lite vorlagen, die Sprachebene ausreichend für den Anwendungsfall und durch einen Reasoner entscheidbar ist, wurde sich für OWL Lite zur Beschreibung des Sternsensors entschieden. Im Folgenden ist mit OWL deswegen immer die Sprachebene OWL Lite gemeint.

2.3.2 SHACL

Die Shapes Constraint Language (SHACL) ist eine Sprache zur Validierung von RDF-Graphen durch eine Menge von Bedingungen. Der W3C Standard ist relativ neu und wurde 2017 veröffentlicht. Die Anforderungen werden durch sogenannte Formen („shapes“) dargestellt und bilden ebenfalls einen RDF-Graph, welcher Formen-Graph („shapes graph“) genannt wird. Die zu prüfende RDF-Menge heißt Daten-Graph („data graph“). Eine große Stärke von SHACL gegenüber ähnlichen Sprachen wie ShEx[12] sind die Validierungsberichte („validation reports“), welche genauere Informationen liefern, ob und gegebenenfalls weshalb die Überprüfung fehlgeschlagen ist. Eine Form besteht dabei grundlegend aus der Definition von Fokusknoten („focus node“), beispielsweise alle Instanzen einer Klasse, eine Menge bestimmter Instanzen oder alle Objekte eines Triples mit einem bestimmten Prädikat, und einer beliebigen Anzahl an Einschränkungskomponenten („constraint components“). In der einfachen Spezifikation SHACL Core gibt es eine Vielzahl von Einschränkungskomponenten,

um zum Beispiel die Kardinalität von Eigenschaften, deren Wertebereich oder die Länge eines String einzuschränken. Außerdem können Formen geschachtelt werden, wodurch auch mit den Fokusknoten über bestimmte Relationen verbundene Knoten im RDF-Graph erreicht und validiert werden können. Beispielsweise könnte die Anzahl von E-Mailadressen, die ein Mitarbeiter besitzt, auf maximal eine beschränkt werden und durch eine weitere Form sichergestellt werden, dass alle E-Mailadressen einem bestimmten Muster entsprechen. Durch die erweiterte Spezifikation SHACL-SPARQL ist es möglich SPARQL-basierte Bedingungen (siehe 2.3.3) und eigene Validierungsberichte auf Grundlage der in der Anfrage verwendeten Variablen zu definieren. Die SPARQL-Abfrage ist dabei so aufgebaut, dass sie alle ungültigen Instanzen zurückgibt. Der Standard definiert sogar eine Möglichkeit um eigene Einschränkungskomponenten mit Hilfe von SPARQL zu definieren.[22]
Da bereits Erfahrungen mit SHACL vorhanden waren und die Sprache genau für diesen Zweck entworfen wurde, war die Wahl von SHACL um die Einschränkung der gültigen Konfigurationen zu modellieren recht eindeutig. Zur Validierung wird die Implementierung TopBraid SHACL API, die auf Apache Jena 2.3.4 aufbaut, verwendet[35].

2.3.3 SPARQL

Die SPARQL Protocol and RDF Query Language (SPARQL) ist ein 2008 veröffentlichter W3C Standard, der eine RDF-Abfragesprache spezifiziert. Diese ist syntaktisch eng mit SQL verwandt und erlaubt das Abrufen und Verändern von Daten im RDF-Format. Eine Abfrage (Query) besteht aus einer beliebigen Anzahl von Tripeln, bei denen Subjekt, Prädikat und/oder Objekt durch eine Variable ersetzt werden können. Außerdem stehen eine Vielzahl von Funktionen zur Verfügung, um die Werte der Variablen zu testen, zu filtern, zu sortieren und zu modifizieren. SPARQL verfügt über die vier verschiedenen Abfrageformen SELECT, CONSTRUCT, ASK und DESCRIBE. Die am meisten verwendete Form ist SELECT, welche die Werte aller gebundenen Variablen als Tabelle zurückgibt. Jede Zeile ist dabei ein Ergebnis des Query. Mit der 2013 veröffentlichten Version 1.1 wurden die Funktionalitäten noch einmal stark erweitert. Der besondere Fokus lag dabei auf der Abfrage, dem

Austausch und der Änderung von RDF-Daten aus verschiedenen Quellen.[36]
Im Rahmen dieser Arbeit wird SPARQL nur in Verbindung mit SHACL 2.3.2 genutzt, um komplexe Einschränkungen zwischen den verschiedenen Konfigurationsmöglichkeiten darzustellen.

2.3.4 Apache Jena

Apache Jena ist ein Semantic Web Framework für die Programmiersprache Java, welches ursprünglich von HP Labs und seit 2010 von der Apache Software Foundation als Open Source Projekt unter der Apache License 2.0 entwickelt wird. Es enthält eine umfangreiche API zum Lesen und Schreiben von RDF-Graphen, zwei Versionen von Triple-Stores (SQL-basiert, nicht SQL-basiert) und unterstützt die Verarbeitung von OWL- und RDFS-Ontologien sowohl durch eingebettete Reasoner als auch die einfache Einbindung des Pellet-Reasoners[6]. Außerdem beinhaltet die Bibliothek eine SPARQL-Engine, welche die vollständige Spezifikation 1.1 unterstützt.[37]
Da Apache Jena das mit Abstand umfangreichste Open Source Framework im Bereich Semantic Web und deshalb weitgehend dokumentiert ist, wurde diese Bibliothek für die Implementierung der Ähnlichkeitsmaße ausgewählt.

3 Sternsensor ASTRO APS

In diesem Kapitel wird der Sternsensor ASTRO APS kurz beschrieben und anschließend die nötigen Anforderungen an eine Ontologie formuliert. Diese werden in Abschnitt 6.1 zur Evaluation genutzt.

3.1 Beschreibung

Der ASTRO APS ist ein autonomer Sternsensor der Jena-Optronik GmbH für LEO ("low earth orbit") und GEO ("geosynchronous equatorial orbit") Anwendungen. Die für den Anwendungsfall wesentlichen Komponenten des modular aufgebauten Geräts werden in diesem Abschnitt beschrieben [3].
Eine Explosionszeichnung des ASTRO APS mit Beschriftung der relevanten Bauteile ist in Abbildung 3.1 zu sehen. Im Zentrum des Geräts befindet sich der Detektor (1), welcher aus einem CMOS Detektorchip und der Optik besteht. Er besitzt eine maximale Updaterate, mit welcher er Bilddaten an den Prozessor liefern kann. Die integrierte Software errechnet aus diesen Daten eine Lagelösung, welche der Sternsensor an das AOCS (Attitude Orbit Control System) kommuniziert. Das geschieht mit einer konfigurationsabhängigen Updaterate der externen Kommunikationsschnittstelle. Diese kann logischerweise niemals größer sein als die maximale Updaterate des Detektors. Um den Detektor herum befindet sich die Hauptleiterplatte (2) und der Spannungswandler (3).
Die Hauptleiterplatte oder Main-PCB verbindet alle aktiven Komponenten des Sternsensors und enthält neben dem Prozessor und anderen Chips auch die Kommunikationsschnittstelle. Diese dient der Kommunikation zwischen ASTRO APS und Raumfahrzeug und kann für verschiedene Kommunikationsprotokolle konfiguriert werden.
Der Spannungswandler oder Power-Converter wird benötigt, um das Gerät mit den Spannungsversorgungen der unterschiedlichen Satelliten und Sonden zu betreiben (satellite bus voltage). Er wandelt deren Versorgungsspannung in die für den Betrieb des ASTRO APS notwendigen Betriebsspannungen um. Für verschiedene Versorgungsspannungen gibt es unterschiedliche Spannungswandler. Zum Schutz vor hohen

Einschaltströmen können "latching current limiter"(LCL) verbaut werden. Da diese nicht kaskadiert werden sollten, ist die Verwendung vom Satellitenbus abhängig. Je nach Konfiguration kann der Spannungswandler außerdem beim Anlegen der Versorgungsspannung ein- oder ausgeschaltet sein. Einige Varianten genügen zudem der Verwendung bei MIL-1553-STD Kommunikation (große Stromspitzen).
Die restlichen Bauteile des ASTRO APS sind entweder strukturell, etwa der Boden oder die Außenwände des Gehäuses, oder schützen die Optik des Sternsensors vor Streulicht (Baffle).

Die verschiedenen Konfigurationen des ASTRO APS können in vier technische Reifegrade unterteilt werden. Die niedrigste Stufe enthält alle Konfigurationen, die nicht existieren und auch technisch nicht möglich sind. Darüber liegen nicht existierende Konfigurationen, welche zwar technisch möglich sind, allerdings mit einem erheblichen einmaligen Aufwand für die Realisierung verbunden sind. Die nächst höhere Stufe sind existierende Konfigurationen, welche ohne großen Aufwand (beispielsweise das Ersetzen eines Spannungswandlers oder des Detektors) gebaut werden können. Die höchste Qualifikationsstufe beinhaltet alle bereits gebauten und qualifizierten Konfigurationen.

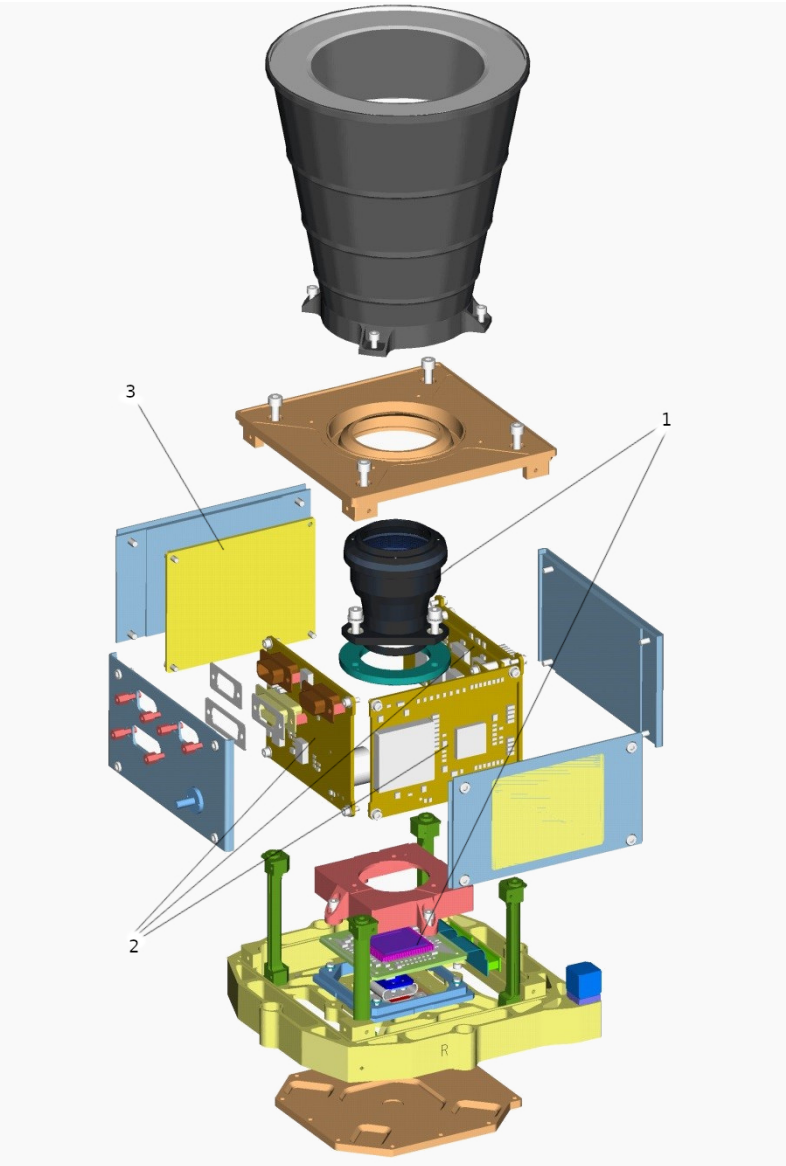

Abbildung 3.1: © Jena-Optronik GmbH,
Explosionszeichnung Sternsensor ASTRO APS

3.2 Anforderungen an die Ontologie

Nach einer allgemeinen Einführung in die Thematik des ASTRO APS durch die Experten von JOP haben sich einige Komponenten als besonders relevant für den Konfigurationsprozess herausgestellt. Obwohl in der Praxis noch deutlich mehr Parameter an die Kundenwünsche angepasst werden können, wurde sich auf eine kleine Teilmenge beschränkt, da die Arbeit eher die prinzipielle Machbarkeit untersuchen soll. Anhand der Anforderungen seitens JOP wurden die folgenden Kompetenzfragen herausgearbeitet, welche die Ontologie beantworten können muss. Diese werden in Abschnitt 6.1 aufgegriffen, um die Eignung der in Kapitel 4 präsentierten Ontologie zu prüfen.

1. Was sind die wesentlichen Komponenten der Konfiguration?
2. Welche Updaterate hat die Konfiguration?
3. Welcher Detektor wird in der Konfiguration verwendet?
4. Ist die Updaterate in Verbindung mit diesem Detektor technisch möglich?
5. Welches Kommunikationsprotokoll verwendet die Konfiguration?
6. Für welche Versorgungsspannung ist der Spannungswandler der Konfiguration ausgelegt?
7. Unterstützt der Spannungswandler der Konfiguration MIL-STD-1553 Kommunikation?
8. Besitzt der Spannungswandler der Konfiguration einen LCL?
9. Welche Qualifikationsstufe besitzt die Konfiguration?

4 Ontologie des Sternsensors

Zur Darstellung der verschiedenen Konfigurationen des Sternsensors ASTRO APS wurde eine einfache Ontologie in der Sprache OWL mit Hilfe des Protégé Editors entworfen. Diese stellt dabei keine vollständige Beschreibung der Domäne der Sternsensoren dar, sondern ist eher ein Testsystem mit deutlich reduzierter Komplexität, basierend auf dem ASTRO APS. Eine umfangreiche Ontologie von Sternsensoren könnte auf Basis der allgemeinen ECSS Norm erfolgen [38]. Um spezielle Einschränkungen der Konfigurationsmöglichkeiten abzubilden, wurde die Ontologie um einige SHACL-Shapes erweitert. Im folgenden Kapitel wird der Aufbau der Ontologie erläutert.
Diese wurde mit der Methodik von Noy und McGuinness [39] erstellt. Auf die Verwendung von anderen Ontologien wurde verzichtet, da es sich nur um eine vereinfachte Beschreibung der Domäne handelt. Es werden die gängigen Präfixe der verschiedenen Spezifikationen (RDF, RDFS, OWL, XML Datatypes), sowie der Präfix *st* für die URI der Ontologie (*http://ontology.dlr.de/spacecraft-parts/star-tracker*) verwendet.

4.1 Klassen

Die Klasse *st:StarTracker* beschreibt die Sternsensoren. Dabei ist der Grundgedanke, dass diese Klasse alle möglichen Sternsensoren umfasst und eine Instanz ein spezifischer Sternsensor mit exakt festgelegten Komponenten und Eigenschaften ist, welcher auch real existieren könnte. Somit ist jede Konfiguration eines Sternsensormodells eine Instanz von *st:StarTracker*. Die Menge aller ASTRO APS Konfigurationen ist also eine Untermenge der Sternsensoren und kann als Unterklasse *st:AstroAps* von *st:StarTracker* modelliert werden.
Die wichtigsten Hard- und Softwarekomponenten werden in der Ontologie als eigene Klassen dargestellt. Das sind der Detektor (*st:Detector*), der Spannungswandler (*st:PowerConverter*) und das Kommunikationsprotokoll (*st:CommunicationProtocol*).

Zu jeder Klasse gibt es eine *owl:ObjectProperty*, welche dem Sternsensor die entsprechende Komponente zuordnet.[1]

4.1.1 Kommunikationsprotokoll

Ein Kommunikationsprotokoll (*st:CommunicationProtocol*) hat in dieser vereinfachten Ontologie keine Eigenschaften.

4.1.2 Detektor

Ein Detektor (*st:Detector*) besitzt eine maximale Updaterate. Diese wird durch die Eigenschaft *st:hasMaxUpdateRate* vom Typ *xsd:double* abgebildet.

4.1.3 Spannungswandler

Die Versorgungsspannung, welche ein Spannungswandler (*st:PowerConverter*) unterstützt, wird auch als Nennspannung bezeichnet und in der Ontologie durch eine Eigenschaft *st:hasNominalVoltage* vom Typ *xsd:double* dargestellt wird. Ob ein Spannungswandler beim Anlegen der Versorgungsspannung eingeschaltet ist, wird durch die boolesche Eigenschaft *st:isInitialOn* abgebildet. Instanzen, welche die MIL-STD-1553 Kommunikation unterstützen, haben die boolesche Eigenschaft *st:isMilCompatible*. Das Vorhandensein eines LCL wird ebenfalls durch eine boolesche Eigenschaft *st:hasLcl* dargestellt.

4.1.4 Sternsensor

Jeder Sternsensor (*st:StarTracker*) hat neben seinen Komponenten noch weitere Eigenschaften, die ihm direkt zugeordnet sind. In diesem Fall wird lediglich noch die Updaterate relevant. Diese wird durch die Eigenschaft *st:hasUpdateRate* modelliert und hat den Typ *xsd:double*. Außerdem muss darauf geachtet werden, dass die maximale Updaterate des verbauten Detektors gleich oder höher ist. Die Darstellung dieser Einschränkung wird unter 4.3 beschrieben.

[1] Das Verhalten des adaptierten Ähnlichkeitsmaßes gegenüber einer beliebigen *owl:DatatypeProperty* kann mit der *owl:AnnotationProperty st:preferredDirection* beeinflusst werden. Für genauere Informationen siehe Abschnitt 5.3.

4.1.5 ASTRO APS

Der ASTRO APS (*st:AstroAps*) ist ein Sternsensor und damit ist die ihn beschreibende Klasse eine Unterklasse von *st:StarTracker*, welche alle möglichen Konfigurationen eines ASTRO APS enthält. Für die Klasse *st:AstroAps* werden keine weiteren Eigenschaften definiert. Den Instanzen kann über die *owl:AnnotationProperty st:qualificationLevel* ihre entsprechende Qualifikationsstufe zugeordnet werden. Diese wird durch die Zahlen 1 bis 3 kodiert. 1 steht dabei für eine nicht existierende Konfiguration, welche technisch möglich ist. 2 steht für existierende Konfigurationen, die ohne großen Aufwand gebaut werden können. 3 für bereits qualifizierte Konfigurationen. Technisch nicht mögliche Konfigurationen werden durch die in Abschnitt 4.3 beschriebene *AstroApsValidationConstraintShape* als ungültig erkannt und nicht betrachtet, könnten aber theoretisch mit 0 kodiert werden.

4.2 Instanzen

Die Informationen über die für den ASTRO APS verfügbaren Komponenten wurden von Jena Optronik bereitgestellt. Diese werden als Instanzen der entsprechenden Klassen in der Ontologie abgebildet und im Anhang unter A.1 kompakt dargestellt. Im weiteren Verlauf der Arbeit, werden die Instanzen mit ihrer *rdfs:label*-Eigenschaft bezeichnet.

4.3 Berücksichtigung bestehender Einschränkungen

Einige Komponenten können nicht mit anderen kombiniert werden oder bedingen in irgendeiner Weise die Eigenschaften des Sternsensors. Dadurch sind bestimmte Konfigurationen nicht möglich. Diese Einschränkungen werden in SHACL-Shapes abgebildet und können anschließend für die Validierung der Instanzen genutzt werden. Da in dieser Arbeit nur der ASTRO APS betrachtet wird, wurde auch nur eine *sh:NodeShape* für die Klasse *st:AstroAps* erstellt. Diese hat die URI *st:AstroApsValidationConstraintShape* und wird durch die Festlegung der *sh:targetClass* als *st:AstroAps* auf alle Instanzen dieser Klasse angewandt. Die tatsächlichen Einschränkungen werden durch *sh:PropertyShape*s definiert und der *sh:NodeShape* durch die Eigenschaften

sh:property oder *sh:sparql* zugeordnet.
In der erstellten Ontologie gibt es insgesamt drei Einschränkungen, welche auf diese Weise abgebildet wurden. Dazu gehört zum einen die Begrenzung der Updaterate des ASTRO APS auf den Bereich zwischen 8Hz und 16Hz, sowie die Beschränkung der Nennspannung eines Spannungswandlers auf das Intervall von 28V bis 100V. Die dritte Einschränkung stellt sicher, dass die Updaterate des Sternsensors kleiner oder gleich der maximalen Updaterate des Detektors ist. Dafür wird die in Auflistung 4.1 dargestellte *SPARQL-based Constraint* verwendet:

```
sh:sparql [
  sh:message "The update rate of the star tracker must be equal or
     less the maximum update rate of the detector." ;
  sh:prefixes _:prefixes ;
  sh:select """
    SELECT $this (st:hasUpdateRate AS ?path) ?value
    WHERE {
      $this st:hasDetector/st:hasMaxUpdateRate ?maxValue .
      $this st:hasUpdateRate ?value .

      FILTER(?value > ?maxValue)
    }
  """ ;
]
```

Auflistung 4.1: SPARQL-based Constraint für maximale Updaterate des Sternsensors

Die Eigenschaft *sh:message* legt die Nachricht fest, welche der SHACL-Prozessor zurückgeben soll, falls die Validierung fehlschlägt. Mit der Aussage über *sh:prefixes* werden vorher definierte Prefixe für die Verwendung im folgenden SPARQL-Query verfügbar gemacht. Die *sh:select* Eigenschaft bekommt schließlich das SPARQL-Query übergeben, welches zur Validierung genutzt werden soll. Dabei muss es sich um ein SELECT-Query handeln, das die inkorrekten Aussagen zurückgibt. Dafür steht die bereits vorher gebundene Variable *$this* zur Verfügung, welche auf die aktuell zu validierende Instanz (hier also eine ASTRO APS Konfiguration) verweist. In diesem Beispiel gibt die Abfrage die Werte der Variablen *$this*, *?path* (fix als *st:hasUpdateRate* gesetzt) und *?value* (Updaterate der aktuellen Konfiguration) für alle Konfigurationen zurück, bei denen die Updaterate des Sternsensors einen höheren Wert hat, als die maximale Updaterate des Detektors (*st:hasDetector/st:hasMaxUpdateRate*).

Weiterhin wurde eine SHACL-Shape mit der URI *st:AstroApsAvailableInstancesShape* zur Generierung der verfügbaren Konfigurationen erstellt. Dabei handelt es sich um mögliche ASTRO APS Versionen, bei denen die Kombinierbarkeit der Komponenten bereits sichergestellt ist und für die kein größerer Aufwand bei der Entwicklung

nötig ist. In der SHACL-Shape wurde dieses Wissen in *SPARQL-based Constraints* abgebildet. Ein Beispiel ist in 4.2 dargestellt.

```
sh:sparql [
  sh:message "Communication protocol UART is only available for
     values 28, 32 and 52 for the nominal voltage." ;
  sh:prefixes _:prefixes ;
  sh:select """
    SELECT $this (st:hasPowerConverter/st:hasNominalVoltage AS
       ?path) ?value
    WHERE {
      $this st:hasCommunicationInterface st:Uart .
      $this st:hasPowerConverter/st:hasNominalVoltage ?value .

      FILTER(?value NOT IN("28"^^xsd:double, "32"^^xsd:double,
         "52"^^xsd:double))
    }
  """ ;
] ;
```

Auflistung 4.2: SPARQL-based Constraint für verfügbare Kombinationen von Kommunikationsprotokoll UART und verschiedenen Nennspannungen

In diesem Beispiel werden die möglichen Kombinationen dem Kommunikationsprotokoll UART mit den Nennspannungen 28V, 32V und 52V und damit mit dem Spannungswandler beschrieben. Eine Übersicht über alle qualifizierten Kombinationen kann den Tabellen unter A.2 entnommen werden.
Das implementierte Programm kann eine Menge aus Eigenschaften und deren möglicher Werte permutieren und daraus Konfigurationen erzeugen. Jede dieser Konfigurationen wird anschließend gegen die *st:AstroApsQualifiedInstancesShape* getestet und nur als verfügbare Konfiguration übernommen, wenn sie den Einschränkungen dieser entspricht. Qualifizierte Konfigurationen erhalten die *owl:AnnotationProperty st:qualificationLevel* mit dem Wert 3 und werden der Konfigurationsmenge hinzugefügt. Dadurch müssen die mehreren tausend Konfigurationen der Qualifikationsstufe 3 nicht per Hand hinzugefügt werden.

5 Ähnlichkeitsmaße und deren Implementierung

Um ähnliche Produktkonfigurationen finden zu können, werden im Rahmen dieser Arbeit drei unterschiedliche Ähnlichkeitsmaße verglichen. Jedes davon verfolgt einen anderen Ansatz (mengen-basiert, probabilistisch oder graphen-basiert), somit sollten die Ergebnisse einen Überblick über die Praktikabilität von möglichen Verfahren im Kontext von Produktkonfigurationen liefern. Da alle Ähnlichkeitsmaße dienen im Allgemeinen dazu, die Ähnlichkeit von zwei Objekten durch eine reelle Zahl darzustellen. Dabei ist die Zahl umso größer je ähnlicher sich die Objekte sind. Somit verhalten sich Ähnlichkeitsmaße umgekehrt zu Distanzmaßen. Die Ergebnisse eines Ähnlichkeitsmaßes können anschließend genutzt werden, um ein Optimierungsproblem, wie das Finden eines möglichst ähnlichen Objektes zu lösen. Dabei muss jedoch darauf geachtet werden, dass die Werte verschiedener Ähnlichkeitsmaße im Allgemeinen nicht miteinander verglichen werden können. Obwohl es keine generelle Definition eines Ähnlichkeitsmaßes in der Mathematik gibt, wird hier eine für diese Arbeit hinreichende Definition gegeben.

Definition 1. Sei O eine endliche Menge von beliebigen Objekten. Eine Funktion $s : O \times O \to [0, 1]$ heißt **Ähnlichkeitsmaß** über O, falls für alle $i, j \in O$ gilt:

- $s(i, j) = 1 \Leftrightarrow i$ und j identisch sind
- $s(i, j) = s(j, i)$

Analog dazu können Distanzfunktionen folgendermaßen definiert werden.

Definition 2. Sei O eine endliche Menge von beliebigen Objekten. Eine Funktion $d : O \times O \to [0, \infty)$ heißt **Distanzmaß** über O, falls für alle $i, j \in O$ gilt:

- $d(i, j) = 0 \Leftrightarrow i$ und j identisch sind
- $d(i, j) = d(j, i)$

Um ein gegebenes Distanzmaß in ein Ähnlichkeitsmaß umzurechnen, kann der folgende Satz verwendet werden. Der Beweis ist trivial.

Satz 1. *Sei d ein Distanzmaß über einer endlichen Menge O von beliebigen Objekten. Ein zu d korrespondierendes Ähnlichkeitsmaß s über O kann über folgenden Zusammenhang gebildet werden:*

$$s(i,j) = \frac{1}{1+d(i,j)} \text{ für } i,j \in O$$

Im Folgenden werden die drei zu vergleichenden Ähnlichkeitsmaße und deren Implementierung beschrieben. Dabei wird auch kurz auf eventuelle Schwierigkeiten, welche im Zusammenhang mit der Ähnlichkeit von ontologie-basierten Produktkonfigurationen auftreten könnten, eingegangen.

5.1 Ähnlichkeitsmaß 1 - Jaccard-Index

Der Jaccard-Index oder auch Jaccard-Koeffizient ist eine vom Schweizer Botaniker Paul Jaccard entwickelte Funktion zur Bestimmung der Ähnlichkeit von Mengen. Zur Berechung des Jaccard-Index zweier Menge A und B, werden die Anzahl der gemeinsamen Elemente durch die Größe der Vereinigungsmenge geteilt:

Definition 3. $$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

Die Formel tauchte das erste Mal in seiner Veröffentlichung *Lois de distribution florale dans la zone alpine* von 1902 auf [24]. Der Jaccard-Index ist eines der ersten Ähnlichkeitsmaße und konnte sich in verschiedenen Bereichen der Mathematik und der Informatik etablieren. Ein bekannter Anwendungsfall ist beispielsweise die Erkennung von Duplikaten in Texten [40].
Der Jaccard-Index wurde als Ähnlichkeitsmaß für diese Arbeit ausgewählt, da er einen einfachen Ansatz besitzt und trotzdem eine gute Möglichkeit zur grundlegenden Bestimmung der Ähnlichkeit zweier Objekte darstellt. Der implementierte Algorithmus besitzt die mit Abstand geringste Komplexität aller implementierten Verfahren

und sollte dadurch auch die schnellste Laufzeit haben. Da er allerdings weder einen Unterschied zwischen numerischen und kategorischen Eigenschaften macht, noch zusätzliche Informationen wie beispielsweise eine Gewichtung mit einbezieht, wird erwartet, dass dieser Ansatz nicht für die praktische Verwendung nutzbar ist.
Im Rahmen der Implementierung wird jede Konfiguration in eine Menge von Eigenschaft-Wert-Paaren überführt. Die Eigenschaften einer Komponente werden mit dem von der *st:AstroAps*-Instanz ausgehenden OWL *Property Path* bezeichnet. Ein Beispiel dafür ist die Nennspannung des Spannungswandlers, welche durch *hasPowerConverter/hasNominalVoltage* dargestellt wird. Für zwei Mengen kann anschließend der Jaccard-Index berechnet werden, wobei zwei Eigenschaft-Wert-Paare genau dann gleich sind, wenn sowohl die Eigenschaft als auch der Wert übereinstimmen.

5.2 Ähnlichkeitsmaß 2 - Distanz im k-Means Clustering Algorithm von Ahmad und Dey

Dieses Ähnlichkeitsmaß stammt aus dem Artikel *A k-mean clustering algorithm for mixed numeric and categorical data* von Ahmad und Dey, der 2007 im Journal *Data & Knowledge Engineering* veröffentlich wurde [27]. Die darin vorgestellte Variante des k-Means-Algorithmus erweitert diesen, um gemischte numerische und kategorische Daten clustern zu können.
Da die Produktkonfigurationen ebenfalls als mehrdimensionale Vektoren mit numerischen als auch kategorischen Werten aufgefasst werden können, kann das in diesem Algorithmus verwendete Distanzmaß auch zur Berechnung der Ähnlichkeit von Konfigurationen verwendet werden. Weil die anderen Teile des Clustering-Algorithmus für das in dieser Arbeit behandelte Problem nicht relevant sind, wird auf diese im Folgenden nicht weiter eingegangen.
Da das Clustern von Daten als Optimierungsproblem betrachtet werden kann, geben Ahmad und Dey in ihrem Paper eine zu minimierende Kostenfunktion an. Diese ist die Summe der Abstände von allen n Objekten d_i zu ihren entsprechenden Clusterzentren C_j:

Definition 4.

$$\zeta(d_i, C_j) = \sum_{i=1}^{n} \vartheta(d_i, C_j) \tag{5.1}$$

$$\vartheta(d_i, C_j) = \sum_{t=1}^{m_r} (w_t(d_{it}^r - C_{jt}^r)^2) + \sum_{t=1}^{m_c} \Omega(d_{it}^c, C_{jt}^c)^2 \tag{5.2}$$

Dabei bezeichnet $\sum_{t=1}^{m_r}(w_t(d_{it}^r - C_{jt}^r)^2)$ den Abstand der numerischen Eigenschaften d_{it}^r eines Objekts d_i von den numerischen Eigenschaften C_{jt}^r seines nähesten Clusterzentrums C_i, gewichtet nach der Signifikanz w_t der Eigenschaft, welche aus der gesamten Menge an Objekten berechnet wird. Die Signifikanz entspricht dabei einer durch den Algorithmus bestimmten Wichtung der Eigenschaft.
$\sum_{t=1}^{m_c} \Omega(d_{it}^c, C_{jt}^c)^2$ ist der Abstand der kategorischen Eigenschaften d_{it}^c eines Objekts d_i von den kategorischen Eigenschaften C_{jt}^c seines nähestens Clusterzentrums C_i. Das Distanzmaß für kategorische Eigenschaften (d_{it}^c, C_{jt}^c) wird aus den tatsächlichen Werten der Objekte und der proportionalen Verteilung aller Werte dieser Eigenschaft in der Datenmenge berechnet. Somit ist die gesamte Distanzfunktion (5.2).
Statt dem Abstand zu einem Clusterzentrum kann natürlich auch der Abstand zwischen zwei beliebigen Objekten mit dieser Funktion berechnet werden. Um dieses Distanzmaß in ein entsprechendes Ähnlichkeitsmaß umzurechnen, kann Satz 1 verwendet werden.
Zur Bestimmung des Abstands der Werte von kategorischen Eigenschaften verwendet der Algorithmus zwei bedingte Wahrscheinlichkeiten. Zum einen die Wahrscheinlichkeit $P_i(\omega/x)$ mit welcher ein Objekt, dass den Wert x für Eigenschaft A_i hat, einen Wert aus einer Teilmenge w aller möglichen Werte der Eigenschaft A_j besitzt. Analog dazu wird die zweite Wahrscheinlichkeit $P_i(\sim \omega/y)$ mit der ein Objekt, welches den Wert y für Eigenschaft A_i hat, keinen Wert aus der Menge w für Eigenschaft A_j besitzt, definiert. Die Distanz zwischen zwei Werten x und y der Eigenschaft A_i wird mit der folgenden Formel berechnet, wobei ω stets die Teilmenge der Werte von Eigenschaft A_j bezeichnet, für welche der Summand maximal wird:

Definition 5.

$$\delta(x,y) = \frac{1}{1m-1} \sum_{j=1\ldots m, i\neq j} P_i(\omega/x) + P_i(\sim\omega/y) - 1 \tag{5.3}$$

Bei der Berechnung dieser Distanz werden auch alle Werte der numerischen Eigenschaften mit einbezogen. Diese werden dafür normalisiert und anschließend in S Intervalle unterteilt, um sie diskretisieren zu können. Außerdem werden die Abstände zwischen den diskretisierten Werten u_{ik} genutzt, um die Signifikanz w_i einer Eigenschaft A_i wie folgt zu ermitteln:

Definition 6.

$$w_i = \frac{2}{S(S-1)} \cdot \sum_{k=1}^{S} \sum_{j>k}^{S} \delta(u_{ik}, u_{ij}) \tag{5.4}$$

Für die Implementierung werden die Konfigurationen auf die gleiche Weise, wie in Abschnitt 5.1 beschrieben, in Mengen von Eigenschaft-Wert-Paaren überführt. Anschließend wird daraus eine Matrix erstellt, die als Spalten die Eigenschaften und als Zeilen die Werte einer Konfiguration enthält. Anhand dieser Matrix werden dann die oben beschriebenen Berechnungen durchgeführt. Eine genauere Beschreibung des Distanzmaßes und der Implementierung der Berechnungen kann dem Paper [27] entnommen werden.
Obwohl der Ansatz dieses Ähnlichkeitsmaß bereits deutlich weitreichender ist als der des Jaccard-Index, stellt sich die Frage zur praktischen Nutzbarkeit für die Berechnung der Ähnlichkeit von Produktkonfigurationen. Ob der Ansatz über die proportionale Verteilung der Eigenschaftswerte in diesem Kontext überhaupt sinnvoll ist, muss sich in der Anwendung zeigen. Ein weiterer Nachteil ist, dass dieser Algorithmus keine Konfigurationen mit unterschiedlich vielen Eigenschaften unterstützt.

5.3 Ähnlichkeitsmaß 3 - Angepasstes Verfahren

Dieses Ähnlichkeitsmaß wurde speziell für den in dieser Arbeit beschriebenen Anwendungsfall unter Rücksprache mit den Experten von JOP konstruiert und nutzt die baumartige Struktur der Konfigurationen in der Ontologie.
Im Gegensatz zu den anderen Verfahren handelt es sich bei diesem nicht um ein Ähnlichkeitsmaß im strengen Sinn, da das Ergebnis von der Reihenfolge der Eingabe abhängt. Es ist also nicht symmetrisch.
Der Algorithmus beginnt bei den Knoten der zu vergleichenden Instanzen und bestimmt dann rekursiv die Ähnlichkeit der per *owl:DatatypeProperty* oder *owl:ObjectProperty* erreichbaren Knoten. Bei der Behandlung von numerischen Eigenschaften bezieht er die *owl:AnnotationProperty st:preferredDirection* mit ein, durch welche angegeben werden kann, dass für diese Eigenschaft ein höherer ("high") oder niedriger ("low") Wert zu bevorzugen ist. Standardmäßig werden immer nahe beieinander liegende Werte ("close") als ähnlich angenommen.
Als Argumente erhält der Algorithmus die beiden Konfigurationsinstanzen i_1, i_2. Da das Verfahren asymmetrisch ist, wird i_1 als Referenz und i_2 als Anfragekonfiguration angenommen. Das Ergebnis gibt demnach an, wie ähnlich i_2 zu i_1 ist. Als weitere Parameter werden zum einen die Gewichtung der einzelnen Eigenschaften benötigt, sowie die Wertebereiche von numerischen Attributen. Im folgenden wird der Algorithmus an Hand von Pseudocode genauer erläutert.

Algorithm 1 Adapted Similarity Algorithm

Require: Instances i_1, i_2
  return instanceSimilarity(i_1, i_2, *null*)

Beim Aufruf des Algorithmus müssen die beiden Instanzen i_1 und i_2 übergeben werden. Das Ergebnis wird anschließend durch die Funktion *instanceSimilarity* berechnet, welche beide Instanzen und den Wert *null* für p_{root} übergeben bekommt.

```
function INSTANCESIMILARITY(Instances i_1, i_2, Property p_root)
    sim = 0.0
    for all Property p in properties(i_1 ∪ i_2) do
        if p is owl:DatatypeProperty then
            sim_local = literalSimilarity(value(i_1, p) ,value(i_2, p), p)
        else if p is owl:ObjectProperty then
            if p != p_root then // Meide Knoten des vorherigen rekursiven Aufrufs
                sim_local = instanceSimilarity(value(i_1, p), value(i_2, p),p)
            end if
        end if
        sim += weight(p) * sim_local
    end for
    if properties(i_1 ∪ i_2) == ∅ then
        // Vergleiche Label falls Instanzen keine Eigenschaften haben
        if label(i_1) == label(i_2) then
            sim = 1.0
        end if
    end if
    return sim
end function
```

Diese Funktion durchläuft alle Eigenschaften p von i_1 und i_2 und berechnet wie ähnlich i_2 zu i_1 ist. Falls es sich bei p um eine *owl:DatatypeProperty* handelt, wird die lokale Ähnlichkeit p_{local} durch die Funktion *literalSimilarity* berechnet. Diese bekommt jeweils die Werte der Instanzen für p und p selbst übergeben. Wenn p eine *owl:ObjectProperty* is, wird zunächst überprüft, ob p ungleich p_{root} ist. Dadurch wird verhindert, dass die Eigenschaft aus dem vorherigen rekursiven Aufruf erneut betrachtet wird. Falls ja, wird p_{local} durch einen rekursiven Aufruf von *instanceSimilarity* ermittelt. Dabei werden als neue Instanzen die Werte von i_1 und i_2 für p übergeben und p_{root} auf p gesetzt.
Die lokalen Ähnlichkeiten aller Eigenschaften werden mit der entsprechenden Wichtung zu *sim* aufsummiert. Falls weder i_1 noch i_2 eine OWL-Eigenschaft hat, werden ihre *rdfs:label*-Werte verglichen und *sim* auf 1.0 gesetzt, wenn diese übereinstimmen. Schlussendlich wird *sim* zurückgegeben.

```
function LITERALSIMILARITY(Literals v1, v2, Property p)
    if range(p) is numeric then
        if preferredDirection(p) == "low" then
            if v2 <= v1 then
                return 1.0
            end if
        else if preferredDirection(p) == "high" then
            if v2 >= v1 then
                return 1.0
            end if
        end if
        return 1.0 − δ^p_rel(v1, v2)
    else// Werte sind kategorisch
        if v2 == v1 then
            return 1.0
        else
            return 0.0
        end if
    end if
end function
```

Die Funktion *literalSimilarity* bekommt zwei Literale v_1 und v_2, sowie die dazugehörige Eigenschaft p übergeben, und berechnet wie ähnlich v_2 zu v_1 ist.
Falls die Werte von p numerisch sind, wird die zugehörige *owl:AnnotationProperty st:preferredDirection* überprüft. Falls deren Wert "low" und v_2 kleiner oder gleich v_1 ist, wird 1.0 zurückgegeben. Analog dazu ist das Ergebnis 1.0, wenn der Wert *high* und v_2 größer oder gleich v_1 ist.
Insofern keiner dieser Fälle eintritt, wird die Ähnlichkeit über $1.0 - \delta^p_{rel}(v_1, v_2)$ berechnet. Dabei steht $\delta^p_{rel}(v_1, v_2)$ für die relative Distanz von v_2 zu v_1 als Werte der Eigenschaft p und ist folgendermaßen definiert.

Definition 7.

$$\delta^{p}_{rel}(v_1, v_2) = \begin{cases} 0 & falls\, v_2 = v_1 \\ 1 - \dfrac{v_1 - v_2}{v_1 - min(p)} & falls\, v_2 < v_1 \\ 1 - \dfrac{v_2 - v_1}{max(p) - v_1} & falls\, v_2 > v_1 \end{cases} \tag{5.5}$$

Hier bezeichnet $min(p)$ das Minimum und $max(p)$ das Maximum des Wertebereichs von p. Durch diese asymmetrische Berechnung zur Ähnlichkeit von numerischen Werten, ist der gesamte Algorithmus nicht symmetrisch.

Die Gewichtung der Eigenschaften wurde im konkreten Fall für die Klasse *st:AstroAps* leicht angepasst. Da das Kommunikationsprotokoll für JOP relevanter als die anderen Komponenten ist, wurde die Eigenschaft *st:hasCommunicationProtocol* mit 0.3 etwas höher gewichtet und die Gewichte der restlichen entsprechend angepasst. Die Eigenschaften der anderen Klassen sind alle gleichmäßig gewichtet.

Tabelle 5.1: Gewichtung der Eigenschaften von *st:AstroAps*

Eigenschaft	Gewicht
st:hasCommunicationProtocol	0.300
st:hasPowerConverter	0.233
st:hasDetector	0.233
st:hasUpdateRate	0.233

Da dieser Algorithmus basierend auf den Vorstellungen der Experten von JOP entworfen wurde, ist anzunehmen, dass er auch die besten Ergebnisse der drei Verfahren liefert. Obwohl er im Gegensatz zu den anderen nicht symmetrisch und damit kein Ähnlichkeitsmaß per Definition mehr ist, steht der praktische Nutzen im Vordergrund und sollte diesen Schritt im Entwurfsprozess rechtfertigen.
Der größte Nachteil dieses Verfahrens ist höchstwahrscheinlich die Laufzeit, da der Algorithmus auf einem komplexen OWL-Graphen arbeitet und die Zugriffszeiten hier deutlich höher liegen dürften als bei den Mengen und Matrizen, welche die anderen nutzen.

Die Implementierung der Ähnlichkeitsmaße wurde zusammen mit der Ontologie und einer Beispielanwendung als Software unter der MIT-Lizenz veröffentlicht [41].

6 Evaluation

Nachdem in den vorangegangenen Kapiteln die erarbeiteten Lösungsansätze für das in der Arbeit zu bearbeitende Problem vorgestellt wurden, erfolgt nun die Auswertung anhand vorher aufgeführter Kriterien. Die Eignung der Ontologie wird dabei durch die Beantwortung aller in Abschnitt 3.2 aufgelisteten Kompetenzfragen geprüft.
Zur Bewertung der Ähnlichkeitsmaße werden einige beispielhafte Konfigurationsanfragen ermittelt und für jede die Ähnlichkeit mit den 14 bestehenden Konfigurationen des ASTRO APS durch alle drei Ähnlichkeitsmaße berechnet. Die Ergebnisse werden anschließend durch die Experten von JOP bewertet.

6.1 Eignung der Ontologie

Im Folgenden wird die Eignung der erstellten Ontologie zur Darstellung der von JOP angegebenen Minimalbeschreibung des Sternsensors ASTRO APS überprüft. Dafür wird aufgezeigt, wie die Kompetenzfragen aus Abschnitt 3.2 durch die Konstrukte der Ontologie beantwortet werden können.

1. **Was sind die wesentlichen Komponenten der Konfiguration?**
   Alle *owl:ObjectProperties* verweisen auf eine Hard- oder Softwarekomponente der Konfiguration. Somit sind die wesentlichen Komponenten die Werte dieser Eigenschaften. Eine entsprechende SPARQL-Anfrage kann in Auflistung A.1 gefunden werden.

2. **Welche Updaterate hat die Konfiguration?**
   Die *st:hasUpdateRate* Eigenschaft gibt die Updaterate der Konfiguration in Hertz an. Eine entsprechende SPARQL-Anfrage kann in Auflistung A.2 gefunden werden.

3. **Welcher Detektor wird in der Konfiguration verwendet?**
   Die *st:hasDetector* Eigenschaft gibt *st:Detector*-Instanz der Konfiguration an. Eine entsprechende SPARQL-Anfrage kann in Auflistung A.3 gefunden werden.

4. **Ist die Updaterate in Verbindung mit diesem Detektor technisch möglich?** Diese Frage ist durch die Validierung der Konfiguration mit Hilfe eines SHACL-Prozessors und der *st:AstroApsValidationConstraintShape* zu beantworten. Falls der Validierungsbericht einen Fehler mit der Nachricht "The update rate of the star tracker must be equal or less the maximum update rate of the detector." entählt, ist die Kombination nicht zulässig.

5. **Welches Kommunikationsprotokoll verwendet die Konfiguration?** Die *st:hasCommunicationProtocol* Eigenschaft gibt die *st:CommunicationProtocol*-Instanz der Konfiguration an. Eine entsprechende SPARQL-Anfrage kann in Auflistung A.4 gefunden werden.

6. **Für welche Versorgungsspannung ist der Spannungswandler der Konfiguration ausgelegt?**
   Über die *st:hasPowerSupply* Eigenschaft kann zunächst der verwendete Spannungswandler bestimmt werden. Dessen Versorgungsspannung wird durch die *st:hasNominalVoltage* Eigenschaft in Volt angegeben. Eine entsprechende SPARQL-Anfrage kann in Auflistung A.5 gefunden werden.

7. **Unterstützt der Spannungswandler der Konfiguration MIL-STD-1553 Kommunikation?**
   Über die *st:hasPowerSupply* Eigenschaft kann zunächst der verwendete Spannungswandler bestimmt werden. Falls dessen Eigenschaft *st:isMilCompatible* den Wert *true* besitzt, wird die MIL-STD-1553 Kommunikation unterstützt. Eine entsprechende SPARQL-Anfrage kann in Auflistung A.6 gefunden werden.

8. **Besitzt der Spannungswandler der Konfiguration einen LCL?**
   Über die *st:hasPowerSupply* Eigenschaft kann zunächst der verwendete Spannungswandler bestimmt werden. Falls dessen Eigenschaft *st:hasLcl* den Wert *true* besitzt, besitzt die Konfiguration einen LCL. Eine entsprechende SPARQL-Anfrage kann in Auflistung A.7 gefunden werden.

9. **Welche Qualifikationsstufe besitzt die Konfiguration?**
   Die *owl:AnnotationProperty st:qualificationLevel* gibt die Qualifikationsstufe

der Konfiguration als Zahl von 1 - 3 an. Für die Bedeutung dieser Zahlen siehe Abschnitt 4.1.5. Eine entsprechende SPARQL-Anfrage kann in Auflistung A.8 gefunden werden.

Die Ontologie kann alle Kompetenzfragen beantworten und ist somit für den geforderten Anwendungsfall geeignet.

6.2 Bewertung der Ähnlichkeitsmaße

6.2.1 Bestimmung aussagekräftiger Anfragen

Um einen möglichst guten Vergleich der verschiedenen Verfahren zu erhalten, werden aussagekräftige Anfragen benötigt. Diese sollten zu unterschiedlichen Ergebnissen zwischen den Algorithmen führen.
Zur Bestimmung solcher Anfragen wurden neue Instanzen der Klasse *st:AstroAps* durch Permutation möglicher Eigenschaftswerte erzeugt. Dabei wurden für das Kommunikationsprotokoll und den Detektor nur die existierenden Instanzen verwendet, während für den Spannungswandler ebenfalls die Eigenschaftswerte permutiert wurden. Für die numerischen Eigenschaften zu Updaterate und Nennspannung wurden die Werte $\{8, 10, 12, 14, 16\}$ beziehungsweise $\{28, 33, 38...93, 98\}$ verwendet.
Anschließend wurde pro erzeugter *st:AstroAps*-Instanz die Ähnlichkeit der 14 qualifizierten Konfigurationen mit jedem Verfahren bestimmt. Nur wenn sich die fünf ähnlichsten Konfigurationen laut den drei Ähnlichkeitsmaßen in mindestens drei Stellen unterscheiden, wurde die Instanz als aussagekräftige Anfrage betrachtet. Aus dieser Menge an Anfragen wurden fünf untereinander möglichst unterschiedliche ausgewählt. Diese sind in Tabelle 6.1 dargestellt.

Tabelle 6.1: Für die Evaluation verwendete Anfragen

Nr	Updaterate	Nennspannung	hat LCL	ist initial an	MIL-kompatibel	Kommunikationsprotokoll	Detektor
1	10	53	✓	✓	✓	SCDI	HAS2
2	8	53	✗	✗	✓	HDLC	HAS2
3	8	33	✓	✓	✓	UART	STAR1000
4	10	88	✗	✗	✓	SCDI	STAR1000
5	8	28	✗	✗	✗	HDLC	STAR1000

6.2.2 Bewertung der Ergebnisse durch JOP

In diesem Abschnitt wird die Bewertung der Ergebnisse der Ähnlichkeitsmaße für die fünf ausgewählten Anfragen durch die Experten von JOP beschrieben. Dafür werden jeweils zuerst die ähnlichsten 5 Konfigurationen jedes Verfahrens inklusive des berechneten Ergebnisses in Tabellenform dargestellt. Die für JOP zur Anfrage passendste Konfiguration wird zur besseren Visualisierung in grün, unpassende Konfigurationen in orange eingefärbt. Der numerische Wert der Ähnlichkeit wird mit so vielen signifikanten Stellen, wie nötig sind um zwei Ergebnisse zu unterscheiden, angegeben, mindestens jedoch mit zwei. Im Anschluss an die Tabelle werden die passendsten Konfigurationen kurz erläutert und anschließend mit den Ergebnissen der Ähnlichkeitsmaße verglichen.

Anfrage 1

Tabelle 6.2: Ergebnisse zu Anfrage 1

Ähnlichkeitsmaß	1	2	3	4	5
Jaccard	7 (0.38)	8 (0.38)	9 (0.38)	12 (0.38)	14 (0.38)
Ahmad Dey	12 (0.58)	9 (0.569)	7 (0.567)	14 (0.561)	8 (0.54)
Angepasst	7 (0.94)	8 (0.642)	14 (0.64)	9 (0.58)	12 (0.58)

Bei dieser Anfrage passt nur Konfiguration 7. Lediglich der Spannungswandler hat hier eine zu hohe Nennspannung. Bei den Konfigurationen 8, 9 und 12 stimmt weder das Kommunikationsprotokoll noch der Spannungsbereich des Spannungswandlers überein, daher sind diese nicht sinnvoll.
Somit liefert das angepasste Verfahren das beste Ergebnis. Der Jaccard-Index ermittelt zwar auch Konfiguration 7, allerdings hat diese den gleichen Ähnlichkeitswert wie 8, 9 und 12. Das Ähnlichkeitsmaß aus dem Algorithmus von Ahmad & Dey bestimmt die Konfigurationen 12 und 9 als passendste und ist somit ungeeignet.

Anfrage 2

]

Tabelle 6.3: Ergebnisse zu Anfrage 2

Ähnlichkeitsmaß	1	2	3	4	5
Jaccard	4 (0.38)	7 (0.29)	8 (0.29)	12 (0.29)	14 (0.29)
Ahmad Dey	12 (0.62)	4 (0.61)	14 (0.51)	9 (0.49)	8 (0.48)
Angepasst	9 (0.77)	4 (0.642)	7 (0.642)	8 (0.642)	14 (0.64)

Konfiguration 9 stimmt sehr gut mit der Anfrage überein. Hier ist wieder nur der Spannungswandler nicht ideal. Dieser ist bei den Konfigurationen 7, 8 und 14 ungefähr passend. Somit können diese dabei helfen, den richtigen Spannungswandler auszuwählen. Praktisch würde man Konfiguration 9 als Grundlage nutzen und dem Kunden einen Spannungswandler mit 28V Nennspannung vorschlagen.
Auch hier liefert das angepasste Ähnlichkeitsmaß das korrekte Ergebnis. Das Ahmad-Dey-Verfahren bestimmt zwar auch die Konfigurationen 4, 8, 9 und 14, allerdings landet die passendste Konfiguration 9 nur auf Platz 4. Der Jaccard-Index ermittelt ebenfalls die Konfigurationen mit passendem Spannungswandler, allerdings taucht Konfiguration 9 hier gar nicht unter den fünf ähnlichsten auf.

Anfrage 3

Tabelle 6.4: Ergebnisse zu Anfrage 3

Ähnlichkeitsmaß	1	2	3	4	5
Jaccard	3 (0.38)	6 (0.38)	10 (0.38)	13 (0.38)	11 (0.29)
Ahmad Dey	3 (0.70)	10 (0.70)	6 (0.65)	2 (0.6147)	5 (0.6144)
Angepasst	14 (0.71)	12 (0.65)	1 (0.64)	2 (0.64)	3 (0.64)

Für diese Anfrage kommen die Konfigurationen 12 und 14 in Betracht. Zwar stimmt hier der Detektor nicht, dieser kann aber prinzipiell ersetzt werden. Konfiguration 14

hat den passenderen Spannungswandler und wird deshalb favorisiert. Die Konfigurationen 11 und 13 sind aufgrund der zu hohen Nennspannung des Spannungswandlers nicht geeignet.
Wieder stimmen die Ergebnisse des angepassten Verfahrens am besten mit den Vorstellungen von JOP überein. Die ähnlichste Konfiguration ist hier die 14 gefolgt von der 12. Sowohl der Jaccard-Index als auch der Algorithmus von Ahmad & Dey liefern kein zufriedenstellendes Ergebnis.

Anfrage 4

Tabelle 6.5: Ergebnisse zu Anfrage 4

Ähnlichkeitsmaß	1	2	3	4	5
Jaccard	1 (0.38)	5 (0.38)	11 (0.38)	3 (0.29)	4 (0.29)
Ahmad Dey	6 (0.62)	5 (0.56)	11 (0.55)	13 (0.54)	1 (0.537)
Angepasst	7 (0.71)	11 (0.64)	1 (0.61)	5 (0.58)	3 (0.52)

Da die Nennspannung bei dieser Anfrage sehr hoch ist, kommen nur Spannungswandler mit 100V in Betracht. Bei Konfiguration 7 passt außerdem das Kommunikationsprotokoll, sodass nur der Detektor ersetzt werden müsste. Diese ist somit die ähnlichste Konfiguration. Bei 5, 6 und 10 ist die Nennspannung des Spannungswandler viel zu niedrig, daher sind diese nicht geeignet.
Das angepasste Ähnlichkeitsmaß bestimmt Konfiguration 7 als am ähnlichsten und liefert damit als einziges Verfahren ein praktisch nutzbares Ergebnis. Die beiden anderen Algorithmen mit den Konfigurationen 5 und 6 auf den vorderen Plätzen gar gänzlich unpassende Empfehlungen.

Anfrage 5

Tabelle 6.6: Ergebnisse zu Anfrage 5

Ähnlichkeitsmaß	1	2	3	4	5
Jaccard	5 (0.29)	6 (0.29)	13 (0.29)	1 (0.2)	2 (0.2)
Ahmad Dey	5 (0.60)	9 (0.57)	2 (0.56)	3 (0.54)	10 (0.54)
Angepasst	9 (0.65)	1 (0.58)	2 (0.58)	5 (0.58)	6 (0.58)

Zu dieser Anfrage passt lediglich Konfiguration 9, da hier sowohl der Spannungsbereich des Spannungswandlers als auch das Kommunikationsprotokoll übereinstimmen. Lediglich der Detektor müsste getauscht werden. Konfiguration 13 das falsche Kommunikationsprotokoll und viel zu niedrige Nennspannungen und sind daher in diesem Fall nicht zu gebrauchen.
Auch bei der letzten Anfrage liefert das angepasste Verfahren die beste Konfiguration. Das Ähnlichkeitsmaß aus dem Ahmad-Dey-Algorithmus ermittelt Konfiguration 9 als am zweit ähnlichsten und erzielt damit auch ein annehmbares Ergebnis. Laut Jaccard-Index ist Konfiguration 13 ähnlicher als Konfiguration 9. Das Verfahren ist für diese Anfrage also ungeeignet.

7 Fazit

Diese Bachelorarbeit hat sich mit der formalen Beschreibung von Produktkonfigurationen durch Ontologien und verschiedenen Verfahren zur Bestimmung der Ähnlichkeit dieser Konfigurationen beschäftigt. Als Produktbeispiel diente der Sternsensor ASTRO APS der Jena-Optronik GmbH.

Die anhand der Informationen von JOP erstellte Ontologie kann alle gestellten Kompetenzfragen beantworten und beschreibt somit den betrachteten Beispielfall ausreichend gut. Die technischen Einschränkungen konnten mit Hilfe von SHACL abgebildet und validiert werden. Auch wenn es sich um ein Minimalbeispiel des ASTRO APS handelte und dadurch relativ einfache Einschränkungen vorgegeben waren, können durch die Ausdrucksmächtigkeit von SHACL-SPARQL auch komplexe Zusammenhänge abgefragt werden.

Um in der Praxis genutzt werden zu können, muss eine Ontologie zur Beschreibung von Konfigurationen möglichst alle Komponenten und Eigenschaften des Produkts abbilden. Eine vereinfachte Version, wie sie in dieser Arbeit vom Sternsensor ASTRO APS verwendet wurde, ist also nicht ausreichend. Der genutzte Ansatz zur Darstellung von Einschränkungen durch SHACL ist vielversprechend und sollte auch bei großen Ontologien verwendet werden können. Es ergeben sich hier interessante Forschungsthemen, wie die automatische Generierung von SHACL-Shapes aus Spezifikationsdokumenten oder die vollständige Beschreibung eines Produkts/einer Produktart (beispielsweise den Sternsensoren) nach internationalen technischen Standards.

Bei der Evaluation der Ähnlichkeitsmaße hat sich das angepasste Verfahren eindeutig als beste Methode herausgestellt. Es lieferte für alle fünf Anfragen die für JOP passendste Konfiguration als ähnlichste. Der einfache Ansatz des Jaccard-Index ist zur Bestimmung von ähnlichen Konfigurationen ungeeignet, da er zum einen zu vielen Konfigurationen den gleichen Wert zuordnet und zum andern häufig unpassende Ergebnisse liefert. Das verteilungsbasierte Ähnlichkeitsmaß aus dem Clustering-Algorithmus von Ahmad & Dey ist zwar eindeutiger bei der Zuordnung des Zahlenwerts, sieht aber auch zu viele unpassende Konfigurationen als ähnlich an.

Anstelle des Jaccard-Index wäre im Nachhinein wohl die Betrachtung eines weiteren semantischen Ähnlichkeitsmaßes interessanter gewesen.
Da das angepasste Verfahren unter Rücksprache mit den Experten von JOP entwickelt wurde, ist es nicht weiter verwunderlich, dass es am besten abschneidet. Es bietet einen guten Ausgangspunkt bei der Implementierung eines voll funktionsfähigen Systems, dass die Mitarbeiter von JOP bei der Auswahl einer passenden Ausgangskonfiguration zu einer Kundenanfrage unterstützt. Obwohl das angepasste Verfahren nicht symmetrisch und dadurch kein Ähnlichkeitsmaß im strengen Sinn ist, wurde die Praktikabilität durch die Evaluation bestätigt. Zusammen mit der Ontologie bildet es einen praktisch nutzbaren Ansatz zur Bestimmung von ähnlichen Produktkonfigurationen. Somit wurde das Forschungsziel der Arbeit erreicht.
Für die Verwendung des angepassten Ähnlichkeitsmaßes mit einer vollständigeren Beschreibung des ASTRO APS mit deutlich mehr Komponenten und Eigenschaften, müssen die Gewichte, wahrscheinlich aber auch die Behandlung von bestimmen Eigenschaftstypen weiter angepasst werden. Vor allem bei der Ähnlichkeit von kategorischen Eigenschaften könnte eine genauere Bewertung durch semantische oder probabilistische Ansätze (wie dem von Ahmad und Dey) sinnvoll sein. Da das Verfahren nacheinander die einzelnen Knoten der baumartigen Beschreibung einer Konfiguration abarbeitet, sollte die Anpassung und Erweiterung der Schritte für bestimmte Knotenarten kein Problem darstellen. Weitere Forschung könnte sich mit der automatischen Bestimmung von Eigenschaftsgewichten durch Methoden der künstlichen Intelligenz oder der Spezifizierung und Entwicklung von Anwendungssoftware zur Bestimmung von ähnlichen Konfigurationen auf Herstellerseite beschäftigen.

Literaturverzeichnis

[1] Jena-Optronik GmbH. *Übersicht - Unternehmen - Jena Optronik*. 2020. URL: https://www.jena-optronik.de/de/unternehmen/profil.html (besucht am 19. 02. 2020).

[2] Jena-Optronik GmbH. *Übersicht - Lageregelungssensoren - Jena Optronik*. 2020. URL: https://www.jena-optronik.de/de/lageregelungssensoren.html (besucht am 19. 02. 2020).

[3] Jena-Optronik GmbH. *Sternsensor ASTRO APS - Lageregelungssensoren - Jena Optronik*. 2020. URL: https://www.jena-optronik.de/de/lageregelungssensoren/sternsensor-astro-aps.html (besucht am 19. 02. 2020).

[4] Thomas R. Gruber. „A translation approach to portable ontology specifications“. In: *Knowledge Acquisition* 5.2 (1993), S. 199–220. ISSN: 1042-8143. DOI: https://doi.org/10.1006/knac.1993.1008. URL: http://www.sciencedirect.com/science/article/pii/S1042814383710083.

[5] Nicola Guarino, Daniel Oberle und Steffen Staab. „What Is an Ontology?“ In: *Handbook on Ontologies*. Mai 2009, S. 1–17. DOI: 10.1007/978-3-540-92673-3_0.

[6] Stardog Union. *GitHub - Pellet: An Open Source OWL DL reasoner for Java*. 2017. URL: https://github.com/stardog-union/pellet (besucht am 04. 03. 2020).

[7] Dmitry Tsarkov und Ian Horrocks. „FaCT++ Description Logic Reasoner: System Description“. In: *Automated Reasoning*. Hrsg. von Ulrich Furbach und Natarajan Shankar. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, S. 292–297. ISBN: 978-3-540-37188-5.

[8] Birte Glimm u. a. „Hermit: An Owl 2 Reasoner“. In: *Journal of Automated Reasoning* 53 (Okt. 2014). DOI: 10.1007/s10817-014-9305-1.

[9] Michael Kifer. „F-Logic: A Higher-Order Language for Reasoning about Objects, Inheritance and Scheme“. In: *ACM SIGMOD* (1989), S. 134–146.

[10] W3C RDF Working Group. *RDF Schema 1.1*. 25. Feb. 2014. URL: https://www.w3.org/TR/rdf-schema/ (besucht am 04. 03. 2020).

[11] W3C Semantic Web Deployment Working Group. *SKOS Simple Knowledge Organization System Reference*. 18. Aug. 2009. URL: https://www.w3.org/TR/skos-reference/ (besucht am 08. 06. 2020).

[12] W3C OWL Working Group. *OWL 2 Web Ontology Language Document Overview (Second Edition)*. 2012. URL: https://www.w3.org/TR/owl2-overview/ (besucht am 04. 03. 2020).

[13] Mark A. Musen. „The Protégé Project: A Look Back and a Look Forward“. In: *AI Matters* 1.4 (Juni 2015), S. 4–12. DOI: 10.1145/2757001.2757003. URL: https://doi.org/10.1145/2757001.2757003.

[14] Timo Soininen u. a. „Towards a general ontology of configuration“. In: *Artificial intelligence for engineering design analysis and manufacturing* 12 (Sep. 1998), S. 357–372. DOI: 10.1017/S0890060498124083.

[15] Jiao Tao. „Integrity Constraints for the Semantic Web: An OWL 2 Dl Extension“. Diss. USA, 2012. ISBN: 9781267679673.

[16] Jiao Tao. „Adding Integrity Constraints to the Semantic Web for Instance Data Evaluation“. In: *International Semantic Web Conference*. 2010.

[17] Boris Motik, Ian Horrocks und Ulrike Sattler. „Bridging the gap between OWL and relational databases“. In: *Journal of Web Semantics* 7.2 (2009), S. 74–89. ISSN: 1570-8268. DOI: https://doi.org/10.1016/j.websem.2009.02.001. URL: http://www.sciencedirect.com/science/article/pii/S1570826809000031.

[18] Yanfeng Shu. „A practical approach to modelling and validating integrity constraints in the Semantic Web“. In: *Knowledge-Based Systems* 153 (2018),

S. 29–39. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2018.04.021. URL: http://www.sciencedirect.com/science/article/pii/S0950705118301862.

[19] Hak-Jin Kim, Wooju Kim und Myungjin Lee. „Semantic Web Constraint Language and its application to an intelligent shopping agent“. In: *Decision Support Systems* 46.4 (2009). IT Decisions in Organizations, S. 882–894. ISSN: 0167-9236. DOI: https://doi.org/10.1016/j.dss.2008.12.004. URL: http://www.sciencedirect.com/science/article/pii/S0167923608002297.

[20] Sandra Bergner u. a. „Methodology for an Ontology-Driven Product Configuration Process“. In: 2016.

[21] TopQuadrant. *SPIN (SPARQL Inferencing Notation)*. 2020. URL: https://www.topquadrant.com/technology/sparql-rules-spin/ (besucht am 08. 03. 2020).

[22] W3C RDF Data Shapes Working Group. *Shapes Constraint Language (SHACL)*. 20. Juli 2017. URL: https://www.w3.org/TR/2017/REC-shacl-20170720/ (besucht am 04. 03. 2020).

[23] F. Ashby und Daniel Ennis. „Similarity measures“. In: *Scholarpedia* 2.12 (Jan. 2007), S. 4116. DOI: 10.4249/scholarpedia.4116.

[24] Paul Jaccard. „Lois de distribution florale dans la zone alpine“. In: *Bulletin de la Société vaudoise des sciences naturelles* 38 (Jan. 1902), S. 69–130. DOI: 10.5169/seals-266762.

[25] Amos Tversky. „Features of similarity“. In: *Psychological Review* 84.4 (1977), S. 327–352. ISSN: 19391471. DOI: 10.1037/0033-295X.84.4.327.

[26] Anil Chaturvedi, Paul E. Green und J. Douglas Caroll. „K-Modes Clustering“. In: *J. Classif.* 18.1 (Jan. 2001), S. 35–55. ISSN: 0176-4268. DOI: 10.1007/s00357-001-0004-3. URL: https://doi.org/10.1007/s00357-001-0004-3.

[27] Amir Ahmad und Lipika Dey. „A k-mean clustering algorithm for mixed numeric and categorical data“. In: *Data & Knowledge Engineering* 63.2 (2007), S. 503–527. ISSN: 0169-023X. DOI: https://doi.org/10.1016/j.datak.2007.03.016. URL: http://www.sciencedirect.com/science/article/pii/S0169023X0700050X.

[28] Dr. Thangavel Kuttiyannan. „Improved K-Modes for Categorical Clustering Using Weighted Dissimilarity Measure“. In: *Computational Intelligence - CI* 5 (Jan. 2009).

[29] Yiu-ming Cheung und Hong Jia. „Categorical-and-numerical-attribute data clustering based on a unified similarity metric without knowing cluster number“. In: *Pattern Recognition* 46.8 (2013), S. 2228–2238. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2013.01.027. URL: http://www.sciencedirect.com/science/article/pii/S0031320313000666.

[30] Doaa Ali, Ayman Ghoneim und Mohamed Saleh. „Data Clustering Method based on Mixed Similarity Measures“. In: Jan. 2017, S. 192–199. DOI: 10.5220/0006245601920199.

[31] Sébastien Harispe u. a. *Semantic Similarity from Natural Language and Ontology Analysis*. Bd. 8. Mai 2015. DOI: 10.2200/S00639ED1V01Y201504HLT027.

[32] Gilles Bisson. „Why and how to define a similarity measure for object based representation systems“. In: *Towards Very Large Knowledge Bases* (1995), S. 236–246.

[33] G. Rydzynski, A. Felic und C. Zirpins. „An ontology-based hierarchical clustering approach for decision support in mass customization environments“. In: *2017 2nd International Conference on Knowledge Engineering and Applications (ICKEA)*. Okt. 2017, S. 199–203. DOI: 10.1109/ICKEA.2017.8169929.

[34] Tim Berners-Lee, James Hendler und Ora Lassila. „The Semantic Web: A New Form of Web Content That is Meaningful to Computers Will Unleash a Revolution of New Possibilities“. In: *ScientificAmerican.com* (Mai 2001).

[35] TopQuadrant. *GitHub - TopBraid SHACL API*. 2020. URL: https://github.com/TopQuadrant/shacl (besucht am 04. 03. 2020).

[36] W3C SPARQL Working Group. *SPARQL 1.1 Overview*. 21. März 2013. URL: https://www.w3.org/TR/owl2-overview/ (besucht am 04. 03. 2020).

[37] Apache Software Foundation. *Apache Jena - What is Jena?* 2020. URL: https://jena.apache.org/about_jena/about.html (besucht am 04. 03. 2020).

[38] ECSS-E-ST-60-20C, Hrsg. *Star sensor terminology and performance specification*. 18. Juli 2008.

[39] N. Noy und Deborah Mcguinness. „Ontology Development 101: A Guide to Creating Your First Ontology“. In: *Knowledge Systems Laboratory* 32 (Jan. 2001).

[40] Bing Liu. *Web Data Mining*. Springer Berlin Heidelberg, 2007. ISBN: 978-3-540-37881-5. DOI: 10.1007/978-3-642-19460-3.

[41] Niklas Baumbach. *Product Configuration Similarity Measures*. Version 1.0.0. Juli 2020. DOI: 10.5281/zenodo.3930663. URL: https://doi.org/10.5281/zenodo.3930663.

A Anhang

A.1 Instanzen in der Ontologie

Tabelle A.1: Instanzen der Klasse *CommunicationProtocol*

URI	rdfs:label
st:Hdlc	HDLC
st:Mil	MIL-STD-1553
st:Uart	UART
st:Scdi	SCDI

Tabelle A.2: Instanzen der Klasse *Detector*

URI	rdfs:label
st:Has2	HAS2
st:Star1000	STAR1000

Tabelle A.3: Instanzen der Klasse *PowerSupply*

URI	rdfs:label	hasNominalVoltage	isInitialOn	isMilCompatible	hasLcl
st:PowerConverterA	A	52.0	✓	✗	✓
st:PowerConverterB	B	28.0	✓	✗	✓
st:PowerConverterC	C	32.0	✗	✓	✓
st:PowerConverterD	D	28.0	✓	✓	✓
st:PowerConverterE	E	28.0	✗	✓	✗
st:PowerConverterF	F	28.0	✗	✓	✓
st:PowerConverterG	G	52.0	✗	✓	✓
st:PowerConverterH	H	100.0	✗	✓	✓
st:PowerConverterI	I	28.0	✓	✓	✗
st:PowerConverterJ	J	28.0	✓	✓	✓

Tabelle A.4: Instanzen der Klasse *AstroAps* (bereits bestehende Konfigurationen)

URI	rdfs:label	hasPowerConverter	hasCommunicationProtocol	hasDetector	hasUpdaterate	qualificationLevel
st:Config1	1	G	MIL-STD-1553	STAR1000	10	2
st:Config2	2	A	MIL-STD-1553	STAR1000	10	2
st:Config3	3	D	MIL-STD-1553	STAR1000	10	2
st:Config4	4	E	MIL-STD-1553	HAS2	10	2
st:Config5	5	F	MIL-STD-1553	STAR1000	10	2
st:Config6	6	C	MIL-STD-1553	STAR1000	8	2
st:Config7	7	H	SCDI	HAS2	16	2
st:Config8	8	H	MIL-STD-1553	HAS2	10	2
st:Config9	9	B	HDLC	HAS2	10	2
st:Config10	10	J	MIL-STD-1553	STAR1000	10	2
st:Config11	11	H	MIL-STD-1553	STAR1000	10	2
st:Config12	12	I	UART	HAS2	10	2
st:Config13	13	H	MIL-STD-1553	STAR1000	8	2
st:Config14	14	G	UART	HAS2	10	2

A.2 Qualifizierte Konfigurationen

		Nennspannung in V			
		28	32	52	100
Protokoll	MIL-STD-1553	✓	✓	✓	✓
	UART	✓	✓	✓	
	HDLC	✓			
	SCDI			✓	

Tabelle A.5: Qualifizierte Kombinationen von Kommunikationsprotokollen und Nennspannungen

		Detektor	
		STAR1000	HAS2
Protokoll	MIL-STD-1553	✓	✓
	UART	✓	✓
	HDLC	✓	
	SCDI	✓	

Tabelle A.6: Qualifizierte Kombinationen von Kommunikationsprotokollen und Detektoren

		Updaterate in Hz		
		8	10	16
Protokoll	MIL-STD-1553	✓	✓	✓
	UART	✓	✓	
	HDLC		✓	
	SCDI			✓

Tabelle A.7: Qualifizierte Kombinationen von Kommunikationsprotokollen und Updateraten

A.3 SPARQL-Anfragen zur Beantwortung der Kompetenzfragen

```
SELECT DISTINCT ?configuration ?component
WHERE {
        ?configuration rdf:type st:AstroAps .
        ?property rdf:type owl:ObjectProperty .
        ?configuration ?property ?component .
}
```

Auflistung A.1: SPARQL-Anfrage zu Komponenten einer Konfiguration

```
SELECT DISTINCT ?configuration ?updateRate
WHERE {
        ?configuration rdf:type st:AstroAps .
        ?configuration st:hasUpdateRate ?updateRate .
}
```

Auflistung A.2: SPARQL-Anfrage zur Updaterate einer Konfiguration

```
SELECT DISTINCT ?configuration ?detector
WHERE {
        ?configuration rdf:type st:AstroAps .
        ?configuration st:hasDetector ?detector .
}
```

Auflistung A.3: SPARQL-Anfrage zum Detektor einer Konfiguration

```
SELECT DISTINCT ?configuration ?protocol
WHERE {
        ?configuration rdf:type st:AstroAps .
        ?configuration st:hasCommunicationProtocol ?protocol .
}
```

Auflistung A.4: SPARQL-Anfrage zum Kommunikationsprotokoll einer Konfiguration

```
SELECT DISTINCT ?configuration ?nominalVoltage
WHERE {
        ?configuration rdf:type st:AstroAps .
        ?configuration st:hasPowerConverter/st:hasNominalVoltage
           ?nominalVoltage .
}
```

Auflistung A.5: SPARQL-Anfrage zur Nennspannung des Spannungswandlers einer Konfiguration

```
SELECT DISTINCT ?configuration ?milCompatible
WHERE {
        ?configuration rdf:type st:AstroAps .
        ?configuration st:hasPowerConverter/st:isMilCompatible
           ?milCompatible .
}
```

Auflistung A.6: SPARQL-Anfrage zur MIL-STD-1553 Unterstützung des Spannungswandlers einer Konfiguration

```
SELECT DISTINCT ?configuration ?lcl
WHERE {
        ?configuration rdf:type st:AstroAps .
        ?configuration st:hasPowerConverter/st:hasLcl ?lcl .
}
```

Auflistung A.7: SPARQL-Anfrage zum LCL des Spannungswandlers einer Konfiguration

```
SELECT DISTINCT ?configuration ?qualificationLevel
WHERE {
        ?configuration rdf:type st:AstroAps .
        ?configuration st:qualificationLevel ?qualificationLevel .
}
```

Auflistung A.8: SPARQL-Anfrage zur Qualifikationsstufe einer Konfiguration

Selbstständigkeitserklärung

Ich erkläre, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.
Seitens des Verfassers bestehen keine Einwände die vorliegende Bachelorarbeit für die öffentliche Benutzung im Universitätsarchiv zur Verfügung zu stellen.

Jena, 05.06.2020