Entwicklung einer formalen Beschreibungssprache zur Modellierung und Nutzung von Operational Design Domains von automatisierten Fahrzeugen

Dissertation

zur Erlangung des Doktorgrades

der Ingenieurwissenschaften

vorgelegt von

Daniel Rohne

aus Hannover

genehmigt von der

Fakultät für Mathematik, Informatik und Maschinenbau

der Technischen Universität Clausthal

Tag der mündlichen Prüfung: 13.03.2025

Dissertation Technische Universität Clausthal, ISSE Dissertation 2025

Dekan
Prof. Dr.-Ing. Armin Lohrengel

Vorsitzender der Promotionskommission
Prof. Dr. Benjamin Säfken

Betreuer
Prof. Dr. Andreas Rausch

Gutachter
PD. Dr. Christoph Knieke
Prof. Dr.-Ing. Thomas Form

Für meine Familie.

Kurzfassung

Die Autoindustrie steht vor einem großen Wandel, der neue Chancen hinsichtlich der alltäglichen Mobilität eröffnet. Die Einführung von teil- und hochautomatisierten Fahrzeugen eröffnet jedoch nicht nur neue Möglichkeiten, sondern birgt auch Herausforderungen seitens der Entwicklung und Freigabe. Ein zentrales Element stellt hierbei die Operational Design Domain (ODD) dar. Diese ist für das sichere Funktionieren dieser Systeme unerlässlich, da präzise definiert werden kann, unter welchen Betriebsbedingungen (z.B. infrastrukturelle Einschränkungen wie Baustellen) hochautomatisierte Fahrzeuge zuverlässig operieren. Die eindeutige Definition dieser Betriebsbedingungen bildet die Basis für die Systemfähigkeiten und -grenzen und ist daher essenziell für die Entwicklung und das Testen der Technologie. Zudem wird durch diese eine solide Grundlage für regulatorische Standards geschaffen und eine öffentliche Kommunikation trägt zur Erhöhung der Akzeptanz bei. Vor diesem Hintergrund ist es umso wichtiger, dass die wachsende Komplexität der Fahrzeugsysteme durch eine sorgfältige Integration in den Entwicklungsprozess und präzisen Beschreibung der ODD adressiert wird. Bestehende Ansätze bieten hierfür zwar einen systematischen Ansatz zur Klassifizierung dieser Bedingungen, sind jedoch entweder schwer verständlich oder erfassen nicht vollständig die Komplexität realer Fahrbedingungen. Dabei kann eine unzureichende Beschreibung der Betriebsbedingungen die Sicherheit des Fahrzeugs beeinträchtigen.

Unter diesen Voraussetzungen wurde eine formale Beschreibungssprache für die ODD entwickelt, die sowohl leicht verständlich als auch technisch präzise ist. Die Sprache ist dabei so konzipiert, dass Betriebsbedingungen formal in Form von logischen Aussagen spezifiziert werden können. Diese sind modular organisiert, wodurch die Anpassung und Wiederverwendung einzelner Komponenten erleichtert wird. Zusätzlich wurden Modellierungspatterns abgeleitet, die die Anwendbarkeit der Sprache vereinfachen und Anwendern helfen, konsistente ODD-Beschreibungen zu erstellen. Aufbauend auf der Beschreibungssprache wurde ein Gesamtkonzept für den Entwurf und die Anwendung der ODD im Entwicklungsprozess entwickelt. Dieses unterstützt den Abgleich der ODD mit dem vorgesehenen Betriebsbereich des ADS und schafft unter anderem dadurch eine gemeinsame Wissensbasis für alle relevanten Stakeholder. Im weiteren Verlauf der Arbeit werden für dieses Konzept zwei werkzeuggestützte Implementierungen aufgezeigt. Abschließend wird demonstriert, wie das entwickelte Lösungskonzept anhand von drei Anwendungsfällen genutzt werden kann. Dafür wird zuerst auf eine Anwendung eingegangen, die es ermöglicht, zulässige geografische Straßennetzwerke (GeoNets) für hochautomatisierte Fahrzeuge zu definieren. Diese GeoNets erlauben es dem Fahrzeug, sicher innerhalb festgelegter Grenzen zu operieren. Zudem wird demonstriert, wie Fahrverhaltensweisen von Fahrzeugen anhand der ODD überprüft und als zulässig oder unzulässig klassifiziert werden können. Abschließend werden SOTIF-Szenarien anhand ihrer Relevanz gegenüber den definierten Betriebsbedingungen klassifiziert. Dies kann dazu genutzt werden, den passenden Scope der Absicherung zu bestimmen und somit kostenintensive Sicherheitsanalysen zu minimieren.

Vorwort

Diese Dissertation entstand in der Zeit von Juli 2019 bis September 2024 in der Entwicklung der Volkswagen AG am Standort Wolfsburg. Die erfolgreiche Fertigstellung dieser Arbeit wäre ohne die Unterstützung und das Vertrauen zahlreicher Personen nicht möglich gewesen.

Besonderer Dank gilt meinem Doktorvater Prof. Dr. Andreas Rausch für die Betreuung und Begutachtung meiner Arbeit. Von Beginn an haben die stets konstruktiven und zielführenden Diskussionen dazu beigetragen, diese Arbeit Stück für Stück zu schärfen. Maßgeblich hat dies zum Gelingen dieser Arbeit beigetragen. Doch nicht nur fachlich, sondern auch menschlich, war es immer eine Zusammenarbeit, die ich so jedem Doktoranden wünsche.

Ebenso möchte ich mich bei PD. Dr. Christoph Knieke und Prof. Dr.-Ing. Thomas Form für das Interesse an der Arbeit und der Übernahme des Gutachtens bedanken. Zudem danke ich Prof. Dr. Benjamin Säfken für die Übernahme des Vorsitzes der Prüfungskommission.

Weiterhin möchte ich meinen Kollegen bei Volkswagen aufrichtigen Dank aussprechen. Ihre fachliche Expertise, ihre wertvollen Anregungen sowie ihre konstruktive Kritik haben maßgeblich zur Qualität dieser Dissertation beigetragen. Kai Grünitz, Eugen Schneider und Ingo Swieter - durch euer initial in mich gesetztes Vertrauen und eure stetige Hilfsbereitschaft konnte ich diese Doktorarbeit erstellen. Dr. Lars Gehrke und Dr. Andreas Richter - eure engagierte Betreuung ist der Grund, dass es diese Doktorarbeit gibt. Neben den tief-technischen Diskussionen, zählt dazu auch der überfachliche Austausch und das besonders gute Arbeitsklima, welches wichtige Diskussionen erst ermöglicht und gefördert hat.

Mein besonderer Dank gilt meinen Eltern Elli und Uwe. Ihr habt mich ein Leben lang unterstützt und gefördert. Ohne euch wäre ich heute nicht der, der ich bin. Abschließend danke ich meiner Frau Regina, die mir stets Freiräume ermöglicht hat und mich zeitgleich ermutigt meine Ziele zu erreichen. Du bist mein Fels in der Brandung.

Meiner Familie widme ich diese Arbeit.

Ergebnisse, Meinungen und Schlüsse dieser Dissertation sind nicht notwendigerweise die der Volkswagen Aktiengesellschaft.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Abkürzungsverzeichnis

ADS Automated Driving System

ASIL Automotive Safety Integrity Level

BCM Business Case Management

COD Current Operational Domain

CPUC California Public Utilities Commission

DAG Directed Acylic Graph

DDT Dynamic Driving Task

DMV Department of Motor Vehicles

DSM Domänenspezifisches Modul

FMEA Failure Mode and Effects Analysis

FTA Fault Tree Analysis

HARA Hazard and Risk Analysis

MaaS Mobility as a Service

NHTSA National Highway Traffic Safety Administration

OD Operational Domain

ODD Operational Design Domain

SAE Society of Automotive Engineers

SBM System Boundary Module

SPOK Single Point of Knowledge

STPA Systems Theoretic Process Analysis

TaaS Transport as a Service

UCM Use Case Module

YAML YAML Ain't Markup Language

1. Einleitung

Zu Beginn der Einleitung wird in Kapitel 1.1 die allgemeine Motivation des Themas erläutert und die Notwendigkeit dieser Arbeit hergeleitet. Darauf aufbauend, wird in Kapitel 1.2 die Zielsetzung der Arbeit beschrieben. Abschließend beschreibt Kapitel 1.3 die Struktur der Arbeit und gibt einen Überblick über die Organisation der Inhalte.

1.1. Motivation

Die Autoindustrie steht vor großen Neuerungen. Neben der Elektrifizierung und Digitalisierung bestimmt vor allem die Entwicklung fahrerloser oder zumindest teilautomatisierter Fahrzeuge über den zukünftigen Erfolg der etablierten Hersteller [Bar16]. Dies bietet jedoch nicht nur Möglichkeiten für neue Geschäftsmodelle und Anbieter, sondern birgt zuallererst Herausforderungen bei der Automobilentwicklung.

Die konventionelle Fahrzeugentwicklung weist lange Produktlebenszyklen auf, da vor allem die Hardwareentwicklung, der Hardwarebau sowie der Hardwaretest sehr zeitintensiv sind. Darüber hinaus trägt das Zusammenspiel von Hard- und Software ebenfalls zu einer Verlängerung der Produktlebenszyklen bei. In der bisherigen Fahrzeugentwicklung werden fast nur monolithische Systeme verwendet, die aus einer Einheit von Hardware und Software bestehen. Klassischerweise zeigt sich dies in der Funktionsentwicklung, bei der für jede Funktion ein eigenes elektronisches Steuergerät designt, gebaut und getestet wird. Dieses ist dann meist speziell auf die auf die Funktion zugeschnitten, wodurch für jede andere Funktion ein neues Steuergerät entwickelt werden muss. Dies hat dazu geführt, dass im Laufe der Jahre die Funktionalität und die Anzahl von E/E-Komponenten und Steuergeräten stark gestiegen ist. [Bau20] Die Folge daraus ist eine wenig vernetzte Fahrzeugarchitektur, die gerade in Bezug auf die Updatefähigkeit ihre Limitierung aufweist. Nahezu jeder Softwarefehler einer Funktion muss in der Werkstatt behoben werden und eine Funktionserweiterung im Produktlebenszyklus ist kaum möglich, bedingt durch die speziell zugeschnittene Hardware. Hinsichtlich des Funktionsumfangs beschränken sich bisherige Systeme auf eine Steigerung des Komforts, indem sie den Fahrer unterstützen, oder auf die Sicherheit, indem sie potenziell gefährliche Situationen eigenverantwortlich regeln (z. B. Notbremsassistent).

Die Entwicklung zukünftiger Systeme unterscheidet sich grundlegend von den bisherigen Verfahren. Dies liegt vor allem am Übergang zur Elektromobilität, dem Wandel vom Automobilhersteller zum Mobilitätsdienstleister und der Entwicklung vom manuellen Fahren zum automatisierten „Bewegt werden“. Zusätzlich treiben gesellschaftliche Entwicklungen wie die Digitalisierung und das zunehmende Streben nach Nachhaltigkeit diese Transformation weiter voran, was erhebliche Auswirkungen auf die gesamte Unternehmenslandschaft und Produktpalette hat und sowohl die Produktentwicklung als auch die Produktion beeinflusst. [Sch21a]

In diesem Kontext spielen neuronale Netze eine zentrale Rolle, da sie für die Interpretation von heterogenen Sensordaten unverzichtbar sind, um sowohl moderne Fahrerassistenzsysteme als auch hochautomatisierte Fahrfunktionen zu ermöglichen. Obwohl solche Systeme bereits in sehr kontrollierten, homogenen Situationen wie dem Fahren auf Autobahnen oder kreuzungsfreien Bundesstraßen funktionieren, steht die Realisierung des hochautomatisierten Fahrens im urbanen Bereich noch aus. Dies liegt darin begründet, dass für die Realisierung eine Vielzahl technologischer Herausforderungen überwunden werden müssen. Speziell zählen dazu die Umsetzung des Verzichts auf einen Fahrzeugführer und die damit verbundene Verantwortungsübergabe. [BFH+18]

Erschwerend kommt hinzu, dass derartig komplexe Software, die eine Vielzahl von Sensoren, Aktoren und Steuergeräten verknüpft, die Entwicklung hochvernetzter und performanter Fahrzeugarchitekturen (bestehend aus Steuergeräten, Sensoren, Kommunikations- und Versorgungssystemen) bedingt. Entsprechend dieser weitreichenden Veränderung der Fahrzeuggrundtopologie, ist auch eine neue Art der Entwicklung von Fahrzeugsoftware notwendig.

War die bisherige Softwareentwicklung darauf ausgerichtet, eine Funktion auf einem Steuergerät zu realisieren, müssen nun sämtliche Softwarekomponenten in einem „System of Systems“ zusammenarbeiten. [Sch21a]

Vergleicht man nun die Entwicklung bisheriger Systeme mit der Entwicklung zukünftiger Systeme, so ergeben sich vor allem in Bezug auf das automatisierte Fahren signifikante Unterschiede, die zu einer hohen Komplexitätssteigerung in der Entwicklung führen. Vordergründig lässt sich festhalten, dass der Verzicht auf den Fahrer und die damit übertragene Verantwortung der sicheren Ausführung der Fahraufgabe, maßgeblichen Einfluss auf den Entwicklungs- und Testaufwand haben. Neben der reinen Steigerung des System- bzw. Funktionsumfangs, sind vor allem die Einbeziehung aller relevanten Umweltelemente eine Herausforderung (vgl. Abbildung 1.1).

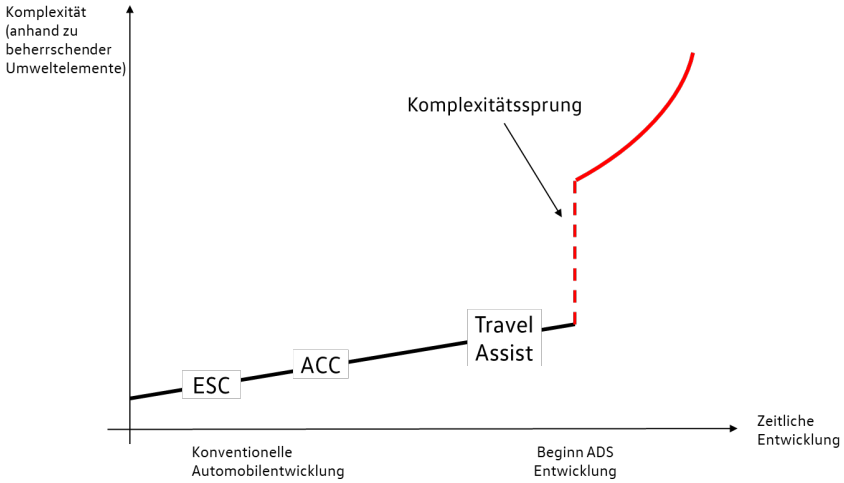

Abbildung 1.1: Visualisierung der Komplexitätssteigerung in der Funktionsentwicklung

Zugespitzt kann der Verlauf der Komplexitätszunahme für die bisherige Systementwicklung, gemessen anhand der Anzahl systemseitig zu beherrschender Umfeldelemente über der Zeit, als linearer Zusammenhang beschrieben werden. Für diesen Zusammenhang galt, dass eine Zunahme gut beherrschbar war und vor allem für etablierte Hersteller den Vorteil zuließ, auf etablierte Entwicklungen zurückzugreifen. Die Differenz zu bereits vorhandenem Wissen war nicht allzu groß und der Entwicklungshub damit gering. Der Beginn der Automated Driving System (ADS) -Entwicklung (vgl. Kapitel 2.1.1) stellt jedoch einen Bruch dieses Zusammenhangs dar. Die Veränderungen, die mit der Entwicklung eines automatisierten, vernetzten Systems einhergehen, weisen einen Komplexitätssprung auf, der keinen linearen Zusammenhang der Entwicklungskomplexität mehr erkennen lässt. Diese Entwicklung lässt sich auch durch die Anmeldung an Patenten im Bereich autonomes Fahren erkennen (vgl. Tabelle 1-1). Aus diesen wird ersichtlich, dass im Zeitraum zwischen 2010 und 2019 die jährliche Anzahl der Patente von 71 auf 342 gestiegen ist. Dieser Anstieg stellt fast eine Verfünffachung dar und unterstreicht den Komplexitätsanstieg aus Abbildung 1.1. [Deu22]

Tabelle 1-1: Entwicklung der Patentanmeldungen in Deutschland autonomes Fahren, nach [Deu22]

Technikgebiet	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019
Autonomes Fahren	71	91	72	91	103	120	150	209	268	342

Der Patentverlauf spiegelt bisher jedoch nur die Entwicklung von teilautonomen Systemen wider, da hoch- und vollautomatisierte Fahrzeuge noch keine Marktreife besitzen.

Die signifikante Zunahme der Komplexität in der Entwicklung von automatisierten und vernetzten Fahrzeugsystemen und der deutliche Anstieg der Patentanmeldungen im Bereich des autonomen Fahrens, verdeutlichen den fortlaufenden Wandel in der Automobilindustrie. Dieser Wandel erfordert

eine Anpassung der Entwicklungsstrategien und eine stetige Erweiterung des technologischen Know-hows. Insbesondere in diesem Zusammenhang spielt die Operational Design Domain (ODD, vgl. Kapitel 2.1.2) eine entscheidende Rolle, indem diese dazu verwendet wird, die spezifischen Bedingungen, unter denen ein automatisiertes Fahrzeugsystem zuverlässig funktioniert, zu definieren. Die Definition dieser Bedingungen ist von großer Wichtigkeit, da sie die Sicherheit und Zuverlässigkeit der Fahrzeuge in unterschiedlichen Fahrumgebungen gewährleistet. [Int23]

Die ODD umfasst dabei eine Vielzahl von Elementen, einschließlich geografischer Beschränkungen, Wetterbedingungen, Straßentypen und Verkehrsbedingungen. Um eine sichere und effektive Funktion der automatisierten Systeme zu ermöglichen, müssen alle diese Elemente sorgfältig analysiert, beschrieben und in die Entwicklung integriert werden. Angesichts der wachsenden Komplexität der Systeme und der zunehmenden Anzahl von Umweltelementen, die diese Systeme beherrschen müssen, wird die Definition einer präzisen ODD notwendig. Diese ermöglicht es den Entwicklern nicht nur, die Grenzen der Technologie zu verstehen und entsprechend zu gestalten, sondern bietet auch den regulatorischen Behörden eine Möglichkeit, die Funktionalität und Sicherheit dieser Technologien zu bewerten. [Int23]

Die Entwicklung und Einbindung einer klar definierten ODD ist damit nicht nur eine technische Notwendigkeit, sondern auch ein regulatorisches Erfordernis, das dazu beiträgt, das Vertrauen der Öffentlichkeit in die aufkommenden Technologien des autonomen Fahrens zu stärken. Es ist daher unabdingbar, die Relevanz der ODD in die aktuellen und zukünftigen Entwicklungen für automatisierte Fahrzeuge zu integrieren, um sowohl technologische als auch gesellschaftliche Akzeptanz sicherzustellen. [Int23]

1.2. Zielsetzung und Beitrag der Arbeit

Das vorherige Kapitel hat eine Vielzahl umfassender Herausforderungen, die durch die fortlaufenden Neuerungen in der Automobilindustrie entstehen, aufgezeigt. Insbesondere im Hinblick auf die Entwicklung von teil- bis vollautomatisierten Fahrzeugen (siehe Kapitel 2.1.1) besteht ein Bedarf, eine ODD zu beschreiben und in die Entwicklung einzubringen. Bisherige Ansätze zur Beschreibung von Betriebsbedingungen für solche Fahrzeuge sind entweder ungeeignet, zu komplex für eine breite Kommunikation, oder zwar einfach und leicht verständlich, jedoch in ihrer Funktionalität so eingeschränkt, dass nicht alle erforderlichen Use-Cases abgedeckt werden können.

Ein Ziel dieser Doktorarbeit ist es daher, eine Sprache zu entwerfen, die eine effektive Kommunikation der Systemfähigkeiten mit einer breiten Stakeholder-Basis, einschließlich Behörden und der allgemeinen Öffentlichkeit, ermöglicht. Gleichzeitig soll die Sprache den Anforderungen der technischen Entwicklung und Absicherung gerecht werden, indem sie beispielsweise die Ableitung von Szenarien und die Bestimmung fahrbarer Straßennetze unterstützt.

Dies wird erfüllt, indem eine leicht verständliche, aber gleichzeitig präzise domänenspezifische Sprache (DSL, Details siehe Kapitel 2.3.4) entwickelt wird, die speziell auf die modulare Beschreibung von ODDs zugeschnitten ist. Diese ermöglicht es, die komplexen Anforderungen und Bedingungen, unter denen automatisierte Fahrzeuge operieren, effektiv zu modellieren und die notwendigen Informationsbedarfe abzubilden. Ein Hauptmerkmal ist die Modularität, durch die einzelne Module in verschiedenen Kontexten wiederverwendet werden können. Des Weiteren werden erste Modellierungspatterns definiert, die die Anwendung der entwickelten DSL vereinfachen und Nutzern helfen, häufig auftretende Modellierungsszenarien effizient zu gestalten.

Zudem muss sichergestellt sein, dass die Sprache zur Beschreibung der ODD für Entwicklungs- und Testprozesse nutzbar ist und in diese integriert werden kann. Dafür muss ein Konzept für den Entwurf und die Anwendung der ODD im Entwicklungsprozess eines ADS definiert werden, welches nicht nur

den Abgleich der ODD mit dem vorgesehenen Betriebsbereich des ADS unterstützt, sondern auch relevante Inputs für die Entwicklung liefert und eine gemeinsame Wissensbasis für alle relevanten Stakeholder schafft. Im Fokus steht dabei die Frage nach einer passenden toolgestützten Umsetzung des Lösungskonzeptes.

Dies wird erreicht, indem das entwickelte Lösungskonzept genutzt wird, das auf der Verarbeitung und Analyse von Inputdaten, wie beispielsweise Kartendaten oder Daten aus Fahrzeugflotten, basiert. Anhand eines Abgleichs zwischen Inputdaten und definierter ODD, können relevante Outputs, wie zulässige oder unzulässige Betriebsbedingungen, bestimmt werden. Auf dieser Grundlage werden für die toolgestützte Umsetzung des Lösungskonzeptes zwei Konzepte für eine Werkzeugunterstützung vorgestellt, die die Integration und Anwendung der DSL in bestehende Entwicklungsprozesse unterstützen. Diese Werkzeuge erleichtern die Implementierung der Sprache und ihre Anwendung. Außerdem wird ein Ausblick gegeben, wie das Lösungskonzept für die Verarbeitung von unvollständigen Inputdaten (z.B. Fahrprotokolle) erweitert werden kann.

Abschließendes Ziel stellt die Demonstration der praktischen Nutzbarkeit der entwickelten DSL und der zugehörigen Werkzeuge anhand verschiedener Anwendungsfälle dar. Dafür wird anhand von drei Anwendungsfällen aufgezeigt, wie die Sprache und die Werkzeuge eingesetzt werden können, um die Entwicklungs- und Testprozesse von automatisierten Fahrsystemen zu unterstützen. Dazu zählen im Detail die Erstellung von zulässigen Straßennetzwerken, die Validierung von Fahrverhalten und die Relevanzbewertung von SOTIF Szenarien. Die in der Problemanalyse als notwendig identifizierte Prozessintegration über mehrere Entwicklungsstufen wird damit aufgezeigt.

1.3. Aufbau der Arbeit

Die Struktur dieser Doktorarbeit ist so konzipiert, dass sie schrittweise durch das komplexe Feld der ODD und der Entwicklung einer DSL führt. Die nachfolgende Übersicht erläutert den Aufbau und den Inhalt der einzelnen Kapitel (vgl. Abbildung 1.2).

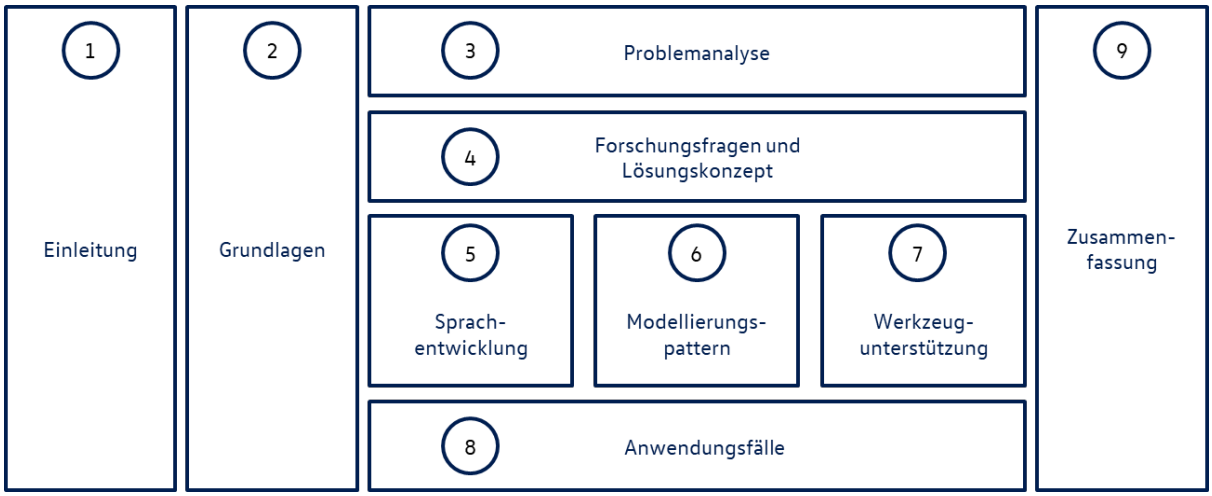

Abbildung 1.2: Überblick über den Aufbau der Arbeit

Kapitel 1: Einleitung Dieses Kapitel präsentiert und motiviert das Problem der Entwicklung eines ADS. Beginnend bei der traditionellen Automobilentwicklung wird die Zunahme der Komplexität hinsichtlich dieser neuen Technologie beschrieben. Darauf aufbauend wird die Bedeutung einer ODD konkretisiert und in diesem Zusammenhang die Notwendigkeit einer Modellierungssprache speziell für die ODD herausgestellt. Zudem wird die Zielsetzung der Arbeit vorgestellt, die sich auf die Schaffung einer sowohl technisch präzisen als auch leicht verständlichen DSL konzentriert.

Kapitel 2: Grundlagen Hier werden die notwendigen theoretischen Grundlagen vermittelt, die zum Verständnis der nachfolgenden Kapitel erforderlich sind. Dazu gehören Grundkonzepte der Fahrzeugautomatisierung und der ODD, relevante Terme aus dem Bereich der sicherheitskritischen Systeme sowie Grundlagen der Logik und der formalen Sprache.

Kapitel 3: Problemanalyse In diesem Kapitel werden die regulatorischen, wirtschaftlichen und technischen Rahmenbedingungen analysiert, die für die Entwicklung von ADS im Kontext der ODD relevant sind. Das Kapitel endet mit einer Auflistung von spezifischen Anforderungen, die die entwickelte Lösung erfüllen muss.

Kapitel 4: Forschungsfragen und Lösungskonzept Dieses Kapitel fasst das Problem zusammen und formuliert auf Basis des Standes der Wissenschaft die Forschungsfragen. Das Kapitel schließt mit der Beschreibung des Lösungskonzeptes ab.

Kapitel 5: Sprachentwicklung Die formale Definition der DSL zur Beschreibung der ODD wird detailliert erörtert. Dabei werden die Syntax und die Semantik der Sprache beschrieben und es wird erläutert, wie der Aufbau der Sprache zur Modularität beiträgt, ohne zusätzliche Komplexität zu verursachen.

Kapitel 6: Modellierungspatterns Dieses Kapitel beschreibt erste Ansätze für Modellierungspatterns, die die Nutzbarkeit der Sprache demonstrieren und den Aufbau einer ODD praktisch umsetzbar machen.

Kapitel 7: Werkzeugunterstützung Anhand zwei verschiedener Werkzeugunterstützungen wird aufgezeigt, wie das entwickelte Lösungskonzept implementiert werden kann.

Kapitel 8: Ausgewählte Anwendungsfälle der formalen Sprache zur Definition von Betriebsbedingungen. Die Nutzbarkeit und Effektivität der entwickelten DSL wird aufgezeigt, indem die Anwendung der Lösung anhand von drei verschiedenen Use-Cases beschrieben wird.

2. Grundlagen

Dieses Kapitel beschreibt die erforderlichen Grundlagen, die für das Verständnis der Arbeit notwendig sind. Eingangs wird in Kapitel 2.1 die verwendete Terminologie definiert. Darauf aufbauend, werden in Kapitel 2.2 die Grundlagen der Logik erklärt, die die Basis für formale Sprachen in Kapitel 2.3 bilden.

2.1. Terminologie der Arbeit

Diese Arbeit baut auf einigen grundlegenden Begriffen auf, die für das Verständnis elementar sind. Nachfolgend werden deshalb in Kapitel 2.1.1 die verschiedenen Stufen der Fahrzeugautomatisierung erklärt, sowie darauf aufbauend in Kapitel 2.1.2 das Konzept der ODD dargestellt. Kapitel 2.1.3 definiert weiterführende Konzepte, die für das Verständnis von sicherheitskritischen Systemen notwendig sind.

2.1.1. Stufen der Fahrzeugautomatisierung

In den letzten Jahren wurde eine Vielzahl von Definitionen zum Thema automatisiertes Fahren veröffentlicht, die helfen sollen, den allmählichen Übergang der Aufgaben von Menschen zu technischen Systemen zu beschreiben. Insbesondere haben die Beiträge der Society of Automotive Engineers (SAE) und der Bundesanstalt für Straßenwesen (BAST) wesentlich zur Harmonisierung der Fachterminologie beigetragen. Nachfolgend werden diese erläutert. [Fra18]

Die SAE klassifiziert die unterschiedlichen Stufen der Fahrzeugautomatisierung, basierend auf der Funktionalität und den Betriebsmerkmalen des ADS. Die unterschiedliche Einstufung folgt aus der Zuweisung der Aufgaben zwischen dem menschlichen Nutzer und dem System während der dynamischen Fahraufgabe (engl. dynamic driving task; DDT). Dementsprechend definiert die SAE die folgenden sechs Stufen:

- Stufe 0 (Keine Automatisierung): Der Fahrer führt alle Aspekte der DDT aus, selbst wenn er partiell durch aktive Warn- oder Sicherheitssysteme unterstützt wird.
- Stufe 1 (Fahrerassistenz): Das System dirigiert entweder die laterale oder die longitudinale Steuerung der Fahrzeugbewegung. Der Fahrer führt den Rest der DDT aus, überwacht das System dauerhaft und verbleibt in der Verantwortung.
- Stufe 2 (Teilautomatisierung): Das System übernimmt die laterale und longitudinale Fahrzeugbewegung, wobei der Fahrer weiterhin für die Erkennung und Reaktion auf Objekte und Ereignisse verantwortlich ist.
- Stufe 3 (Bedingte Automatisierung): Ab dieser Stufe übernimmt das System die Umfeldkontrolle/-überwachung und führt die gesamte DDT unter normalen Betriebsbedingungen (ODD) aus. Der Fahrer muss nicht ständig überwachen, sollte aber bereit sein, bei Aufforderung durch das System oder bei Systemausfällen einzugreifen.
- Stufe 4 (Hohe Automatisierung): Das System kann die gesamte DDT sowie die Übernahme im Notfall ohne menschliche Überwachung ausführen. Es kann die Kontrolle übernehmen, wenn der Nutzer nicht eingreift. Dies gilt für festgelegte Betriebsbedingungen (ODD).
- Stufe 5 (Vollautomatisierung): Das System führt die DDT und die Notfallübernahme unter allen Bedingungen selbstständig aus und unterscheidet sich dahingehend nicht mehr von einem menschlichen Fahrer. Es agiert unabhängig von den Betriebsbedingungen (ODD). [Sae21]

Grundlegend definiert die BAST die Stufen der Fahrzeugautomatisierung und die Nomenklatur sehr ähnlich zu den Stufen der SAE, mit der Ausnahme, das im Bereich der Hoch- und Vollautomatisierung leichte Unterscheidungen getroffen werden. [GWK+12] Der zentrale Unterschied besteht in der Definition der Rückfallebene im Notfall während des hochautomatisierten Fahrens. Die BAST beschreibt im Gegensatz zur SAE einen Zielzustand, indem der Fahrer sich vom Verkehrsgeschehen

abwenden darf und mit ausreichender Zeitreserve die Fahraufgabe wieder übernehmen können muss. Dies bringt mit sich, dass das Fahrzeug dabei nicht in der Lage ist, bei ausbleibender Fahrerübernahme aus jeder Ausgangssituation den risikominimalen Zustand selbstständig herbeizuführen. [Fra18]

Entsprechend den oben beschriebenen Definitionen zu den Stufen der Fahrzeugautomatisierung, leitet sich auch die Begrifflichkeit des automatisierten Fahrsystems (engl. automated driving system) ab. Dies kann definiert werden als:

Definition 1 - Automated Driving System (ADS): Das ADS beschreibt einen Verbund aus Hardware und Software, der gemeinsam in der Lage ist, die gesamte DDT dauerhaft auszuführen, unabhängig davon, ob sie auf eine ODD beschränkt ist. Der Begriff wird speziell verwendet, um die Automatisierungsstufen 3, 4 oder 5 zu beschreiben. [Sae21]

2.1.2. Definition der Operational Design Domain (ODD)

Aus den Stufen der Fahrzeugautomatisierung wird deutlich, dass ein wesentliches Merkmal der Unterscheidung nicht nur die Ausführung der DDT betrifft, sondern auch den Umfang der spezifischen Betriebsbedingungen, unter denen das System funktioniert, adressiert. Die Stufen 1-4 der Fahrzeugautomatisierung beschreiben diese Bedingungen explizit mit Hilfe der ODD. Im Gegensatz dazu, hat die Stufe 5 keine ODD, da diese unter allen Bedingungen funktionieren muss. Für die Definition der ODD folgt:

Definition 2 - Operational Design Domain: Die ODD definiert die spezifischen zulässigen und unzulässigen Betriebsbedingungen, für die ein automatisiertes System oder eine Funktion designt ist und innerhalb derer das System fehlerfrei funktioniert. Diese Bedingungen schließen das Vorhandensein oder das Fehlen bestimmter Umwelteinflüsse, geografische Einschränkungen, Tageszeiten sowie notwendige oder fehlende Verkehrs- oder Straßenmerkmale ein. [Sae21]

Basierend auf dieser Definition, können Beispiele je Automatisierungsstufe gegeben werden. Für eine Level 1 Adaptive Cruise Control (ACC) Funktion können die zulässigen Betriebsbedingungen beispielsweise auf hohe Geschwindigkeitsbereiche und gute Wetterbedingungen beschränkt sein. Für ein Level 4 ADS-System, welches designt ist, in einem Hafen Teile abzuholen und diese auf einer spezifischen Route zu einem 20 km entfernten Lager zu bringen, wird der Umfang der Betriebsbedingungen entsprechend größer. So könnten Limitationen in Bezug auf die Tageszeit bestehen, die Verkehrsinfrastruktur, die das System händeln können sollte oder auch in Bezug auf die zulässigen Wetterbedingungen. Auch wenn das Betriebsgebiet bereits beschränkt ist, bringt dies eine Vielzahl von Bedingungen mit sich, da das System die gesamte DDT verantwortet und die Umwelt entsprechend verstehen, überwachen und damit umgehen können muss. [Sae21]

Diese Beispiele verdeutlichen, dass die Automatisierungsstufen 1-4 auf eine begrenzte ODD beschränkt sind, die im Allgemeinen die Leistungsfähigkeit des jeweiligen Systems widerspiegelt. Wie im vorherigen Absatz exemplarisch beschrieben, ergeben sich Unterschiede hinsichtlich des Umfangs der ODD je nach Automatisierungsstufe (vgl. Abbildung 2.1).

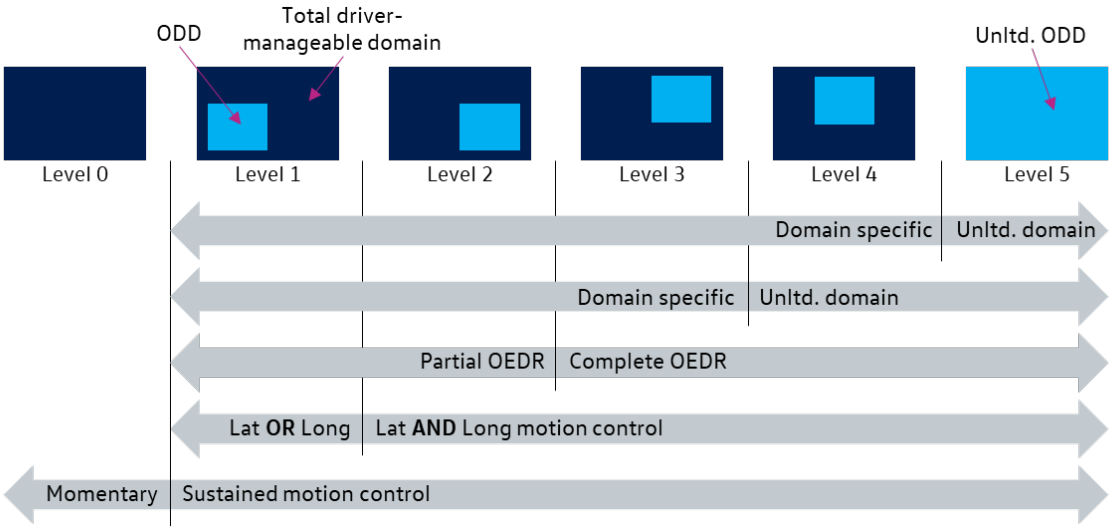

Abbildung 2.1: Einordnung der ODD-Relevanz nach Automatisierungsstufen, nach [Sae21]

Wichtig ist hierbei, dass die Funktion nur wie vorgesehen arbeiten kann, wenn alle in der ODD definierten Betriebsbedingungen durch die Designkriterien erfüllt sind und das System innerhalb dieser funktioniert. Trotz dessen kann nicht ausgeschlossen werden, dass das System in Situationen gebracht wird, in denen es außerhalb seiner ODD agiert oder ein Event auftritt, welches das System dazu bringt, nicht mehr die komplette DDT fehlerfrei ausführen zu können (vgl. Abbildung 2.2). In diesem Fall gilt für Automatisierungsstufen 3-5, dass ein DDT-Fallback durch das System selbst ausgeführt werden muss, um das System in einen risikominimalen Zustand zu bringen. [Sae21] Dieser ist definiert als:

Definition 3 - Minimal Risk Condition (MRC): Das MRC beschreibt einen stabilen, gestoppten Zustand, in den ein Nutzer oder ein ADS ein Fahrzeug bringen kann, nachdem der DDT-Fallback ausgeführt wurde. Der Zustand wird hergestellt, um das Risiko eines Unfalls zu verringen, wenn eine Fahrt nicht fortgesetzt werden kann, da sich das System bspw. außerhalb seiner ODD befindet (vgl. Abbildung 2.2). [Sae21]

Bei Level 3-Systemen kann der Fahrer den DDT-Fallback in vielen Fällen noch übernehmen, für Level 4- oder Level 5-Systeme muss das System selbst gewährleisten, einen MRC herzustellen. Dies könnte beispielweise durch eine Fahrt an den Straßenrand, das Einschalten der Warnblinkanlage, die Deaktivierung des Antriebssystems oder dem Rufen der Pannenhilfe erreicht werden.

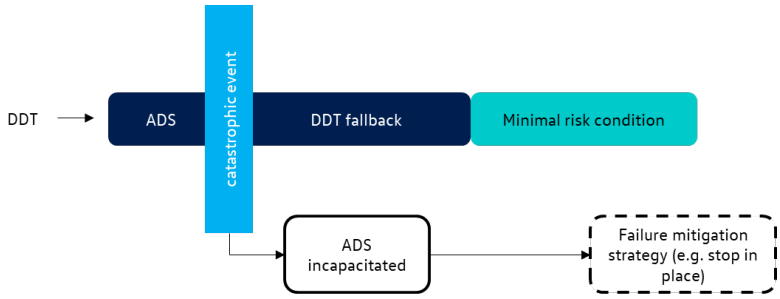

Abbildung 2.2: Herstellung des MRC bei SAE Level 4 Fahrzeugen, nach [Sae21]

Anhand der ODD können weitere Konzepte definiert werden, die damit im direkten Zusammenhang stehen. Während der Ausführung der DDT muss das ADS permanent sicherstellen, dass es sich noch innerhalb seiner definierten Systemgrenzen befindet. Dafür muss das ADS die Umgebung nahezu in

Echtzeit überwachen und sich kontinuierlich seiner Betriebsumgebung bewusst sein. Durch den Abgleich der extern vorhandenen Bedingungen mit der ODD kann das System entscheiden, ob es sich weiterhin innerhalb der definierten Grenzen befindet oder ein MRC hergestellt werden muss. [Int23] Diese externen Bedingungen werden ähnlich zur ISO 34503 definiert als Current Operational Domain (COD):

Definition 4 - Current Operational Domain (COD): Die COD definiert ein spezifisches Set von Umgebungsbedingungen an einem spezifischen Ort und zu einer spezifischen Zeit. [Int23]

Ähnlich zur ODD, können diese Bedingungen unter anderem Umwelt-, geografische und tageszeitliche Einschränkungen sowie das notwendige Vorhandensein oder Fehlen bestimmter Verkehrs- oder Straßenmerkmale beinhalten. Aggregiert man die spezifischen Sets von Umgebungsbedingungen aus der COD, erhält man eine zusammenfassende Darstellung der Informationen über ein Zeitintervall und/oder ein definiertes Gebiet. [Int23]

Diese aggregierte Form wird ähnlich zur ISO 34503 als Operational Domain bezeichnet und definiert als:

Definition 5 - Operational Domain (OD): Die OD beschreibt eine aggregierte Menge von Umgebungsbedingungen (COD) über ein Zeitintervall und/oder ein definiertes Gebiet. [Int23]

Diese Form stellt zusammenfassende Informationen über ein Gebiet anhand der CODs bereit. Auf Basis der COD und OD können mit Hilfe der ODD weitere Aussagen in Bezug auf ein Gebiet getroffen werden. Indem ein Abgleich zwischen diesen Informationen stattfindet, können die Straßen bestimmt werden, die anhand der Systemfähigkeiten befahren werden können. Dieses Sub-Set von einem Straßennetz, das mit dem ADS befahren werden kann, wird als GeoNet definiert.

Definition 6 – GeoNet: Zusammenhängendes Straßennetz, das auf Basis der ODD die vom ADS befahrbaren Straßen beschreibt.

Die Betriebsbedingungen der ODD repräsentieren eine Vielzahl unterschiedlicher Elemente, um die Komplexität der Domäne angemessen zu repräsentieren, innerhalb derer sich das System bewegen soll. Um die Vielzahl an Elementen zu strukturieren, gibt es bereits eine Vielzahl von Taxonomievorschlägen. [Int23, Bri20, Nat20, Cza18] Für alle diese Vorschläge gilt, dass Erweiterungen je nach Anwendungsfall möglich sind und diese Vorschläge nicht als fixes Konzept anzusehen sind. Obwohl die Taxonomie erweiterbar ist, müssen Erweiterungen, die im Konflikt mit festgelegten Attributen stehen, vermieden werden. Gleichermaßen gilt auch, dass, falls Attribute für die Betriebsbedingungen des ADS nicht relevant sind, diese in der Taxonomie nicht berücksichtigt werden müssen. Die [Int23] und die [Bri20] strukturieren die ODD auf der obersten Ebene anhand von drei Top-Level Attributen (vgl. Abbildung 2.3). Diese betreffen die Szenerie der Verkehrsinfrastruktur, die Umweltbedingungen und die dynamischen Elemente. [Int23, Bri20]

An diese Top-Level Attribute werden weitere Konzepte angehängt. Für die Szenerie umfasst dies räumlich fixierte Elemente der Verkehrsinfrastruktur (z. B. Straßeneigenschaften, Ampeln etc.). Das Attribut „Umweltbedingungen“ umfasst Wetter- und atmosphärische Bedingungen (einschließlich Konnektivität). Das Attribut „Dynamische Elemente“ umfasst die beweglichen Elemente der ODD und zielt unter anderem auf Verkehrsteilnehmer ab. [Int23]

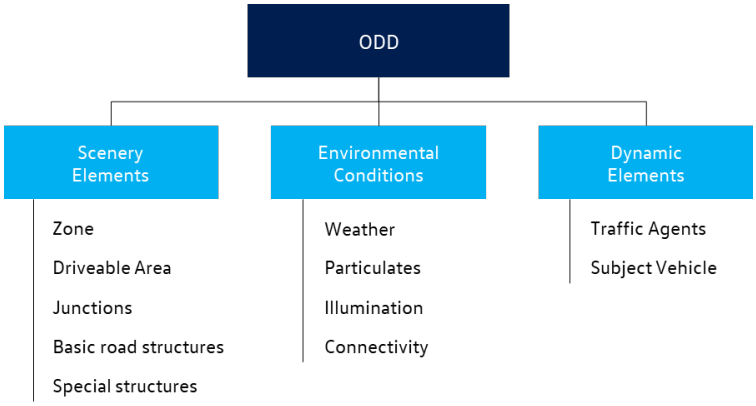

Abbildung 2.3: Top Level Taxonomie mit Attributen, nach [Int23]

Die [Nat20] dagegen definiert sechs Top-Level Elemente und orientiert sich damit mehr in die Richtung der Szenarienrepräsentation von Pegasus [Peg19] (vgl. Abbildung 2.4). Diese Top-Level Attribute umfassen physische Infrastruktur, operationelle Einschränkungen, Objekte, Konnektivität, Umweltbedingungen und Zonen. Auch diese stellen lediglich einen ersten Strukturierungsvorschlag dar und ergänzende Anpassungen müssen je nach Anwendungskontext und Domäne vorgenommen werden. [Nat20]

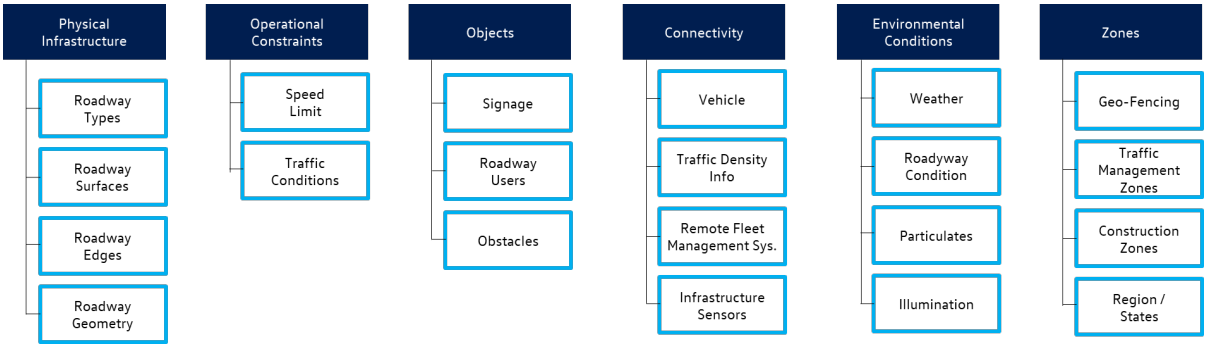

Abbildung 2.4: Klassifikation der ODD nach Top-Level Kategorien, nach [Nat20]

2.1.3. Terminologie für sicherheitskritische Systeme

Für diese Arbeit steht die Entwicklung eines ADS auf Basis einer ODD im Fokus. Dies stellt die Entwicklung eines sicherheitskritischen Systems dar, für dessen Verständnis weitere grundlegende Definitionen notwendig sind. Nachfolgend werden daher Begrifflichkeiten in alphabetischer Sortierung erklärt, die im Kontext der für diese Entwicklung maßgebenden Normen (ISO 26262 und ISO 21448) bedeutend sind.

Definition 7 - Automotive Safety Integrity Level (ASIL): Die ASIL ist eine Klassifikationsmethode aus der ISO 26262, die dazu dient, das notwendige Sicherheitsniveau anhand des Schweregrades, der Auftrittswahrscheinlichkeit und der Kontrollierbarkeit eines Risikos für elektrische und elektronische (E/E) Systeme festzulegen. Dabei werden vier Sicherheitsniveaus unterteilt, wobei D das höchste und A das niedrigste Sicherheitsniveau darstellt. Jede Stufe gibt den Grad der erforderlichen Sicherheitsmaßnahmen an, um ein akzeptables Risiko zu erreichen. [Int11h]

Definition 8 - Closed World Assumption: Die CWA besagt, dass alles, was nicht explizit als wahr bekannt ist, als falsch angenommen wird. [Rei78]

Dies wird oft in datenbankgestützten Anwendungen verwendet, in denen es unpraktisch oder unmöglich ist, jede mögliche Information ausdrücklich zu speichern. Insbesondere in relationalen Datenbanken, in denen Informationen explizit gespeichert sind, erlaubt die CWA die Annahme, dass jede nicht vorhandene Information falsch ist. Wenn zum Beispiel eine Datenbank, die Informationen über registrierte Autos speichert, kein Auto mit einer bestimmten Fahrzeugidentifikationsnummer (VIN) aufzeigt, nimmt die CWA an, dass kein solches Auto registriert ist. CWA erlaubt es zudem, Schlussfolgerungen aus dem Fehlen eines Beweises zu ziehen, was typisch für viele Logik-basierte Systeme und einige Programmieransätze ist. [Rei78] Die CWA wird oft als pragmatische Lösung für Probleme mit unvollständigen Daten verwendet. Die Annahme ist besonders in bestimmten Domänen sinnvoll, in denen die explizite Verneinung jeder nicht vorhandenen Information unpraktikabel wäre. [Rei78]

Definition 9 - funktionale Sicherheit: Die funktionale Sicherheit bezeichnet die Abwesenheit von unangemessenem Risiko auf Grund von Hazards, die durch fehlerhaftes Verhalten der E/E-Systeme verursacht wird. [Int11b]

Definition 10 - gefährliches Ereignis (engl. hazardous event): Ein gefährliches Ereignis beschreibt die Kombination aus einem Hazard und einem Szenario, welches während des Fahrzeuglebens auftreten kann. [Int11b]

Definition 11 - Hazard and Risk Assessment (HARA): Die HARA wird in der ISO 26262 als eine Methode definiert, die genutzt wird, um gefährliche Ereignisse zu identifizieren und zu kategorisieren, Sicherheitsziele festzulegen und die zugehörigen ASILs zu spezifizieren. Dies geschieht mit dem Ziel, die mit den Gefahren verbundenen Risiken zu verhindern oder zu mildern, um ein unangemessenes Risiko zu vermeiden. [Int11b]

Definition 12 - Harm: Harm beschreibt nach der ISO 26262 physische Verletzungen oder Schaden an der Gesundheit einer Person. [Int11b]

Definition 13 – Hazard: Ein Hazard beschreibt die potenzielle Quelle von Harm, welche von gefährlichem Systemverhalten oder falsch funktionierenden Items ausgeht. [Int11b]

Definition 14 - Open World Assumption: Die Open World Assumption (OWA) ist eine grundlegende Annahme in der Welt der semantischen Technologien und Datenbanken, die besagt, dass die Abwesenheit von Information nicht als Beweis für ihre Falschheit interpretiert werden sollte. [Rei78]

Diese Annahme wird besonders in Bereichen angewendet, in denen Daten potenziell unvollständig sind, wie zum Beispiel im Semantic Web oder in wissensbasierten Systemen. Unter der OWA wird also nicht davon ausgegangen, dass alles, was nicht bekannt oder nicht in der Datenbank vorhanden ist, falsch ist. Vielmehr wird angenommen, dass bestimmte Informationen einfach unbekannt sein können. Dies ermöglicht eine flexiblere Handhabung von Daten, da nicht vorausgesetzt wird, dass die Datenbank oder das Wissenssystem vollständig sind. [Rei78]

Definition 15 – Risiko: Ein Risiko beschreibt die Kombination der Auftrittswahrscheinlichkeit von Harm und der Schadensschwere von Harm. [Int11b]

Definition 16 - verbleibendes Restrisiko (engl. residual risk): Das verbleibende Restrisiko beschreibt das Risiko, welches auch nach der Anwendung aller Sicherheitsmaßnahmen verbleibt. [Int11b]

Definition 17 - Safety of The Intended Functionality: Sicherheit der beabsichtigten Funktionalität bedeutet das Fehlen eines unangemessenen Risikos aufgrund von Gefahren, die durch funktionale Mängel der beabsichtigten Funktionalität oder ihrer Umsetzung entstehen können. [Int19]

Definition 18 - sicherheitskritisches System: Ein sicherheitskritisches System ist ein System, dessen Versagen das Leben von Menschen gefährden, erhebliche wirtschaftliche Verluste verursachen oder weitreichende Umweltschäden nach sich ziehen kann. [Kni02] In einem sicherheitskritischen System hat die Sicherheit des Systems Vorrang vor anderen Entwurfszielen [Alu15].

Definition 19 – System: Ein System wird als eine Zusammenstellung von Elementen betrachtet, die miteinander interagieren und spezifischen Regeln folgen, um eine bestimmte Funktion oder einen Satz von Funktionen innerhalb einer bestimmten Umgebung zu erfüllen. Ein Element kann eine Hardwareausrüstung, ein Softwareprogramm oder eine Person umfassen. [Iso10]

Definition 20 - Szenario: Ein Szenario ist eine zeitliche Sequenz von Szenen und Ereignissen, die den Fortschritt von einer Szene zur nächsten ermöglichen. Es kann auch die Ziele und Werte des Systems spezifizieren. [Int19, UMR+ 15]

Da eine ODD-Definition testbar sein muss, spielen ODD-Attribute und deren Definition eine zentrale Rolle im szenariobasierten Testen. ODD und Szenarien sind dabei zwei unterschiedliche, aber verwandte Konzepte: Während die ODD die Betriebsbedingungen des ADS beschreibt, beschreibt ein Szenario das Verhalten der Verkehrsteilnehmer und das gewünschte Verhalten des Ego-Fahrzeugs innerhalb oder außerhalb einer ODD.

Die ODD-Definition wird als Eingabe für das szenariobasierte Testen gemäß ISO 34502 verwendet. [Int22] Ein erster Schritt im Verifikations- und Validierungsprozess eines ADS ist die Analyse der ODD, um Testszenarien zu erstellen. Der nächste Schritt besteht darin, das ADS-Verhalten mithilfe einer Verhaltensbibliothek zu testen, wobei auch unerwünschte Verhaltensweisen einbezogen werden können. Eine Instanz der ODD, kombiniert mit dem gewünschten Verhalten und der Beschreibung des Verkehrsverhaltens, liefert eine Szenariodefinition. Diese Szenarien können in funktionale, abstrakte, logische und konkrete Szenarien detailliert werden. [Peg15] Die ODD-Definition dient als Kriterium, um festzustellen, ob Testszenarien innerhalb, außerhalb oder an der Grenze einer ODD liegen. Es ist wichtig, Szenarien außerhalb der ODD zu testen, um Missbrauch zu verhindern. Ein Vergleich zwischen Testszenarien und der ODD-Definition hilft bei der Analyse der Testraumabdeckung. Abschließend ist es relevant herauszustellen, dass Szenarien im Vergleich zu ODD-Definitionen komplexer sind und zusätzliche Konstrukte wie Ereignisse und Manöver umfassen. [Int23]

Definition 21 - Use-Case: Ein Anwendungsfall definiert für ein System unter Entwicklung mindestens ein Szenario, in dem das System operieren soll, inklusive seines funktionalen Umfangs, des gewünschten Verhaltens und der Systemgrenzen. [Int19, UMR+15]

2.2. Grundlagen der Logik

In diesem Kapitel werden die Grundlagen der Aussagenlogik, der Beschreibungslogik und der typisierten Prädikatenlogik erster Ordnung vorgestellt. Diese Grundlagen werden benötigt, um im späteren Verlauf der Doktorarbeit die passende Logik für die Beschreibungssprache der ODD auszuwählen.

2.2.1. Aussagenlogik

Die Aussagenlogik stellt die elementarste Theorie der formalen Logik dar und fokussiert auf die Verknüpfung einfacher Aussagen zu komplexen Aussagen. [GJ90] Gegenstand der Definition sind hierbei atomare Aussagen und die Beziehungen, die zwischen diesen Aussagen bestehen. Die Aussagenlogik weist dabei einen einfachen Aufbau und eine klare Struktur auf, hat jedoch im Vergleich zu anderen Logiken nur eine begrenzte Ausdrucksstärke. Trotz dessen ist die Bedeutung der Aussagenlogik fundamental. Zum einen ist diese wesentlich für den Hardwareentwurf, indem sich das

Verhalten kombinatorischer Schaltungen anhand aussagenlogischer Formeln abbilden lässt. Zum anderen bildet die Aussagenlogik die Basis für alle anderen Logiken und dient somit als kleinster gemeinsamer Nenner. [Hof18]

Zur Erklärung der Aussagenlogik wird zuerst die Syntax beschrieben, die definiert, nach welchen Regeln aussagenlogische Ausdrücke (nachfolgend als Formeln bezeichnet) aufgebaut werden dürfen. Für die Syntax gilt, dass die Menge der aussagenlogischen Formeln über dem Variablenvorrat V = $\{A_1;A_2,\ldots\}$ rekursiv definiert ist:

- 0 und 1 sind Formeln.
- Jede Variable A_i €€ V ist eine Formel.
- Sind F und G Formeln, dann sind es auch ($\neg F$), (F ∧ G), (F ∨ G), (F → G), (F ↔ G), (F ↮ G). [Hof18]

Anhand dessen lassen sich die wichtigsten Elemente der Syntax definieren.

Definition 22 - Atomare Formeln Aussagenlogik: Atomare Formeln sind die einfachsten Aussagen und können nicht weiter zerlegt werden. Normalerweise werden diese durch Variablen oder Konstanten repräsentiert, die eigenständige Wahrheitswerte (1 für wahr und 0 für falsch) annehmen können.

Definition 23 - Logische Operatoren Aussagenlogik: Die logischen Operatoren verbinden atomare Aussagen zu komplexeren Aussagen. Jeder Operator hat dabei spezifische Regeln:

- Negation (¬): Negiert den Wahrheitswert einer Aussage.
- Konjunktion (∧): Liefert wahr, wenn beide verbundenen Aussagen wahr sind.
- Disjunktion (∨): Liefert wahr, wenn mindestens eine der verbundenen Aussagen wahr ist.
- Implikation (→): Liefert falsch nur dann, wenn die erste Aussage wahr und die zweite falsch ist.
- Äquivalenz (↔): Liefert wahr, wenn beide Aussagen den gleichen Wahrheitswert haben.

Definition 24 - Formelkonstruktionen Aussagenlogik: Komplexere Formeln werden durch rekursive Anwendung der Operatoren auf einfache Formeln oder das Ergebnis anderer Operationen gebildet.

Definition 25 - Klammerung und Priorität Aussagenlogik: Klammern werden verwendet, um die Auswertungsreihenfolge der Operatoren klarzustellen. Operatoren können unterschiedliche Bindungsstärken haben, wobei üblicherweise die Negation die stärkste Bindung hat, gefolgt von Konjunktion, Disjunktion, Implikation und Äquivalenz.

Diese Formeln sind anfänglich noch eine Folge von bedeutungsleeren Symbolen, die nach festgelegten Regeln kombiniert werden dürfen. Erst die Semantik gibt diesen Symbolen eine Bedeutung und spezifiziert, wie die Zeichen und Symbole einer Logikformel zu interpretieren sind. Für die Semantik bildet die Interpretation die Basis.

Definition 26 - Interpretation Aussagenlogik: Eine Interpretation weist jeder atomaren Formel in einer logischen Aussage einen Wahrheitswert zu und wird deshalb auch als Belegung bezeichnet. Dies bildet die Grundlage dafür, den Wahrheitswert komplexerer Formeln zu bestimmen. [Hof18]

Die Interpretation bildet die Grundlage für die weitere formale Definition der Semantik. Für diese gilt, dass wenn F und G aussagenlogische Formeln und I eine Interpretation sind, die Semantik der Aussagenlogik durch die induktiv definierte Modellrelation ⊨ gegeben ist:

$$I \models 1$$

$$I \not\models 0$$

$$I \vDash A_i : \Leftrightarrow I(A_i) = 1$$

$$I \vDash (\neg F) : \Leftrightarrow I \nvDash F$$

$$I \vDash (F \wedge G) : \Leftrightarrow I \vDash F \; und \; I \; \vDash G$$

$$I \vDash (F \vee G) : \Leftrightarrow I \vDash F \; oder \; I \; \vDash G$$

$$I \vDash (F \rightarrow G) : \Leftrightarrow I \nvDash F \; oder \; I \; \vDash G$$

$$I \vDash (F \leftrightarrow G) : \Leftrightarrow I \vDash F \; genau \; dann, wenn \; I \; \vDash G$$

$$I \vDash (F \nleftrightarrow G) : \Leftrightarrow I \nvDash (F \leftrightarrow G)$$

Eine Interpretation I mit $I \vDash F$ heißt Modell für F. [Hof18] Des Weiteren gilt, dass jede aussagenlogische Formel F mit n Variablen als Boolesche Funktion betrachtet werden kann. Dafür gilt:

$$f^F(b_1, \ldots, b_n) = \begin{cases} 1 \; falls \; I \vDash F \\ 0 \; falls \; I \nvDash F \end{cases}$$

Das Ganze basiert auf einem diskreten Definitionsbereich, aufgrund dessen sich eine n-stellige Boolesche Funktion in Form einer Wahrheitstabelle darstellen lässt. Dafür werden alle möglichen Kombinationen der Eingangsvariablen A_1,…A_n zusammen mit dem zugeordneten Funktionswert zeilenweise aufgelistet. Für die beschriebenen aussagenlogischen Operatoren folgen daraus die Wahrheitstabellen (siehe Abbildung 2.5). [Hof 2018]

Negation

	A	¬A
0	0	1
1	1	0

Konjunktion

	A	¬A	$A \wedge B$
0	0	0	0
1	0	1	0
2	1	0	0
3	1	1	1

Disjunktion

	A	¬A	$A \vee B$
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	1

Implikation

	A	¬A	$A \rightarrow B$
0	0	0	1
1	0	1	1
2	1	0	0
3	1	1	1

Äquivalenz

	A	¬A	$A \leftrightarrow B$
0	0	0	1
1	0	1	0
2	1	0	0
3	1	1	1

Anitvalenz

	A	¬A	$A \nleftrightarrow B$
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	0

Abbildung 2.5: Wahrheitstabellen der Aussagenlogik, nach [Hof18]

Darauf aufbauend gilt für zusammengesetzte Ausdrücke, das ausgehend vom Basisterm zunächst die Teilformeln und dann der Gesamtausdruck ausgewertet werden. Je nach Auswertung kann daraus für die Erfüllbarkeit abgeleitet werden, dass eine aussagenlogische Formel F:

- Erfüllbar heißt, falls F mindestens ein Modell besitzt
- Unerfüllbar heißt, falls F kein Modell besitzt
- Allgemeingültig heißt, falls ¬F unerfüllbar ist. [Hof18]

2.2.2. Beschreibungslogik

Beschreibungslogiken sind formale Rahmenwerke zur Wissensrepräsentation, die genutzt werden, um systematisch Wissen über eine Welt oder ein spezifisches Anwendungsgebiet zu strukturieren und zu verarbeiten. Sie dienen dazu, Konzepte und Rollen innerhalb eines definierten Bereichs zu definieren und verwenden diese Strukturen, um die Eigenschaften von Objekten oder Individuen innerhalb einer Domäne zu beschreiben. [Fil09]

Im Detail besteht die Beschreibungslogik aus Konzeptkonstruktoren und einer Wissensbasis. Die Konzeptkonstruktoren legen die Ausdrucksmittel fest, die zur Formulierung von Fakten erlaubt sind. Die Wissensbasis wiederum besteht aus zwei Hauptkomponenten, der Terminologischen Box (T-Box) und der Assertionalen Box (A-Box). Die T-Box beinhaltet das Wissen über die Konzepte einer Domäne, wodurch das Vokabular der Anwendungsgebiete festgelegt wird. Die A-Box hingegen beinhaltet das Wissen über die Entitäten oder Instanzen dieser Konzepte sowie deren Beziehungen untereinander und repräsentiert den aktuellen Zustand der modellierten Welt.

Für die Konzeptkonstruktoren der meisten Beschreibungslogiken gilt, dass diese verschiedene Konzepte und Rollen bieten. Dazu gehören:

- Atomare Konzepte (A): Dies sind die grundlegenden, nicht weiter unterteilbaren Konzepte.
- Top-Konzept ($\top$): Ein universelles Konzept, das alle Individuen der Domäne umfasst.
- Bottom-Konzept ($\bot$): Ein spezielles Konzept, das keine Individuen enthält.
- Atomare Negation ($\neg A$): Das logische Gegenteil eines atomaren Konzepts, schließt alle Individuen aus, die zu A gehören.
- Konzeptkonjunktion ($C \sqcap D$): Ein Konzept, das Individuen umfasst, die sowohl zu C als auch zu D gehören.
- Universelle Quantifizierung ($\forall r.C$): Ein Konzept, das alle Individuen umfasst, bei denen alle Beziehungen vom Typ R zu Individuen führen, die zum Konzept C gehören.
- Existenzielle Quantifizierung ($\exists r.T$): Ein Konzept, das Individuen umfasst, die mindestens eine Beziehung vom Typ R zu einem Individuum haben, das zum Konzept T gehört. [Uni 2014a]

Basierend auf diese allgemeinen Definitionen, gilt für die Syntax der T-Box

- Konzeptinklusion ($C \subseteq D$): Dies ist ein fundamentales syntaktisches Element der T-Box, das besagt, dass alle Instanzen des Konzepts C auch als Instanzen des Konzepts D gelten. Diese Form der Inklusion wird oft als "C impliziert D" interpretiert.
- Äquivalenz ($C \equiv D$): Verwendet als Abkürzung für $C \subseteq D$ und $D \subseteq C$, was bedeutet, dass C und D identisch sind in Bezug auf ihre Instanzen. [Uni 2014a]

Für die Semantik der T-Box folgt

- Eine Interpretation I erfüllt die Konzeptinklusion $C \subseteq D$ genau dann, wenn die Menge der durch C interpretierten Elemente eine Teilmenge der durch D interpretierten Elemente ist ($C^I \subseteq D^I$).
- Ein Modell von T ist eine Interpretation, die alle Konzeptinklusionen in der T-Box erfüllt. [Uni14b]

Für die Syntax der A-Box folgt:

- Konzeptassertion (C(a)): stellt fest, dass das Individuum a ein Mitglied des Konzepts C ist.
- Rollenassertion (r(a, b)): besagt, dass eine Beziehung r zwischen den Individuen a und b besteht. [Uni 2014a]

Für die Semantik der A-Box gilt:

- Eine Interpretation I erfüllt eine Konzeptassertion C(a) genau dann, wenn das Individuum a in der Interpretation von C enthalten ist ($a \in C^I$).
- Eine Interpretation erfüllt eine Rollenassertion r(a, b) genau dann, wenn das Paar (a, b) in der Interpretation von r enthalten ist ($(a, b) \in r^I$).
- Eine Interpretation ist ein Modell der A-Box, wenn sie alle in der A-Box enthaltenen Assertionen erfüllt. [Uni14b]

Zusätzlich gilt für Modelle der A-Box, dass diese zusätzliche Assertionen wahr machen dürfen, die in der A-Box nicht explizit vorkommen, was darauf hinweist, dass das durch A-Boxen repräsentierte Wissen unvollständig sein kann.

Die Beschreibungslogik unterscheidet sich von der Aussagenlogik, wenngleich zum Teil dieselben Konzepte genutzt werden. Die Syntax ist dahingehend anders, dass Beschreibungslogiken Konzepte und Rollennamen aufweisen, anstatt Variablen wie die Aussagenlogik (vgl. Kapitel 2.2.1). Auch die Semantik ist unterschiedlich. Aussagenlogik basiert auf Wahrheitswertzuweisungen, während die Beschreibungslogik Interpretationen bzw. relationale Strukturen nutzt. Im Detail würde die Aussagenlogik genau dann der Beschreibungslogik entsprechen, wenn diese keine Quantoren und Rollennamen hätte sowie lediglich die Interpretation mit nur einem Element. [Uni14a] Ebenfalls gibt es Parallelen zur Prädikatenlogik. Die theoretische Basis von Beschreibungslogiken bildet oft eine Untermenge der Prädikatenlogik erster Stufe, ist jedoch im Gegensatz dazu immer entscheidbar (vgl. Kapitel 2.2.3). Dies ermöglicht es, aus vorhandenem Wissen neues Wissen zu gewinnen und Schlussfolgerungen zu ziehen. [Fil09] Die Anwendung der Beschreibungslogik wird in Kapitel 5.5.1.2 demonstriert.

2.2.3. Typisierte Prädikatenlogik erster Ordnung

Die Prädikatenlogik erster Stufe basiert auf der Aussagenlogik und erweitert diese um quantifizierende und mehrstellige Prädikate. Dies ermöglicht ein umfassenderes logisches Schließen, wodurch komplexere logische Zusammenhänge, wie jene im klassischen Beispiel zu Sokrates, abgebildet werden können. Dieses Beispiel beschreibt:

- Sokrates ist ein Mensch.
- Alle Menschen sind sterblich.
- Folglich ist Sokrates sterblich. [Hof18]

Hierbei zeigt sich, dass die Eigenschaft "Mensch zu sein" von der konkreten Person abhängt, die als Argument eingesetzt wird. Das stellt einen klassischen Fall für Prädikate dar. Diese Logik kann erneut erweitert werden, indem eine explizite Typisierung von Objekten, Variablen, Funktionen und Prädikaten eingeführt wird. Bezeichnet wird dies dann als typisierte Prädikatenlogik erster Ordnung [Smu12]. Durch diese Typisierung wird eine Argumentation über die Domäne anhand von Typen anstatt individueller Objekte ermöglicht. Dabei wird jedem Objekt der betrachteten Domäne ein Typ zugeordnet. [BHS07] Entsprechend dieser Grundlagen können weitere Definitionen für die Logik getroffen werden.

Definition 27 - Datentyp: Ein Datentyp ist eine Kategorie von Dingen mit gemeinsamen Merkmalen.

In typisierter Prädikatenlogik erster Ordnung werden die Datentypen in einer Typhierarchie organisiert.

Definition 28 - Datentyphierarchie: Eine Datentypenhierarchie ist definiert als Quadrupel $\mathbb{T}=(\mathcal{T},\mathcal{TA},\mathcal{TD},\sqsubseteq)$ bestehend aus:

- $\mathcal{T}$, einer Menge von Datentypen
- $\mathcal{T}_A$, einer Untermenge von abstrakten Datentypen
- $\mathcal{T}_D$, einer Untermenge von dynamischen Datentypen
- $\sqsubseteq$, eine Relation, die die Untertyp Beziehung zwischen Datentypen bezeichnet.

Dabei gilt $\mathcal{T}_A \cup \mathcal{T}_D = \mathcal{T}$ und $\mathcal{T}_A \cap \mathcal{T}_D$. [BHS07]

Zur weiteren Definition der Syntax ergeben sich folgende Definitionen:

Definition 29- Signatur: Für eine gegebene Datentyphierarchie $\mathbb{T}=(\mathcal{T},\mathcal{T}A,\mathcal{T}D,\sqsubseteq)$ wird die Signatur als Tupel $\Sigma=(V,F,P,\alpha)$ definiert, wobei:

- V eine Menge von Variablen ist.
- F eine Menge von Symbolen für Funktionen für die die Anzahl von Parametern $n \geq 0$ ist.
- P eine Menge von Symbolen für Prädikate mit der Anzahl von Parametern $n \geq 0$ ist.
- α eine Typisierungsfunktion ist, die jeder Variable und jedem Funktions- und Prädikatssymbol einen entsprechenden Typ zuweist. [BHS07]

Teil der Signatur ist außerdem die Typisierungsfunktion, welche definiert ist als:

***Definition 30* - Typisierungsfunktion:** Die Typisierungsfunktion α weist jeder Variablen $x \in V$, jedem Funktionssymbol $f \in F$ und jedem Prädikatssymbol $p \in P$ einen entsprechenden Typ zu.

Darauf aufbauend können die Terme und Formeln definiert werden. Um eine Formel in typisierte Prädikatenlogik erster Ordnung interpretieren und auswerten zu können, muss ein sogenanntes Modell definiert werden. Dieses Modell umfasst die Objekte, Funktionen und Prädikate, über die die Terme und Formeln interpretiert und ausgewertet werden. Um die Terme interpretieren zu können, muss klar sein, wie Funktionen und Variablen in typisierter Prädikatenlogik erster Ordnung interpretiert sind. Dafür wird definiert, wie Funktionen interpretiert sind. Darauf aufbauend kann definiert werden, wie die Prädikate interpretiert sind, Variablen belegt werden sowie Terme und Formeln interpretiert werden. Weiterführend Details dazu finden sich in [BHS07].

2.3. Grundlagen der formalen Sprache

Das nachfolgende Kapitel definiert die Grundlagen in Bezug auf formale Sprachen. Dafür werden zuerst in Kapitel 2.3.1 die Themen Sprache und Grammatik erklärt. Kapitel 2.3.2 baut darauf auf und definiert die verschiedenen Typklassen der Sprache nach Chomsky. Im nachfolgenden Kapitel 2.3.3 wird auf die kontextfreie Sprache eingegangen, ehe in Kapitel 2.3.4 domänenspezifische Sprachen betrachtet werden.

2.3.1. Sprache und Grammatik

Formale Sprachen beschäftigen mit der Struktur und den Regeln, die in Computersystemen genutzt werden, um Anwendungen präzise zu beschreiben. Dazu besteht die formale Sprache aus einem Alphabet, dass eine endliche Menge an Zeichen beinhaltet. Aus dieser können Wörter gebildet werden. Zusätzlich wird eine Syntax festgelegt, die definiert, welche Kombinationen der Zeichen gültige Ausdrücke einer Sprache sind. Die Bedeutung dieser durch die Syntax erzeugten Ausdrücke wird dann durch die Semantik beschrieben.

Grammatiken spielen dabei eine zentrale Rolle, da diese es ermöglichen, Wörter einer Sprache anhand festgelegter Regeln zu konstruieren, anstatt alle möglichen Sätze aufzulisten. In der natürlichen Sprache (z.B. Deutsch), werden durch eine Abfolge von Artikeln, Adjektiven und Substantiven Sätze

gebildet. Als Beispiel kann ein Auszug aus der Grammatik der deutschen Sprache genutzt werden, um zu verstehen, wie aus Platzhaltern wie <Satz> und <Subjekt> vollständige Sätze entstehen:

- <Satz> ! <Subjekt> <Prädikat> <Objekt>
- <Subjekt> ! <Artikel><Adjektiv><Substantiv>
- <Artikel> ! Der | Die | Das
- <Adjektiv> ! große | kluge | mutige
- <Substantiv> ! Löwe | Drache | Ritter | Einhorn | Zauberer
- <Prädikat> ! entdeckt | bewacht | schützt
- <Objekt> ! Schatz | Burg | Wald [Hof18]

Innerhalb der formalen Sprachen unterscheidet man zwischen Terminalen und Nonterminalen. Die Terminale bilden die endgültigen Bestandteile des Alphabets (z.B. „Der," „große," „Löwe"). Die Nonterminale (auch als Nicht-Terminal bezeichnet) hingegen dienen als Platzhalter für komplexe Strukturen (z.B. <Satz>, <Subjekt>). [Hof18] Für die beschriebene Beispielgrammatik erhalten wir durch diese Unterteilung die nachstehende Einteilung:

- Terminal:
  Der, Die, Das, große, kluge, mutige, Löwe, Drache, Ritter, Einhorn, Zauberer, entdeckt, bewacht, schützt, Schatz, Burg, Wald.
- Nonterminal:
  <Satz>, <Subjekt>, <Artikel>, <Adjektiv>, <Substantiv>, <Prädikat>, <Objekt>.

Dem Nonterminal <Satz> kommt in dem Beispiel zudem eine besondere Bedeutung zu. Dies ist das Startsymbol einer Grammatik und stellt ein spezielles Nonterminal dar, mit dem die Konstruktion von Ausdrücken beginnt. [Hof18]

Für die Grammatik G kann damit abgeleitet werden, dass diese als Quadrupel $(V, \Sigma, \mathrm{P}, S)$ definiert ist. Dabei steht V für die Menge der Nonterminalsymbole, Σ für das endliche Terminalalphabet ohne Überlappung mit V, P für die Produktionsregeln, die festlegen, wie die Sprache aufgebaut wird und S für das Startsymbol. Mithilfe der Produktionsregeln kann man aus dem Startsymbol vollständige, syntaktisch korrekte Sätze erzeugen oder die Korrektheit einer vorgegebenen Zeichenfolge überprüfen. [Hed 2012]

2.3.2. Sprachhierarchie

Formale Grammatiken sind ein vielseitiges Instrument zur Erzeugung verschiedener Sprachformen. Diese reichen von einfachen Wortlisten bis hin zu komplexen Strukturen, die dem gesprochenen Wort ähneln. Die Art und Weise, wie die Produktionsregeln in einer Grammatik G strukturiert sind, beeinflusst dabei maßgeblich die Eigenschaften der erzeugten Sprache $L(G)$. Laut Noam Chomsky lassen sich Grammatiken in vier unterschiedliche Klassen einteilen, die in einer hierarchischen Beziehung zueinanderstehen. Bekannt sind diese als Chomsky-Hierarchie (vgl. Abbildung 2-6) [Cho 2002]. Diese Hierarchie umfasst die folgenden vier Grammatiktypen, die auf ihren jeweiligen Ersetzungsregeln basieren:

- Phrasenstrukturgrammatiken (Typ-0-Grammatiken). Diese bilden die allgemeinste Form der Grammatiken und haben keine Einschränkungen auf die Struktur der Produktionsregeln. Jede Grammatik zählt per Definition zu den Typ-0-Grammatiken, da hier keine speziellen Regeln über die Form der Produktionen festgelegt werden.
- Kontextsensitive Grammatiken (Typ-1-Grammatiken). Diese Grammatikart erlaubt es, ein Nicht-Terminal in einem bestimmten Kontext durch eine beliebige Kombination aus

Terminalen und Nicht-Terminalen zu ersetzen. Die Grammatik ist kontextsensitiv, wenn für jede Produktionsregel l →r entweder die Beziehung |r|≥|l| erfüllt ist oder diese die Form S→ε aufweist.

- Kontextfreie Grammatiken (Typ-2-Grammatiken). Hierbei besteht die linke Seite jeder Produktionsregel aus genau einer Variablen. Ein einzelnes Nicht-Terminal wird durch eine Folge aus Terminalen und Nicht-Terminalen ersetzt.
- Reguläre Grammatiken (Typ-3-Grammatiken). Diese sind eine Unterklasse der kontextfreien Grammatiken und besitzen die zusätzliche Einschränkung, dass die rechte Seite einer Produktion entweder aus dem leeren Wort ε oder einem Terminalsymbol, gefolgt von einem Nicht-Terminal, besteht. Die Produktionen folgen einem einfachen Schema, wodurch sich reguläre Sprachen leicht durch Automaten darstellen lassen. [Cho02]

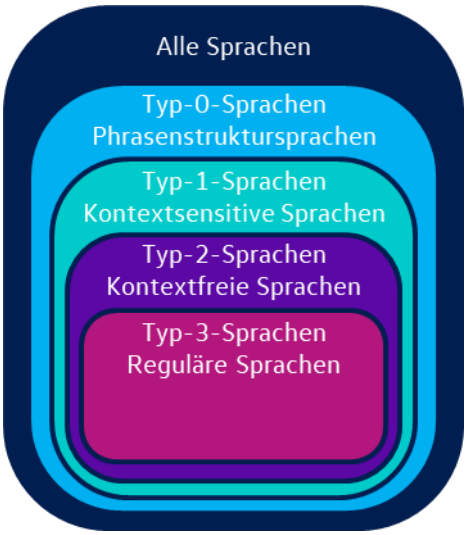

Abbildung 2.6: Sprachhierarchie nach Chomsky, nach [Cho02]

2.3.3. Kontextfreie Sprachen

Kontextfreie Grammatiken erweitern das Konzept der regulären Grammatiken, indem diese mehr Flexibilität bieten. In beiden Fällen müssen die Produktionen auf der linken Seite ein einzelnes Nicht-Terminal enthalten. Während reguläre Grammatiken jedoch strenge Einschränkungen für die rechte Seite einer Produktion mit sich bringen, erlauben kontextfreie Grammatiken dort beliebige Kombinationen aus Terminalen und Nicht-Terminalen. Formaler ausgedrückt ist eine Grammatik dann kontextfrei, wenn jede Regel die Form $l \rightarrow r$ mit $l \in V$ und r $\in$ ($\Sigma \cup$ V) hat.

Dank der Ausdrucksstärke eignen sich kontextfreie Sprachen besonders gut zur Beschreibung der Syntax von Programmiersprachen. So wurde bereits in den frühen 1960er-Jahren eine solche Grammatik vom Informatikpionier John Backus genutzt, um die Struktur der Programmiersprache Algol60 festzulegen. Die von ihm entwickelte Notation ist seitdem auch bekannt als Backus-Naur-Form (BNF) und ist inzwischen zum Standard für die Definition von Programmiersprachen geworden. Für noch mehr Flexibilität wurde später die Erweiterte Backus-Naur-Form (EBNF) eingeführt. Mit dieser lassen sich Wiederholungen und optionale Elemente einfacher ausdrücken, ohne dabei auf Rekursion oder alternative Produktionen zurückgreifen zu müssen. Die entsprechende Notation der EBNF, die für diese Arbeit genutzt wird, ist in Tabelle 2-1 angegeben. [Hed12, Pri15]

Tabelle 2-1: Regeln zur Definition einer Grammatik in EBNF-Notation, nach [Pri15]

Notation	Bedeutung
<Zeichenkette>	Nicht-Terminal
„valide(s) Zeichen“	Terminal
+	Und Verknüpfung
|	Oder Verknüpfung
(Ausdruck)?	Optionalität des Ausdrucks
(Ausdruck)+	Einmaliges oder beliebig häufiges Vorkommen
(Ausdruck)*	Optionales oder beliebig häufiges Vorkommen

2.3.4. Domänenspezifische Sprachen

Eine domänenspezifische Sprache (engl. Domain Specifc Language, DSL) ist eine Programmiersprache, die speziell auf einen bestimmten Anwendungsbereich zugeschnitten ist. Im Gegensatz zu Allzweck-Programmiersprachen (engl. General Purpose Languages, GPLs) wie Java, C oder Python, die für eine Vielzahl von Aufgaben verwendet werden können, konzentrieren sich DSLs auf die speziellen Anforderungen einer bestimmten Domäne. [Kra10] Ihre Befehle und Operatoren sind direkt auf die Problembereiche ausgerichtet, wodurch eine effizientere zielgerichtete Problemlösung ermöglicht wird.

GPLs zeichnen sich zudem durch ihre Turing-Vollständigkeit aus, was bedeutet, dass sie in der Lage sind, alle berechenbaren Probleme zu lösen, die auch mit einer Turing-Maschine darstellbar sind. DSLs hingegen sind in der Regel nicht Turing-vollständig, da sie auf spezifische Problemstellungen zugeschnitten sind. In der Praxis haben sich beispielsweise folgende DSLs etabliert:

- SQL: Abfragesprache für Datenbanken
- LaTeX: Textsatzsystem; vorrangig zur Erstellung von wissenschaftlichen Arbeiten
- Matlab: Skriptsprache für technische Berechnungen (an der Grenze zur GPL)
- HTML: Sprache zur Strukturierung von Webdokumenten
- make: Programmierwerkzeug zur Automatisierung von Build-Prozessen [Pri15]

Ähnlich wie eine GPL, umfasst auch die DSL mehrere Schlüsselelemente [HR04]:

- **Konkrete Syntax**: Bestimmt, welche Sprachkonstrukte und Ausdrücke zulässig sind.
- **Abstrakte Syntax**: Beschreibt die grundlegende Struktur und Logik der Sprache. Diese stellt die zentrale Datenstruktur dar, auf der spätere Verarbeitungsschritte wie Codegenerierung und Analyse aufbauen. Im Kontext von DSLs wird die abstrakte Syntax oft auch als Domänenmodell bezeichnet. [DKL84, Wil97]
- **Kontextbedingungen**: Setzen Einschränkungen, die sicherstellen, dass nur valide Sprachkonstrukte gebildet werden können. Diese ergänzen damit die Syntax, indem sie zusätzliche Regeln definieren, die nicht direkt durch die formalen Strukturen ausgedrückt werden können.
- **Semantik**: Gibt die Bedeutung der Sprachkonstrukte vor. Diese kann durch unterschiedliche Ansätze wie denotationelle oder operationale Semantik definiert werden. Beide basieren auf der abstrakten Syntax. [Plo81, SS71]

Die Definition von DSLs kann durch verschiedene Methoden erfolgen. Grundsätzlich wird dabei zwischen grammatikbasierten und modellbasierten Ansätzen unterschieden. Grammatikbasierte Ansätze nutzen verschiedene Formen von Grammatiken zur Definition der konkreten und abstrakten Syntax. Kontextfreie Grammatiken führen oft zu einer Baumstruktur der Sprachinstanzen. Es gibt

spezielle Techniken wie Attributgrammatiken, um aus diesen Strukturen weitere Zusammenhänge zu errechnen. Alternativ ermöglichen Graphgrammatiken [Nag79] die Spezifikation visueller Sprachen mit Graphstrukturen.

Modellbasierte Ansätze hingegen setzen zur Definition der abstrakten Syntax auf datenmodellähnliche Strukturen, wie diese beispielsweise (auch in Klassendiagrammen verwendet werden. Diese Ansätze sind besonders nützlich für die Gestaltung grafischer Modellierungssprachen wie etwa der Unified Modeling Language (UML) [Kra10].

3. Problemanalyse

Eingangs wurde beschrieben, das automatisierte Systeme ein sicherheitskritisches System darstellen, an dessen Entwicklung besondere Anforderungen gestellt werden. Eine besondere Bedeutung kommt dabei der ODD zu, mit dessen Hilfe die zulässigen und unzulässigen Betriebsbedingungen beschrieben werden. Um die ungelösten Probleme im Zusammenhang mit der Beschreibung und Nutzung der ODD im Kontext der Entwicklung zu analysieren, geht dieses Kapitel auf verschiedene Phasen relevanter Entwicklungsprozesse ein. Im Detail wird die Entwicklung eines ADS unter Berücksichtigung eines konventionellen Entwicklungsmodells beleuchtet und diese Betrachtung um die regulatorischen Rahmenbedingungen, dem Business Case Management und dem Betrieb erweitert. Der Fokus liegt darauf, einen umfassenden Überblick darüber zu geben, wie die ODD in verschiedenen Prozessen eingesetzt wird und welche potenziellen Herausforderungen dabei auftreten können. Zum Abschluss dieses Kapitels wird eine zusammenfassende Problemanalyse gegeben, aus der spezifische Anforderungen an eine Lösung abgeleitet werden (vgl. Abbildung 3.1).

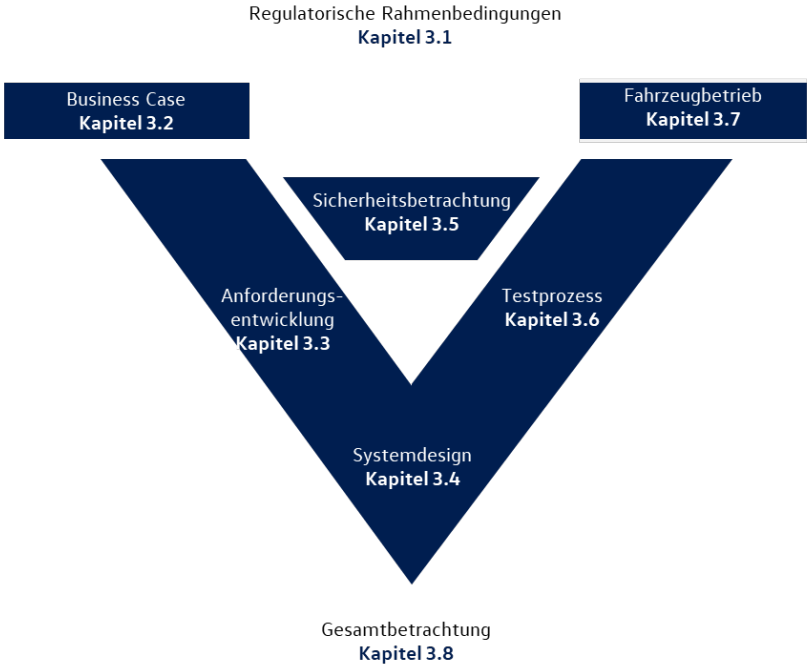

Abbildung 3.1: Darstellung der Kapitelstruktur anhand ODD-relevanter Prozesse

3.1. Regulatorische Rahmenbedingungen

Dieses Kapitel untersucht die aktuelle und zukünftige Rechtslage bezüglich der Zulassung und des Betriebs von ADS. Es wird der rechtliche Rahmen für die Zulassung in Europa erläutert und auf nationale Besonderheiten in Deutschland eingegangen. Zusätzlich dazu wird der rechtliche Rahmen in den USA und in China grob beleuchtet. Abschließend werden veröffentlichte Beispiele für die bisherige Beschreibung einer ODD gegeben, die im Sinne des aktuellen Rechtsrahmen sind und zur Kommunikation der Systemfähigkeiten mit der Öffentlichkeit genutzt werden.

3.1.1. Genehmigungsprozess Europa mit Fokus Deutschland

Für die Europäische Union spielt im Typgenehmigungsverfahren die Nachvollziehbarkeit eine zentrale Rolle. Es ist von großer Bedeutung, dass ADS als reife und verlässliche Technologien wahrgenommen werden, die unter festgelegten Bedingungen sicher agieren können. Eine klare Kommunikation über die Einsatzbedingungen und Grenzen dieser Systeme ist essenziell, um deren Funktionsweise zu

verstehen. Für das Gewinnen des Vertrauens der Behörden, der Endnutzer und der Öffentlichkeit ist es daher unerlässlich, ein transparentes Genehmigungsverfahren zu gewährleisten. Diese strategischen Überlegungen sind wegweisend für die Ausgestaltung der Typgenehmigungsverfahren in der EU und prägen maßgeblich den Diskurs über die regulatorischen Rahmenbedingungen für ADS. Das Beispiel von Cruise, einer Tochtergesellschaft von General Motors, verdeutlicht dabei nachdrücklich die Bedeutung dieses Verfahrens. Der Rückruf aller Robotaxis nach einem Unfall, die anschließende Sicherheitsprüfung und die temporäre Suspendierung der Betriebslizenzen durch die kalifornischen Behörden, heben die kritische Notwendigkeit einer transparenten und offenen Kommunikation sowie eines umfassenden Sicherheitsmanagements hervor. [Mat24]

Die EU-Verordnung zur Typgenehmigung vollautomatisierter Fahrzeuge zielt auf die Regulierung von ADS und deren spezifische Anwendungsfälle ab. Wesentlich dabei ist, dass das ADS in der Lage sein muss, die gesamte DDT innerhalb seiner ODD eigenständig zu bewältigen. Entscheidend ist dabei die Fähigkeit des ADS, relevante Objekte und Ereignisse, andere Verkehrsteilnehmer, Hindernisse, Unfälle, Verkehrsstaus und Umweltbedingungen zu erkennen und darauf angemessen zu reagieren. Zudem werden weitere Anforderungen an das System gestellt. Dieses muss kritische Verkehrsszenarien bewältigen können, sich seiner Systemfähigkeiten jederzeit bewusst sein, sowie dessen Grenzen verstehen, um bei deren Überschreitung angemessen zu reagieren. Die Hersteller sind verpflichtet, die ODD zu definieren, innerhalb der das ADS sicher operieren kann. Diese Anforderungen umfassen verschiedenste Aspekte wie Wetterbedingungen, Tageszeit, Lichtintensität und Straßenbeschaffenheit. [Eur22]

Für die weitere Erhöhung der Sicherheit verlangt die EU-Verordnung von den Herstellern, dass diese die funktionale und operative Sicherheit ihrer ADS während des gesamten Entwicklungsprozesses berücksichtigen. Dies bedeutet, dass die Hersteller nachweisen müssen, dass ihre Systeme so entworfen und entwickelt wurden, dass sie innerhalb der festgelegten ODD ohne unverhältnismäßige Risiken für die Fahrzeuginsassen und andere Verkehrsteilnehmer funktionieren. Dies beinhaltet die Einrichtung klar definierter Abnahmekriterien, die auf einer gründlichen Analyse vorhandener Unfalldaten, der Leistung manuell gefahrener Fahrzeuge und des aktuellen Stands der Technik basieren. [Eur22]

Weiter wird in der EU-Verordnung konkretisiert, das spezifische Testverfahren und Bewertungskriterien, die auf die Überprüfung der Funktionsfähigkeit und Sicherheit des ADS unter realen Bedingungen abzielen, integriert werden müssen. Dies gewährleistet, dass die Systeme nicht nur theoretisch innerhalb ihrer ODD sicher operieren können, sondern auch praktisch bewiesen haben, dass sie in der Lage sind, auf eine Vielzahl von Verkehrsszenarien entsprechend der Spezifikation zu reagieren. Diese praktischen Tests und Bewertungen bilden eine wesentliche Grundlage für die Typgenehmigung und tragen maßgeblich dazu bei, eine solide Basis für die Sicherheit und Zuverlässigkeit vollautomatisierter Fahrzeuge zu schaffen. [Eur22]

Ergänzend gilt für den deutschen Rechtsrahmen, dass die Typgenehmigung für Level 4 autonome Fahrsysteme von der nationalen Behörde, in diesem Fall dem Kraftfahrt-Bundesamt, erteilt wird. Diese Behörde ist für die Überprüfung und Genehmigung von Fahrzeugen verantwortlich und führt diese Aufgabe unter Maßgabe der europäischen Vorschriften aus. Zusätzlich zu den europäischen Vorschriften gibt es in Deutschland eine Reihe von Vorschriften zur Absicherung des Betriebsbereichs. Insbesondere beinhaltet diese die Genehmigung festgelegter Betriebsbereiche gemäß den Bestimmungen des Kraftfahrt-Bundesamts, sofern das Amt feststellt, dass das Fahrzeug gemäß den Angaben in der Betriebserlaubnis die Fahraufgabe in diesem bestimmten Betriebsbereich automatisiert und sicher bewältigen kann. Des Weiteren werden Testszenarien und Bestehenskriterien festgelegt, um sicherzustellen, dass der Betrieb des ADS im definierten Betriebsbereich keine

Beeinträchtigung der Verkehrssicherheit darstellt und keine übermäßige Gefährdung von Personen hervorruft. Dies bedeutet, dass sowohl die technischen Anforderungen des ADS als auch die Infrastruktur entlang der Streckenführung den Vorgaben entsprechen müssen. [Bun22]

3.1.2. Internationaler Rechtsrahmen

Zwischen Europa und den USA bestehen hinsichtlich der Zulassungsverfahren für ADS große Unterschiede, vor allem aufgrund der jeweiligen regulatorischen Rahmenbedingungen und Ansätze zur Förderung von Sicherheit und Innovation. So ist in den USA die Regulierung von autonomen Fahrzeugen durch eine Mischung aus bundesstaatlichen Richtlinien und einzelstaatlichen Vorschriften geprägt. Die National Highway Traffic Safety Administration (NHTSA) bietet lediglich freiwillige Leitlinien für Hersteller, während die einzelnen Bundesstaaten eigene Gesetze und Vorschriften erlassen, die den Test und Betrieb autonomer Fahrzeuge regeln. Ein repräsentatives Beispiel hierfür ist der Bundesstaat Kalifornien, der detaillierte Betriebsstandards und Testanforderungen für autonome Fahrzeuge festlegt hat. Die dortige Behörde, das California Department of Motor Vehicles (DMV), bietet unter anderem Workshop-Videos an, die die Öffentlichkeit über die Mindestbetriebsstandards und Testanforderungen für autonome Fahrzeuge in Kalifornien informieren sollen. Diese Videos ermöglichen es Herstellern, Vertriebspartnern und Betreibern von autonomen Fahrzeugen, die Anforderungen für den Betrieb ihrer Fahrzeuge auf kalifornischen Straßen zu überprüfen und einzuhalten. Darüber hinaus haben die California Public Utilities Commission (CPUC) und die Stadt San Francisco Genehmigungen für Unternehmen wie Waymo und Cruise erteilt, die den zahlungspflichtigen Betrieb von fahrerlosen Fahrzeugen mit Passagieren erlauben. [Cal23a, Cal23b, Nat20, SHC23]

China hingegen setzt bei der Entwicklung und Regulierung von ADS fast komplett auf einen dezentralen Ansatz, der durch lokale Standards und Regularien geprägt ist. Besonders sticht hierbei die Shenzhen Special Administrative Region heraus, da bereits spezifische Regelungen existieren, die autonome Fahrzeuge von den normalen Zulassungsbedingungen befreien. Dies ermöglicht es theoretisch sogar, ADS zu verkaufen und zu registrieren, selbst wenn diese die üblichen Zugangsbedingungen für Straßenfahrzeuge nicht erfüllen. Dies gilt so lange, wie diese neue Technologien, Techniken oder Materialien nutzen. Zusätzlich gibt es hinsichtlich der Haftung bei Verkehrsunfällen und -verstößen klare Regelungen. [GW20, SZ21]

3.1.3. Veröffentlichte Beispiele der ODD

Um das Verständnis der bisherigen Erläuterungen hinsichtlich der ODD und regulatorischen Rahmenbedingungen zu erhöhen, werden zwei Beispiele erläutert, die aktuell veröffentlichte ODDs im Rahmen der Zulassung beschreiben. Das Erste beschreibt ein Level 3-System von Mercedes, den Drive Pilot, welcher das erste zugelassene Serienfahrzeug nach Level 3 weltweit ist. Trotz dessen liegt für dieses System keine ODD im Detail vor und es lassen sich lediglich abstrakte Informationen zu den Einsatzbedingungen ableiten. Diese Informationen zeigen, dass der Drive Pilot für den Einsatz auf bestimmten Autobahnabschnitten in Deutschland und bei moderatem bis hohem Verkehrsaufkommen bis zu einer Geschwindigkeit von 60 km/h ausgelegt ist. [Mac22] Zusätzlich wird erwähnt, dass das System nur bei guten Wetterbedingungen funktioniert, was darauf hindeutet, dass die ODD bestimmte Wetterbedingungen und Verkehrsdichten umfasst. [Car23] Weitere öffentlich dokumentierte Beschreibungen dieser Bedingungen und Einschränkungen fehlen jedoch.

Das zweite Beispiel beschreibt ein Level 4 ADS von Nuro, welches in Kalifornien, für den Test- und Erprobungsbetrieb zugelassen ist. Laut der veröffentlichten Informationen zur ODD ist das System für den Rund-um-die-Uhr-Einsatz in San Mateo County und Santa Clara County berechtigt und funktioniert

unter verschiedenen Wetterbedingungen wie trockenem oder nassem Asphalt, leichtem Regen und leichtem bis mäßigem Nebel. Die Geschwindigkeit des Systems liegt maximal bei 35 mph. [Cal22b]

Beide Beispiele zielen darauf ab, die Funktionalität leicht verständlich zu kommunizieren und sind nicht dafür gedacht, als technisch präzise Beschreibung innerhalb des Entwicklungsprozesses genutzt zu werden.

3.2. Betrachtung des Business Case für ADS

Mit der Einführung des serienmäßigen Betriebs von ADS sind hohe Kosten verbunden. Die Zielgruppe sind daher nicht Privatpersonen, sondern Flottenbetreiber, die mit Hilfe eines ADS Mitarbeiter einsparen können, um Personen (Mobility as a Service (MaaS)) oder Güter (Transport as a Service) im Sinne einer Serviceleistung zu transportieren. In diesem Kontext spielt das Business Case Management eine entscheidende Rolle, indem es sich auf die sorgfältige Planung und Ausgestaltung dieser Services konzentriert. Dazu gehört auch die Definition von wirtschaftlich sinnvollen Betriebsbereichen, für die ein Service angeboten werden soll und entsprechende Fahrzeugfähigkeiten entwickelt werden sollen. Um dies zu realisieren, wird auf Basis einer spezifischen ODD eine Abschätzung der potenziell möglichen Betriebszeiten sowie ein technisch realisierbares GeoNet benötigt.

Für die Planung der Servicezeiten ist eine sorgfältige Abstimmung der ODD mit den spezifischen Umweltbedingungen des Zielgebiets notwendig. Vor allem temporäre Umweltfaktoren, wie beispielsweise die Wetterverhältnisse, spielen eine zentrale Rolle. Ein präziser Abgleich dieser Variablen ermöglicht es, die Servicezeiten differenziert nach Betriebsphasen (Uptime) und Nicht-Betriebsphasen (Downtime) zu bestimmen. Durch die Analyse von historischen Wetterdaten in Verbindung mit der ODD eines ADS, lassen sich verschiedene Städte hinsichtlich ihrer voraussichtlichen Servicezeiten vergleichen. Diese Servicezeiten beeinflussen direkt die Wirtschaftlichkeit des Systems, da längere Betriebszeiten die Wirtschaftlichkeit erhöhen. Ein ADS, das aufgrund von Umweltbedingungen außerhalb seiner ODD, nicht operieren kann, generiert keine Einnahmen. Das Beispiel in Abbildung 3.2 illustriert dies anhand der Städte London und Phoenix und zeigt auf, das auf Basis der technisch anzubietenden Servicezeit die Stadt Phoenix im Vergleich zu London zu bevorzugen wäre.

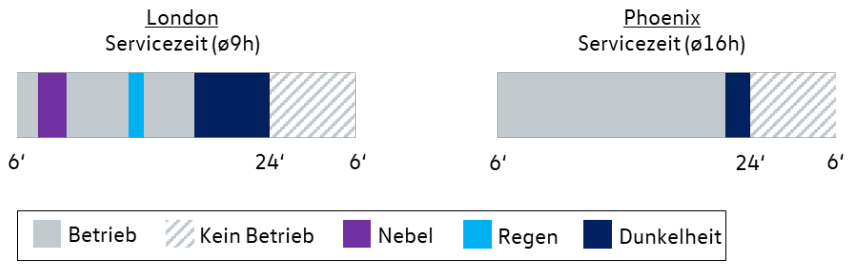

Abbildung 3.2: Abschätzung der prognostizierten Servicezeiten anhand der ODD

Die Bestimmung des technisch realisierbaren GeoNets, hängt von den befahrbaren Straßen ab, auf denen das ADS auf Grundlage seiner ODD sicher operieren kann. Ähnlich zu der Bestimmung der Servicezeiten, kann auf Basis des intendierten Betriebsbereich (vgl. Abbildung 3.3, erster Input) und der Systemfähigkeiten bzw. ODD (vgl. Abbildung 3.3, zweiter Input), ein Abgleich vorgenommen werden und ein befahrbares GeoNet bestimmt werden (vgl. Abbildung 3.3, Output). Im Vergleich zur Bestimmung der Servicezeiten ändern sich hierbei lediglich die Inputdaten.

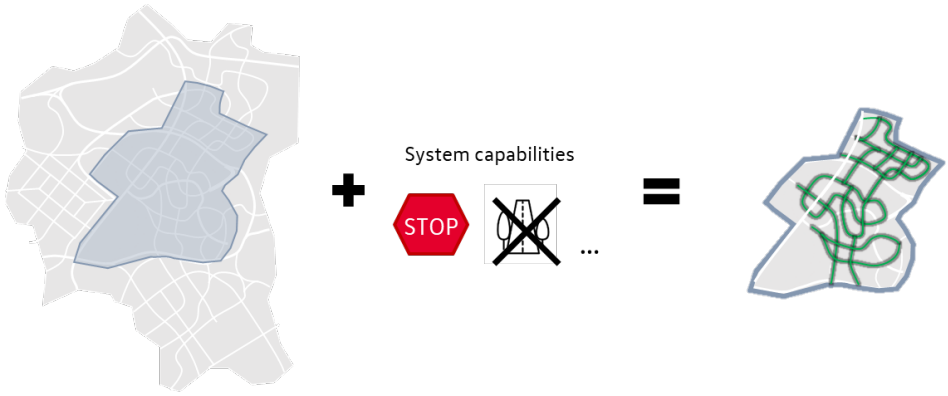

Abbildung 3.3: Abschätzung des verfügbaren GeoNets anhand der ODD

Neben der Bestimmung der Servicezeiten und des befahrbaren Straßennetzes ist die stetige Erweiterung der Fahrzeugfunktionalität und des Servicenetzes ein weiterer entscheidender Aspekt. Die kontinuierliche Verbesserung und die Anpassung an veränderte Nutzeranforderungen und technologische Fortschritte sichern die langfristige Attraktivität und Wettbewerbsfähigkeit des Mobilitätsangebots. Aus Sicht des Business Case Managements ist es daher wichtig, bestimmen zu können, welche noch nicht beherrschte Funktionalität, den potenziell größten Mehrwert für den Service bringt. Es ist daher von Interesse, auf Basis einer ODD sowie den bereits beherrschten Straßennetzwerken und Servicezeiten, die Funktionserweiterungen mit dem potenziell größten Mehrwert zu bestimmen. Im Detail bedeutet dies, zu überprüfen, welcher zusätzliche Funktionsumfang den größten GeoNet- oder Servicezeitenzuwachs ermöglicht und damit als nächstes entwickelt werden sollte.

3.3. Anforderungsentwicklung

Für die Fahrzeugentwicklung ist eine genaue Analyse und Definition von Anforderungen entscheidend, um gewährleisten zu können, dass ein zuverlässiges und sicheres System für die beabsichtigten Betriebsbedingungen entwickelt wurde. Je nach Automatisierungsgrad, variieren die Anforderungen des Systems jedoch stark. Während Systeme nach SAE-Level 2 stets auf den Fahrer als überwachendes und kontrollierendes Organ zurückgreifen können, müssen Level 4-Systeme vollständig automatisiert innerhalb ihrer spezifizierten Betriebsbedingungen agieren. Dies führt zu unterschiedlichen Herausforderungen in der Anforderungsanalyse und –definition, insbesondere in Bezug auf die ODD. Nachfolgend wird auf diese detaillierter eingegangen.

3.3.1. Prozess zur Anforderungsentwicklung auf Basis einer ODD

Der Prozess zur Definition der ODD für ADS gliedert sich in drei wesentliche Schritte (siehe Abbildung 3.4). Diese strukturierte Herangehensweise bildet die Grundlage für die Definition einer ODD, indem sie sicherstellt, dass das System den spezifischen Anforderungen der Einsatzumgebung entspricht. [Aut20]

Im ersten Schritt erfolgt die Identifikation des relevanten Straßennetzes, auf dem das ADS operieren soll. Diese Phase ist entscheidend, um den geografischen und infrastrukturellen Rahmen festzulegen, innerhalb dessen das ADS sicher und effizient navigieren kann. Der zweite Schritt umfasst die Identifikation der relevanten Merkmale des Straßennetzes. Hierbei werden spezifische Eigenschaften wie Verkehrsfluss, Kreuzungstypen, Beschilderung, Straßenmarkierungen und weitere Umgebungsvariablen analysiert. Diese detaillierte Betrachtung ermöglicht es, die Herausforderungen und Anforderungen zu verstehen, die an das ADS gestellt werden, um eine sichere Interaktion mit der

Umgebung zu gewährleisten. Im dritten und letzten Schritt wird die ODD bestimmt und daraus die spezifischen Anforderungen für das ADS abgeleitet. Nach Abschluss dieser Schritte wird das ADS entsprechend entwickelt, getestet und betrieben. Sollten sich Änderungen an der Route oder an den Merkmalen der Route ergeben, ist es notwendig, den zweiten Schritt erneut zu durchlaufen, um sicherzustellen, dass das ADS weiterhin den Anforderungen entspricht. Dieser adaptive Prozess ermöglicht eine kontinuierliche Anpassung und Optimierung des Systems, um eine maximale Sicherheit und Effizienz im Betrieb zu gewährleisten. In der nachfolgenden Betrachtung wird detailliert auf diesen Prozess und seine Bedeutung für die Entwicklung und den Einsatz von ADS eingegangen. [Aut20]

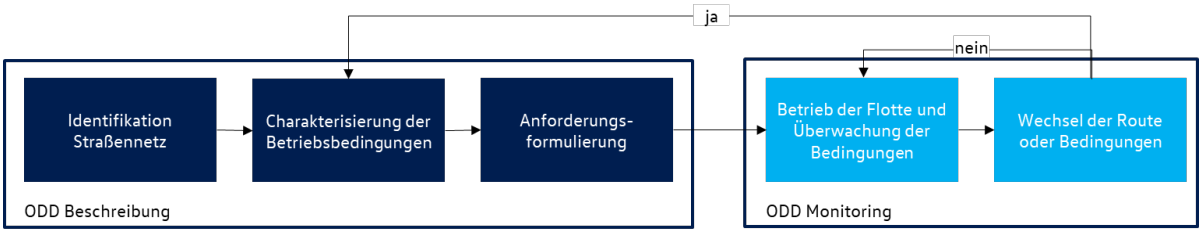

Abbildung 3.4: Prozess zur Anforderungsentwicklung einer ODD, nach [Aut20]

Schritt 1 – Identifizierung des Straßen-/Routennetzes: Die Identifizierung des für den Betrieb eines ADS relevanten Straßennetzes beginnt mit einer präzisen Beschreibung des Services, der angeboten werden soll (vgl. Abbildung 3.5). Dieser Service kann entweder eine Mobilitätsdienstleistung oder eine Transportdienstleistung umfassen. Abhängig von der Art des Services, ergeben sich spezifische Anforderungen an das ADS, die bereits in dieser frühen Phase berücksichtigt werden müssen. Darauf folgt die Festlegung des geografischen Einsatzgebietes, in dem der Service angeboten werden soll. Hier fließen Informationen aus dem Business Case Management (BCM) ein, wobei die Entscheidung für ein bestimmtes Gebiet auf wirtschaftlichen Kriterien oder den Präferenzen der Kunden basiert (vgl. Kapitel 3.2). Dies ist entscheidend, da diese die Basis für die Auswahl des Straßennetzwerks legt, in dem das ADS operieren soll. Dabei geht es nicht nur um die Identifikation geeigneter Routen, sondern auch um die Anpassung an den vorgesehenen UseCase des Fahrzeugs innerhalb eines klar abgegrenzten geografischen Bereichs. Die geografische Begrenzung des Bereichs erleichtert es Entwicklern, im nächsten Schritt relevante Anforderungen zu identifizieren. Zudem werden klare Grenzen geschaffen, welche Umgebungsbedingungen (z. B. Schnee, Regen, Wind) erwartet werden können. Dabei ist zu beachten, dass mit der Zeit und der Weiterentwicklung des ADS sich Änderungen an den Bedingungen und Einschränkungen ergeben können. Diese Veränderlichkeit macht eine Rückkopplungsschleife notwendig, die Anpassungen an der ursprünglichen Definition des Services, des Servicegebiets oder des Straßennetzwerks ermöglicht. [Aut20]

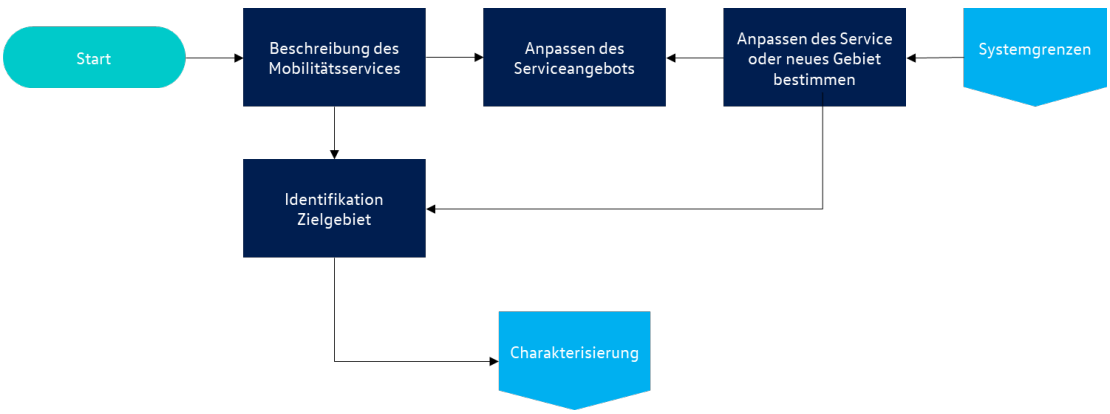

Abbildung 3.5: Beschreibung des Serviceangebots, nach [Aut20]

Schritt 2 – Charakterisierung des festgelegten Routennetzwerks und der Infrastruktur: Im nächsten Schritt wird das festgelegte Routennetzwerkes charakterisiert, um anhand dessen die ODD bestimmen zu können. Dafür werden die Merkmale des Straßennetzwerkes dokumentiert. Im Einzelnen umfassen diese die für den Betrieb eines ADS relevanten Elemente der OD und ermöglichen es den Entwicklern, sich auf spezifische lokale Straßen-/Routenmerkmale und Bedingungen (z. B. lokales Wetter und infrastrukturelle Spezifika) zu konzentrieren. Die Merkmale können zudem spezialisierte Zonen umfassen, erforderliche infrastrukturelle Elemente und/oder Bereiche, die operationelle Herausforderungen für das ADS darstellen (z. B. schwer einsehbare Kurven, Kreisverkehre oder Tunnels). Diese Merkmale haben dabei verschiedene Eigenschaften und können klar abgegrenzt und fest (z. B. Schulzonen), klar abgegrenzt und dynamisch (z. B. Baustellen) oder Bereiche ohne physische Markierungen sein. Zu letzterem zählen beispielsweise überschwemmungsgefährdete Gebiete, die somit über die Zeit durch ihre physischen Merkmale bekannt werden. Die Auswahl und Dokumentation dieser Merkmale erfolgen in der Regel auf Basis einer vorgegebenen Taxonomie (vgl. Kapitel 2.1.2). Diese strukturierte Herangehensweise definiert präzise den Problemraum und erleichtert die Auswahl relevanter Merkmale für die ODD-Definition. Durch diesen Ansatz wird die Komplexität bei der Entwicklung und Implementierung von ADS reduziert und eine zielgerichtete Fokussierung auf die kritischen Aspekte der intendierten Umgebung ermöglicht. [Aut20]

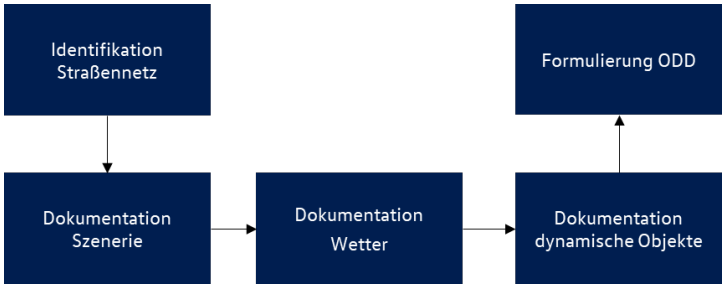

Abbildung 3.6: Formulierung der ODD I, nach [Aut20]

Schritt 3 – Formulierung der ODD-Anforderungen: Im letzten Schritt der ODD-Entwicklung erfolgt ein Abgleich der Fähigkeiten des Fahrzeugs mit den Merkmalen der OD, in der es fahren soll. Dieser Vorgang hilft dabei, die zulässigen und unzulässigen Betriebsbedingungen zu bestimmen. Gesamtheitlich führt dies zur Festlegung der Bedingungen, unter denen das Fahrzeug sicher betrieben werden kann, und aktualisiert entsprechend das Straßennetz, auf dem es fahren darf. Als Ergebnis entsteht die ODD. [Aut20] Bei der Erstellung der ODD gilt es, weitere Anforderungen zu berücksichtigen, auf die im nachfolgenden Kapitel eingegangen wird.

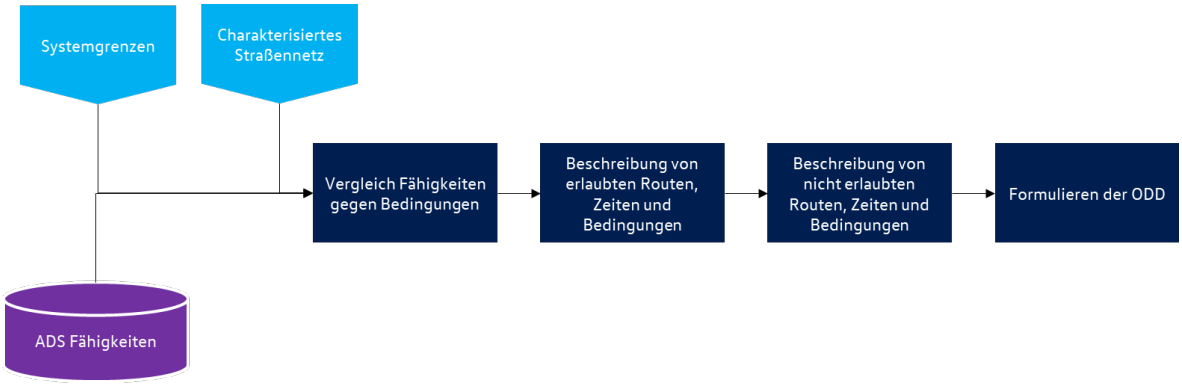

Abbildung 3.7: Formulieren der ODD II, nach [Aut20]

3.3.2. Anforderungen an die ODD-Dokumentation

Der vorherige Prozess hat beschrieben, wie relevante ODD-Bedingungen identifiziert werden können, um daraus nachfolgend Anforderungen an die Systementwicklung stellen zu können. Dabei gelten sowohl für die ODD als auch für daraus abgeleitete Anforderungen allgemeine Kriterien, die beachtet werden müssen, damit die Anforderungen nutzbar für die Entwicklung sind und ein gewünschtes und sicheres Produkt entwickelt werden kann. In Bezug auf die Dokumentation und Definition der ODD haben einige dieser Kriterien eine besondere Relevanz. Nachfolgend wird detailliert darauf eingegangen.

Ambiguität – Je nach gewählter Notation können sich Herausforderungen bezüglich der Mehrdeutigkeit von Anforderungen und der damit verbundenen potenziellen Fehlinterpretation von ergeben. Diese ist vor allem auf die natürliche Sprache zurückzuführen, jedoch nicht vollständig darauf beschränkt. Generell sollen Anforderungen Erwartungen an ein Produkt zum Ausdruck bringen. Da Erwartungen jedoch sehr subjektiv sind, macht es deren genaue Beschreibung schwierig. Für die Dokumentation der ODD muss deswegen eine Notation definiert werden, die Mehrdeutigkeiten vermeidet. [NHS+14]

Vollständigkeit – Neben der Ambiguität bzw. Mehrdeutigkeit ist im Kontext der Anforderungserhebung die Vollständigkeit eine weitere Herausforderung. Diese beschreibt die Eigenschaft, im ausreichenden Maße sowohl alle geäußerten als auch nicht geäußerten Erwartungen der Stakeholder zu erfüllen. Dabei ist Vollständigkeit komplex, da in einigen Umgebungen schwer feststellbar ist, ob alle relevanten Aspekte berücksichtigt wurden. In besonderem Maße trifft dies auf die ODD zu, da schwer festzustellen ist, was relevant ist und ob dies in der OD vorhanden ist. [HD04]

Rückverfolgbarkeit – Die Rückverfolgbarkeit kann bei großen Ingenieursprojekten mit einer Vielzahl von Anforderungen ebenfalls schwierig werden. Für die Entwicklung von Automobilen könnte dies jedoch zutreffen, da die Entwicklung von E/E-Komponenten bereits oftmals hunderte Seiten von Spezifikationen und zugehörigen Dokumenten beinhaltet. [WW03] Dabei ist es mitunter schwierig, sicherzustellen, dass jede Anforderung, mit Ausnahme von Top-Level-Anforderungen, auf mindestens eine übergeordnete Anforderung rückverfolgbar ist. Eins zu eins gilt dies auch für die Spezifikation der ODD.

Modularität – Eine weitere Herausforderung stellt die Modularität dar. Gerade in Bezug auf natürlich sprachlich formulierte Anforderungen ist es oftmals schwierig, alle verknüpften Anforderungen zu identifizieren. Bei Änderungen von Anforderungen sind die potenziellen Auswirkungen dann nicht immer ersichtlich und jede Anforderung muss einzeln betrachtet werden. Eine potenzielle Lösung bildet die Gruppierung oder Modularisierung, diese ist jedoch nicht immer leicht umzusetzen. [Som11]

Detaillierungsgrad – Ebenfalls relevant ist die Frage nach der Granularität bzw. dem Detaillierungsgrad. Gyllenhammer et al. liefern dazu ein anschauliches Beispiel, welches die Komplexität der Fragestellung unterstreicht [GJW+20]. Er beschreibt ein ADS, welches die Aufgabe hat, autonom zwischen zwei schwedischen Städten mit einer Höchstgeschwindigkeit von 110 km/h zu fahren. Dabei darf dieses Fahrzeug nur auf einer Autobahn betrieben werden und lediglich tagsüber zwischen April und Oktober bei trockenem Wetter (< 1mm Regen pro Stunde) aktiv sein. Auf den ersten Blick sind alle relevanten Fakten vorhanden. Betrachtet man jedoch die Einzelaussagen im Detail, wird schnell ersichtlich, dass praktisch keine relevanten Informationen ausdrücklich angegeben sind. Es ist aber entscheidend, Anforderungen an das ADS zu erhalten, die überprüfbar sind. Eine ausdrücklich angegebene Dimension ist das Geschwindigkeitslimit. Dies ist einfach zu entwerfen und leicht mit einer technischen Anforderung zu verbinden. Die Beschränkung auf Frühling-Herbst gibt zudem einige zusätzliche

Informationen, die weiter ausdetailliert werden können. So kann der Angabe der Jahreszeit entnommen werden, dass höchstwahrscheinlich nicht mit Schnee umgegangen werden muss und nur in wirklich seltenen Fällen Temperaturen unter null Grad Celsius auftreten. Auf Basis der ursprünglichen Aussage kann jedoch nicht gesagt werden, ob dies innerhalb der Fähigkeiten des ADS sein muss oder nicht. Für diese Einschränkungen gibt es wahrscheinlich gute Gründe, die den Kundennutzen oder die Einfachheit der Implementierung und des Designs berücksichtigen. Diese Gründe sind jedoch in einer "Use Case-Aussage" verpackt und verschleiern die Auswirkungen auf das Design. Deutlicher wird dies am Beispiel von Regen. Der Regen selbst ist nicht das Problem für die Entwicklung des ADS, sondern der Verlust der Sicht und die Reduzierung der Reibung auf der Straße. Ähnlich verhält sich dies mit der Tageszeitbeschränkung. Wahrscheinlich wird diese genannt, um sicherzustellen, dass genügend Beleuchtung vorhanden ist. Auch hier nennt die Aussage jedoch nicht alle Fakten, die theoretisch berücksichtigt werden müssten. Beispielsweise gibt es immer noch das seltene Ereignis der Sonnenfinsternis. Für dieses stellt sich die Frage, ob es bewältigt werden muss oder, auch wenn es tagsüber auftritt, außerhalb der Spezifikation liegt. Aus der ursprünglichen Beschreibung ergeben sich weitere Unklarheiten. Was bedeutet es für die Anforderungen, auf eine Autobahn zwischen zwei Städten beschränkt zu sein? Gibt es Barrieren? Was sind die Geschwindigkeitsbegrenzungen? Welche Krümmungen und Fahrbahnmarkierungen können erwartet werden? Welche Details sollten notiert werden, weil sie relevant sind? Noch größer werden die Unsicherheiten nach der richtigen Detaillierung in Bezug auf das Verkehrsverhalten. Die ursprüngliche Aussage beschreibt, dass das Fahren dem schwedischen Gesetz entsprechen muss, jedoch gibt es viele Nuancen, die im Verkehrsverhalten auftreten können, jedoch auch nicht immer müssen. Um dies beschreibbar zu machen, muss für die Modellierung auf probabilistische Methoden zurückgegriffen werden. Zum Beispiel ist es unwahrscheinlich, dass Fußgänger auf der Autobahn angetroffen werden, es ist jedoch auch nicht gänzlich auszuschließen. Eine Panne am Straßenrand kann schnell dazu führen, dass Fußgängern auf dem Seitenstreifen stehen und einen vorherigen Ausschluss ad absurdum führen. Diese Nuancen müssen im richtigen Maße berücksichtigt werden, da sonst ein ADS entwickelt wird, das entweder unnötig große oder unzureichende Sicherheitsvorkehrungen implementiert hat. Gyllenhammer et al. zeigen mit diesem Beispiel sehr präzise, wie herausfordernd bei der Entwicklung eines ADS die Beschreibung der ODD werden kann. Es wird aufgezeigt, wie unzureichend detaillierte Anforderungen zu Mehrdeutigkeiten führen können oder große Lücken bei der Frage nach den zu entwickelnden Systemfähigkeiten hinterlassen. Diese Herausforderung ist nicht neu, wird durch die betrachtete Domäne der ODD aber maßgeblich verstärkt. [GJW+20]

Dynamische Umgebungen – Eine weitere Herausforderung betrifft das Handhaben von dynamischen Umgebungen. Das bedeutet, wie geht die Anforderungsanalyse mit sich ändernden Szenarien um, die z. B. Baustellen, temporäre Straßensperrungen oder plötzliche Wetteränderungen betreffen. Zwar müssen diese auch schon bei diversen Assistenzsystemen, wie dem Travel Assist, berücksichtigt werden, doch machen auch hier der Umfang der zu betrachtenden Elemente sowie das Fehlen einer menschlichen Rückfallebene den Unterschied aus. Die Unvorhersehbarkeit und Vielfalt möglicher Szenarien in dynamischen Umgebungen erschweren die vollständige Erfassung. Es ist daher schwierig, alle möglichen Situationen und Bedingungen zu antizipieren und zu modellieren. Diese ständigen Veränderungen in dynamischen Umgebungen erfordern, dass flexible und adaptive Ansätze zur Anforderungsanalyse und dem Anforderungsmanagement genutzt werden, um diese Änderungen auch nachträglich noch berücksichtigen zu können. Der in Kapitel 3.3.1 beschriebene Prozess berücksichtigt dies zwar theoretisch, was aber nicht heißt, dass dies in der Praxis auch leicht umzusetzen ist. [EWL+21]

Art der Anforderungsanalyse – Die Art der Anforderungsanalyse verändert sich durch die Entwicklung eines ADS ebenfalls. So können die Anforderungen für Level 2-Systeme als evolutionär betrachtet

werden, da sie auf bestehenden Systemen aufbauen und diese erweitern. Bei Level 4-Systemen handelt es sich jedoch um einen revolutionären Schritt, der eine völlige Neugestaltung der Anforderungsanalyse erfordert. [ZZM19]

Kommunikation zwischen Stakeholdern – Ein weiteres kritisches Element der Anforderungsanalyse ist die Kommunikation zwischen den verschiedenen Stakeholdern. Während es bei Level 2-Systemen oft genügt, die Anforderungen innerhalb eines Teams oder einer Organisation zu kommunizieren, erfordern Level 4-Systeme eine engere Zusammenarbeit zwischen verschiedenen Teams, Organisationen oder sogar Ländern. Dies führt nicht nur zu einer benötigten Multilingualität der Anforderungen, sondern auch zur Berücksichtigung von nationalen regulatorischen Besonderheiten (vgl. Kapitel 3.1). Auch hier stellt sich die Frage, wie die ODD diesen Punkt bestmöglich berücksichtigen kann. [KVK20]

Nationale Besonderheiten – Nationale Besonderheiten stellen weitere signifikante Herausforderungen für die Beschreibung einer ODD und die Entwicklung von ADS dar. Die diversen Verkehrsregeln und -gesetze der verschiedenen Länder erfordern eine sorgfältige Anpassung der Systeme, um Konformität und Sicherheit in jeder spezifischen Umgebung sicherzustellen. Diese regionalen, auch rechtlichen, Unterschiede, beeinflussen sowohl das Design als auch die Funktion der automatisierten Fahrsysteme. Hinzu kommen national unterschiedliche infrastrukturelle Gegebenheiten. Die Varianz in Straßentypen, Beschilderungen und Straßenmarkierungen zwischen verschiedenen Ländern kann enorm sein und erfordert eine gründliche Anpassung der Anforderungen. Zusätzlich spielen kulturelle Unterschiede eine zentrale Rolle. Da die Erwartungen und das Verhalten der Verkehrsteilnehmer kulturell geprägt sind, muss ein automatisiertes Fahrsystem in der Lage sein, auf ein breites Spektrum von Fahrverhalten zu reagieren und sich dementsprechend anzupassen. Dies erfordert eine sorgfältige Analyse und Einbeziehung kultureller Aspekte in die Anforderungsanalyse und Systemkonzeption. Des Weiteren müssen klimatische und geografische Besonderheiten berücksichtigt werden. Die Fähigkeit des Systems, in verschiedenen klimatischen Bedingungen und geografischen Gegebenheiten zu operieren (z. B. kalt bis heiß oder flach bis bergig), ist entscheidend und die Anforderungsanalyse muss daher diese verschiedenen Bedingungen in Betracht ziehen, um das gewünschte ADS zu entwickeln. [Sei22]

3.3.3. Informelle Sprachen zur Anforderungsbeschreibung

Vorausgehend wurde aufgezeigt, wie ein Definitionsprozess für die ODD aussehen kann und welche zentralen Herausforderungen dabei zu beachten sind. Darauf aufbauend, soll die aktuelle Entwicklungspraxis eingeordnet werden. Bei bisherigen Systementwicklungen, wie der Entwicklung eines Level 2-Spurhaltesystems, kommen vor allem informelle Beschreibungssprachen zur Anwendung. [AHD+18] Diese haben zwar ihre Vorteile, insbesondere in der frühzeitigen Phase des Systementwurfs oder bei der Kommunikation mit nicht technisch versierten Stakeholdern, bringen jedoch auch bedeutende Nachteile mit sich. Nachfolgend wird aufgezeigt, warum informelle Sprachen für die detaillierte Spezifikation einer ODD ungeeignet sein können:

- Mehrdeutigkeit: Das Hauptproblem informeller Beschreibungen liegt in ihrer potenziellen Mehrdeutigkeit. Ein und derselbe Satz oder Ausdruck kann auf verschiedene Weisen interpretiert werden, was zu Missverständnissen führen kann.
- Mangelnde Präzision: Informelle Beschreibungen können oft vage oder unvollständig sein, was zu Fehlinterpretationen führt und die Verifizierbarkeit der Anforderungen beeinträchtigt.
- Schwierige Verifikation: Es ist oft nicht möglich, informelle Anforderungen automatisch zu verifizieren. Dies kann den Test- und Validierungsprozess erschweren und das Risiko von Fehlern im Endprodukt erhöhen.

- Inkonsistenzen: Ohne eine formale Struktur können sich Inkonsistenzen in den Anforderungen einschleichen, die möglicherweise nicht sofort erkannt werden.
- Skalierungsprobleme: Bei großen und komplexen Systemen wird es schwieriger, den Überblick über informelle Anforderungen zu behalten und sicherzustellen, dass alle Aspekte berücksichtigt wurden.
- Schwierigkeit bei der Modellierung komplexer Bedingungen: Einige Aspekte, wie probabilistische Ereignisse oder komplexe Abhängigkeiten, sind mit Hilfe von informellen Sprachen schwer darstellbar.
- Fehlende Standardisierung: Informelle Beschreibungen folgen oft keinen festen Standards, was die Zusammenarbeit und Integration über verschiedene Teams oder Unternehmen hinweg erschweren kann. [Sha24]

3.4. Systemdesign

In der Phase des Systemdesigns liegt der Schwerpunkt auf der Umsetzung der zuvor definierten Anforderungen. Diese Phase ist entscheidend, da sie die Brücke von der theoretischen Planung hin zur praktischen Anwendung bildet. Das Entwicklungsteam arbeitet daran, die spezifischen ODD-Anforderungen in detaillierte Systemdesigns und technische Spezifikationen zu übersetzen. [BK21] Für diese Arbeit hat das Systemdesign eine geringe Bedeutung, da diese eine nachgelagerte Phase darstellt, in der die formalen Beschreibung angewendet wird. Die Forschung zielt darauf ab, die Grundlage für ein strukturiertes und effizientes Systemdesign zu schaffen, nicht jedoch das Design selbst zu entwickeln.

3.5. Sicherheitsbetrachtungen

Bei der Entwicklung von autonomen Fahrsystemen, insbesondere bei Level 4-Systemen, steht die Sicherheitsanalyse als eine der zentralen Herausforderungen im Fokus. Diese Systeme versprechen eine Revolution der Mobilität, bergen aber gleichzeitig ein vollkommen neues Ausmaß von Sicherheitsrisiken, welches es so vorher noch nicht gegeben hat. [BGM21] Insbesondere zwei Aspekte erfordern eine gründliche Analyse der potenziellen Sicherheitsrisiken. Erstens, der Wegfall eines verantwortlichen Fahrzeugführers, wodurch das ADS auch im Fehlerfall selbstständig sicher agieren können muss. Zweitens, das Beherrschen aller Umfeldelemente, die innerhalb der ODD spezifiziert sind. Beide Aspekte bergen eine Menge potenzieller Hazards und bedürfen einer umfassenden Betrachtung, um die Gesamtsicherheit des Systems zu garantieren. Dabei sind vor allem die Sicherheitsstandards ISO 26262 [Int11a] und die ISO/PAS 21448 [Int19] von besonderer Bedeutung.

Für beide gilt gleichermaßen, dass einer der kritischsten Schritte in der Sicherheitsanalyse der Sicherheitsnachweis ist. [Kni02] Dabei geht es darum, nicht nur theoretische Sicherheitskonzepte zu entwickeln, sondern auch die empirische Bestätigung ihrer Wirksamkeit im realen Betrieb nachzuweisen und zu argumentieren. Dies erfordert den Einsatz spezifischer Nachweismethoden, die die Sicherheit des Systems in verschiedenen Betriebsumgebungen und unter unterschiedlichen Bedingungen validieren. Für die erfolgreiche Entwicklung von ADS ist dies der Schlüssel, birgt aber sowohl für den Nachweis der funktionalen Sicherheit als auch für den Nachweis der SOTIF mehrere Herausforderungen. Speziell in Bezug auf die ODD werden diese nachfolgend im Detail beleuchtet.

3.5.1. Funktionale Sicherheit

Elektrik/Elektronik-Systeme sind sicherheitskritische Systeme, die im Straßenverkehr wichtige Entscheidungen im Namen des Menschen treffen. Diese Entscheidungen können die Sicherheit anderer Fahrzeuge, Objekte und Personen potenziell gefährden und zu Unfällen beitragen. [Bei12] Für den Betrieb im öffentlichen Verkehr müssen daher alle Entscheidungen des automatisierten Fahrzeugs

in allen gängigen und kritischen Situationen sowie bei internen Systemfehlern und dem Ausfall von Hardwarekomponenten sicher sein. Systemfehler dürfen nicht zu kritischen Situationen führen, in denen Personen oder Objekte zu Schaden kommen. Ein etabliertes Sicherheitsprinzip für sicherheitskritische Systeme ist die Annahme, dass diese Systeme so lange als unsicher gelten, bis eine überzeugende Argumentation des Gegenteils geliefert werden kann. [KW16] Der Nachweis der Produkthaftung und funktionalen Sicherheit hat daher eine entscheidende Bedeutung für den Automobilhersteller.

Der Standard an sich wurde jedoch mit traditionellen automobilen Systemen entwickelt. Dieser setzt daher implizit voraus, dass der Fahrer mit einer gewissen Wahrscheinlichkeit in der Lage ist, ein Versagen, das vom E/E-System ausgeht, zu kompensieren. Für die Entwicklung eines ADS bleiben die Schlüsselprozesse zwar weiterhin gültig, jedoch erweist sich der Nachweis einiger Aufgaben innerhalb der ISO 26262 als deutlich schwieriger. So ist nämlich der Nachweis der Vollständigkeit der HARA sowie die Vollständigkeit der Verifikation nicht mit der Komplexität bei traditionellen automobilen Systemen vergleichbar. Ein Grund für diese Schwierigkeit ist, dass ein ADS, um ein SAE-Automatisierungslevel 3-5 zu erreichen, eine komplexe Menge von ineinandergreifenden Funktionen und Subsystemen benötigt, um die DDT selbstständig zu bewältigen. Ein weiterer Grund ist, dass das ADS mit der vollständigen Unsicherheit seiner Umgebung zurechtkommen muss. Traditionelle automobile Systeme haben dafür eine implizite ODD (allem, was sie auf der Straße ausgesetzt sein können), aber für ein ADS kann die ODD verwendet werden, um den Inhalt der HARA einzuschränken und anschließend den Umfang der Verifikation zu begrenzen. Ein geeignetes Format und ein effizienter Inhalt der ODD können somit die Vollständigkeit der HARA unterstützen sowie die erforderliche Verifikation beschränken. [GJW+20] Um dies verständlicher zu machen, wird nachfolgend auf die einzelnen Phasen der ISO 26262 eingegangen und Anknüpfungspunkte zur Integration der ODD werden beschrieben.

Gefahrenidentifikation – Die Identifikation von Gefahren in komplexen Steuerungssystemen erfordert eine umfassende Herangehensweise, die von der Definition des Begriffs „Item“ bis zur spezifischen Analyse verschiedener Automatisierungsniveaus reicht. Für Level 4-Systeme, die hochautomatisiert funktionieren, ist diese Gefahrenidentifikation nicht trivial. Hier müssen komplexe Szenarien berücksichtigt werden, in denen das System vollständig autonom agiert, auch in unvorhergesehenen oder nicht normierten Situationen. Die Gefahrenidentifikation muss dabei auch die Fähigkeit des Systems umfassen, ohne menschliches Eingreifen auf eine Vielzahl von Situationen innerhalb und auch außerhalb der ODD zu reagieren, einschließlich technischer Defekte oder ungewöhnlicher Verkehrsbedingungen. Dabei sind je nach Item und ODD angepasste Gefahren- und Risikobewertungen notwendig. Jedes ODD-Element, wie beispielsweise die Autobahn, bringt spezifische Risiken mit sich, die berücksichtigt werden müssen. Zum Beispiel könnten bei der Betrachtung einer ODD für Autobahnen die hohen Geschwindigkeiten und das Verhalten von LKW als spezifische Risikofaktoren analysiert werden. Für die frühzeitige Identifikation dieser Risikofaktoren können Analysetechniken wie FMEA oder FTA eingesetzt werden. Diese ermöglichen es, das Verhalten von „Items“ in verschiedenen Szenarien zu simulieren und potenzielle Gefahren frühzeitig zu erkennen. [Int11c, Int11d]

Risikobewertung – Die Vertiefung des ASIL-Konzepts und die damit verbundene Risikobewertung sind zentrale Aspekte in der Entwicklung und Bewertung von E/E-Systemen. Diese Klassifizierung nach ASIL basiert auf einer detaillierten Analyse, die sowohl die Art des „Items“, seine Anwendung, als auch die potenziellen Auswirkungen eines Fehlers berücksichtigt. Bei Level 2-Systemen, die eine aktive Interaktion des Fahrers erfordern, ist die Risikobewertung besonders durch die Kontrollierbarkeit des Systems durch den Fahrer geprägt. Es ist wichtig, das Risiko einer mangelnden Aufmerksamkeit oder eines verzögerten Eingreifens des Fahrers in die Bewertung einzubeziehen. Da der Fahrer letztlich die

Verantwortung trägt, ist die Bewertung der Kontrollierbarkeit durch den Fahrer entscheidend. Im Gegensatz dazu konzentriert sich die Risikobewertung bei Level 4-Systemen stärker auf die Systemzuverlässigkeit und dessen Fähigkeit, eigenständig und sicher in einer Vielzahl von Szenarien zu operieren. Die Bewertung der Schwere des potenziellen Schadens und der Wahrscheinlichkeit des Auftretens wird komplexer, da das System autonom in vielfältigeren und möglicherweise unvorhergesehenen Umgebungen agiert. Daher erfordern verschiedene ODD-Bedingungen unterschiedliche Sicherheitsmaßnahmen und Risikobewertungen. Beispielsweise könnte in einer urbanen Umgebung das Risiko von Kollisionen mit Fußgängern höher bewertet werden als auf einer Autobahn. [Int11c, Int11h]

Sicherheitsanforderung – In verschiedenen Phasen der Systementwicklung werden sicherheitsrelevante Anforderungen formuliert, die darauf abzielen, die zuvor identifizierten Risiken zu minimieren. Für Level 2-Systeme konzentrieren sich die sicherheitsrelevanten Anforderungen maßgeblich auf die Unterstützung oder Interaktion mit dem Fahrer, da dieser letztendlich die Verantwortung und Kontrolle behält. Die Sicherheitsanforderungen für Level 4-Systeme sind dagegen deutlich komplexer, da die gesamte Verantwortung für die Fahrzeugsteuerung in den definierten ODDs übernommen wird. Dazu gehören fortschrittliche Erkennungssysteme, Entscheidungsfindungsalgorithmen und Redundanzen in kritischen Systemen, um die Sicherheit auch bei Systemausfällen zu gewährleisten. So könnten beispielsweise Systeme, die in einer ODD für schlechtes Wetter konzipiert sind, spezifische Sensoren oder Algorithmen erfordern, um angemessen auf rutschige Straßenverhältnisse zu reagieren. [Int11d, Int11e, Int11f]

V & V – Der Schritt der Verifikation und Validierung (V & V) innerhalb der ISO 26262 befasst sich mit der Überprüfung, ob die Systemanforderungen korrekt und vollständig implementiert wurden. Bei Level 2-Systemen konzentrieren sich die Verifikation und die Validierung darauf sicherzustellen, dass bspw. die Assistenzsysteme korrekt funktionieren und effektiv mit dem menschlichen Fahrer interagieren. Die Tests umfassen die Überprüfung von Systemreaktionen in typischen Fahrszenarien und die Sicherstellung, dass das System den Fahrer korrekt alarmiert, wenn eine menschliche Intervention erforderlich ist. Für Level 4-Systeme ist die Verifikation und Validierung komplexer und umfasst Tests in simulierten und realen Umgebungen, um zu gewährleisten, dass das System innerhalb der ODD sicher navigieren, Hindernisse erkennen und auf unvorhersehbare Ereignisse reagieren kann. Dabei bestimmt die ODD, unter welchen spezifischen Bedingungen das System funktionieren soll und daher auch getestet werden muss. Ein System, das für den Betrieb in Schnee konzipiert ist, sollte in schneereichen Bedingungen getestet werden, um seine Zuverlässigkeit und Sicherheit aller E/E-Systeme in dieser spezifischen Umgebung zu gewährleisten. [Int11d, Int11g]

Dokumentation – Jede Phase des Sicherheitslebenszyklus verlangt spezifische Dokumentationsanforderungen. Die Dokumentation dient als Beweismittel dafür, dass der Entwicklungsprozess den Anforderungen des ISO 26262-Standards entspricht. Für Level 2-Systeme muss die Dokumentation darlegen, wie die Assistenzsysteme den Fahrer unterstützen und wie die Interaktion zwischen Mensch und Maschine gestaltet ist. Besonderes Augenmerk wird auf die Dokumentation der Benutzerhinweise und Warnungen gelegt, die das System im Falle notwendiger menschlicher Eingriffe bereitstellt. Für Level 4-Systeme ist eine umfangreichere Dokumentation erforderlich, die die vollständige Automatisierung in bestimmten ODDs beschreibt. Es müssen detaillierte Informationen über die Fähigkeit des E/E-Systems, in seiner spezifischen ODD automatisiert zu operieren, bereitgestellt werden, einschließlich der Strategien für unerwartete Szenarien und Systemausfälle. Die Dokumentation muss auch umfangreiche Testergebnisse und Validierungsverfahren umfassen, die zeigen, dass das System in der Lage ist, ohne menschliche Überwachung sicher zu operieren. [Int11g]

Die vorherige Ausführung zur ISO 26262 hat gezeigt, wie der Entwicklungsstand von traditionellen E/E-Systemen im Automobil aussieht und wo Unzulänglichkeiten in Bezug auf die ADS-Entwicklung bestehen. Dabei wurde auch beleuchtet, wie und in welchem Umfang die ODD mit einzelnen Prozessen des Standards interagiert. Darauf aufbauend soll nachfolgend exemplarisch gezeigt werden, wie mit Hilfe einer ODD, die Durchführung der funktionalen Sicherheitsanalysen unterstützt werden kann. Das Ziel der ODD muss es sein, die funktionalen Sicherheitsanalysen auf den Anwendungsfall zu beschränken. Dazu wird der Anwendungsfall selbst mit Hilfe der ODD modelliert und alle Betriebsbedingungen rund um das ADS werden quantifiziert. Wichtig ist hierbei die Einbeziehung aller relevanten Betriebsbedingungen in die ODD, da anders keine Vollständigkeit angenommen werden kann. Auf Basis der ODD kann dann eine HARA durchgeführt werden. Die sich daraus ergebenden Anforderungen bilden dann den Großteil der Systemspezifikation. Das ADS kann anschließend implementiert und gegen diesen Anforderungssatz verifiziert werden. Gemäß den Vorgaben aus der ISO 26262 [Int11a] kann damit angenommen werden, dass alle Entwicklungsprozesse, die auf der definierten ODD basieren, sicher sind und ein ausreichend niedriges Restrisiko aufweisen. Damit kann das System aus Perspektive der funktionalen Sicherheit freigegeben werden, sollte aber, sobald freigegeben, die Umgebung überwachen und die Betriebsbedingungen fortlaufend überprüfen. Weiterhin sollte das ADS auch im Feld überwacht werden, um mögliche Verstöße gegen die Sicherheitsanforderungen zu erfassen, bevor ein Ereignis mit erheblichem Schaden eintritt. [Int11g]

Für einen Teil dieses Vorgehens liefert Gyllenhammer [GJW+20] ein weiteres Beispiel, indem er einen Anwendungsfall mithilfe der ODD quantifiziert. Im Fokus der Analyse steht der Geschwindigkeitsunterschied während eines Überholvorgangs auf der Autobahn. Für das Beispiel analysiert Gyllenhammer Daten aus 500 Fahrstunden in Europa, den USA und China. Diese statistischen Daten zeigen die Häufigkeit der Geschwindigkeitsunterschiede (siehe Abbildung 3.8) und können genutzt werden, um die Ereignisse gemäß ISO26262 [Int11h] nach Schweregraden zu klassifizieren. Für die Ermittlung des Schweregrads wird das Szenario angenommen, dass der bewertete Schweregrad für den Fall gilt, dass das eigene Fahrzeug in die Gegenfahrbahn ausweicht. In diesem Fall kann die Deltageschwindigkeit direkt mit verschiedenen Schweregraden verknüpft werden. Je nach Klassifizierung resultieren dann unterschiedliche Anforderungen an die funktionale Sicherheit des ADS. So könnte ein Anwendungsfall mit geringen Geschwindigkeitsunterschieden und niedrigem Schweregrad, wie etwa das Fahren in städtischen Gebieten, weniger strenge Sicherheitsanforderungen benötigen als ein Anwendungsfall mit hohen Geschwindigkeitsdifferenzen und schwerwiegenderen Folgen. Letzteres könnte beispielsweise beim Fahren auf Autobahnen der Fall sein. [GJW+20]

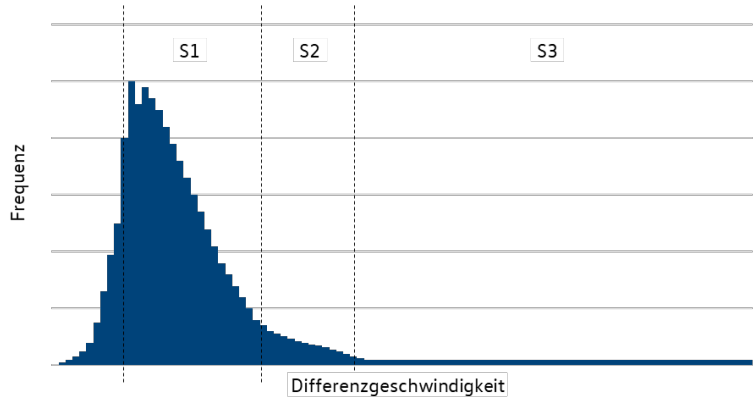

Abbildung 3.8: Schweregrad je Differenzgeschwindigkeit bei Überholmanövern, nach [GJW+20]

Der Fokus liegt darauf, für einen definierten Anwendungsfall mit Hilfe einer ODD die funktionale Sicherheit zu gewährleisten. Um dies zu ermöglichen, stellt sich vor allem die Frage, wie eine ODD beschrieben sein muss, um diese nutzbar für die Prozesse der ISO 26262 zu machen.

3.5.2. Sicherheit der intendierten Funktionalität

Die SOTIF ist für Level 4-Systeme aufgrund ihres höheren Automatisierungsgrades und der komplexeren Herausforderungen, die diese mit sich bringen, besonders relevant. Der Standard wurde entwickelt, um sicherzustellen, dass diese Systeme sicher funktionieren können, auch in Szenarien, die nicht durch herkömmliche Sicherheitsstandards abgedeckt sind. Dieses Ziel kann mit Hilfe eines Modells dargestellt werden (siehe Abbildung 3.9). Das Modell beinhaltet vier Bereiche, die Szenarien nach Bekanntheit und Gefahrenpotential klassifizieren.

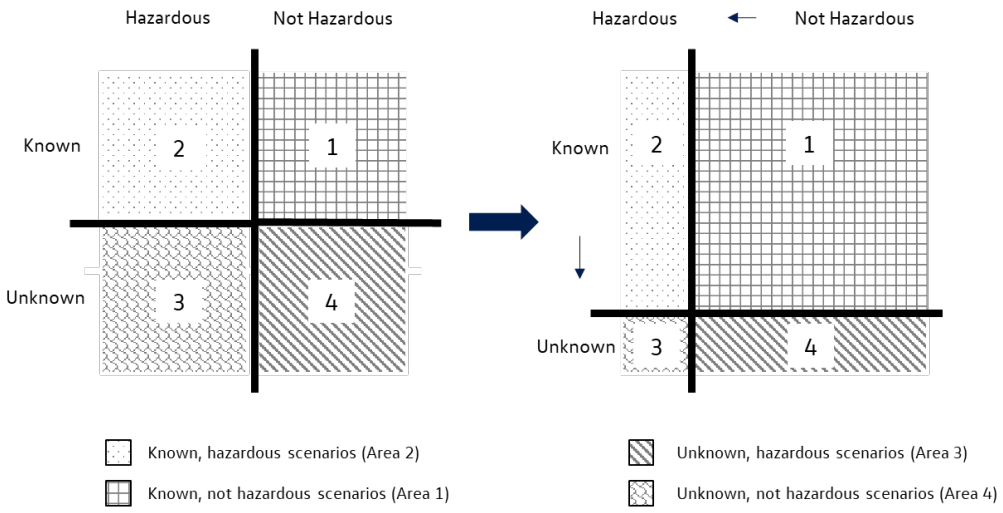

Abbildung 3.9: Entwicklung der Szenarienkategorien nach SOTIF, nach [Int19]

Der erste Bereich enthält bekannte und ungefährliche Szenarien. Dieser zielt darauf ab, den Satz bekannter Szenarien zu maximieren oder aufrechtzuerhalten, während die Bereiche 2 und 3 minimiert werden sollen. Diese Maßnahme erhält oder verbessert die Funktionalität. Der zweite Bereich enthält bekannte und gefährlich Szenarien. Dieser soll durch technische Maßnahmen auf ein akzeptabel niedriges Niveau minimiert und das potenzielle Risiko bewertet werden. Gegebenenfalls sollen gefährliche Szenarien durch Verbesserung der Funktion oder Einschränkung ihrer Nutzung (z. B. ODD-Restriktionen) in Bereich 1 verschoben werden. Im dritten Bereich sind unbekannte und gefährliche Szenarien enthalten. Der Bereich soll mit einem akzeptablen Aufwand so weit wie möglich minimiert werden, beispielsweise mit Hilfe von Validierungsaktivitäten (jedes entdeckte gefährliche Szenario wird in Bereich 2 verschoben). Der vierte Bereich enthält unbekannte und ungefährliche Szenarien. Die in diesem Bereich entdeckten Szenarien können dokumentiert und gemeldet werden, um den Bereich 1 zu maximieren. Das letztendliche Ziel der SOTIF-Aktivitäten ist die Bewertung des potenziell gefährlichen Verhaltens in den Bereichen 2 und 3 und die Bereitstellung eines Arguments, dass diese Bereiche minimal genug sind. Dies bedeutet, dass diese unter den definierten Akzeptanzkriterien liegen und das Restrisiko, das durch diese Szenarien verursacht wird, somit ausreichend gering ist. Während das Risiko, das von bekannten Szenarien in Bereich 2 ausgeht, explizit bewertet wird, wird das Risiko, das von unbekannten Szenarien in Bereich 3 ausgeht, nur durch branchenübliche Best Practices, wie systematische Analysen oder spezielle Experimente, abgeschätzt. Auf die erforderlichen SOTIF-Aktivitäten, um diese Form der Sicherheit zu gewährleisten, wird nachfolgend im Detail eingegangen (siehe Abbildung 3.10).

Die Aktivitäten der SOTIF beginnen mit der Erstellung einer Funktions- und Systemspezifikation, ähnlich einem Entwicklungsprozess oder dem Vorgehen der ISO26262. Diese wird laufend an neue Erkenntnisse angepasst und beschreibt die Ziele der beabsichtigten Funktion, die Abhängigkeiten zu anderen Fahrzeugfunktionen und -systemen, relevante Umweltbedingungen sowie die Ausgestaltung der Mensch-Maschine-Schnittstelle. Damit legt diese Spezifikation den Grundstein, um den Kontext für die nachfolgende Gefahren- und Risikoanalyse zu schaffen. [Int19]

Innerhalb der Gefahren- und Risikoanalyse werden zunächst potenzielle Szenarien identifiziert, in denen das System operieren soll. Dazu gehören typische Betriebsbedingungen sowie unerwartete oder seltene Ereignisse, insbesondere solche, in denen das System zwar korrekt funktioniert, aber dennoch ein Sicherheitsrisiko besteht. Die ODD spielt hierbei eine Schlüsselrolle, indem diese hilft, relevante Betriebsszenarien zu definieren. Im nächsten Schritt folgt die Gefahrenidentifikation. Hierbei werden potenzielle Gefahrenquellen identifiziert, die die sichere Ausführung der beabsichtigten Funktion beeinträchtigen könnten, wie z. B. Fehlinterpretationen von Sensordaten, unzureichende Algorithmen oder unerwartete Nutzeraktionen. Die ODD wird dabei genutzt, um zu verstehen, unter welchen spezifischen Bedingungen und in welchen Szenarien diese Gefahren auftreten könnten. Anschließend erfolgt die Risikoanalyse. Nach der Identifikation der Gefahren werden diese nach Häufigkeit (Exposure), Schwere (Severity) und Kontrollierbarkeit (Controllability) bewertet. Dieses Vorgehen ist analog zur ISO 26262. Bei L4-Systemen, bei denen kein Fahrer mehr anwesend ist, stellt die Bewertung der Kontrollierbarkeit von SOTIF-Risiken eine besondere Herausforderung dar. In traditionellen Systemen wird die Kontrollierbarkeit oft im Hinblick darauf bewertet, wie gut ein Fahrer auf ein potenzielles Risiko reagieren kann. Bei L4-Systemen muss dieser Ansatz angepasst werden, sodass der Schwerpunkt auf der Fähigkeit des Systems selbst liegt, Risiken zu erkennen, zu bewerten und darauf zu reagieren. Die Kontrollierbarkeit wird daher auf die Systemleistung bezogen, einschließlich Faktoren wie Systemreaktion, Selbstüberwachung und Redundanz, Kommunikation mit der Außenwelt, Anpassungsfähigkeit, Lernfähigkeit sowie Fallback-Strategien. [Int19]

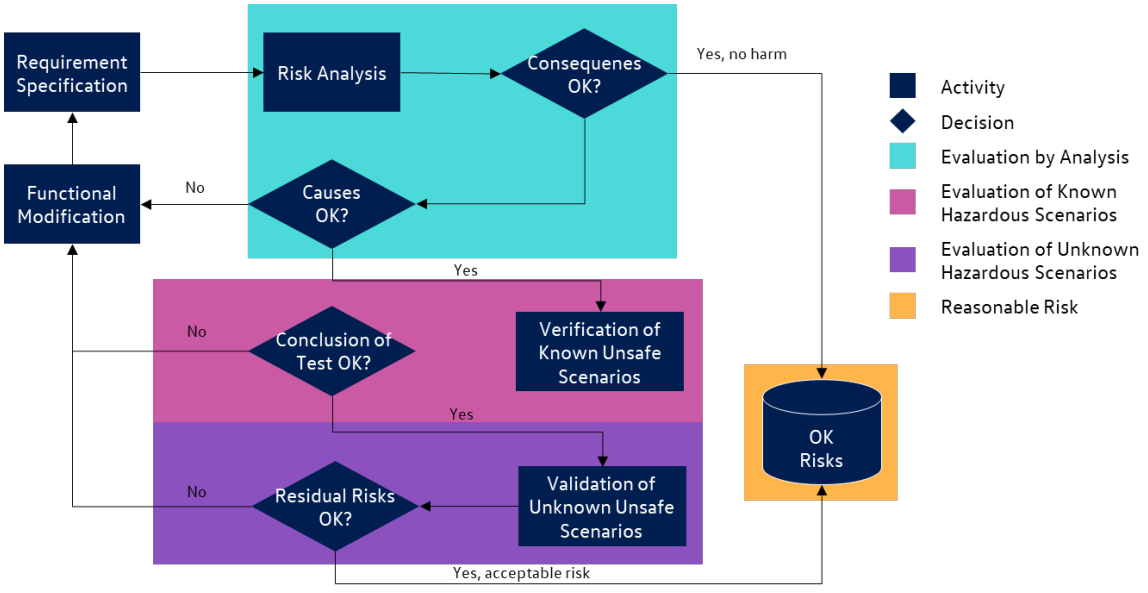

Abbildung 3.10: Abhängigkeiten der SOTIF-Aktivitäten, nach [Int19]

Der nächste Schritt befasst sich mit dem Erkennen und Bewerten von Unzulänglichkeiten in der Funktionalität eines Systems und den Bedingungen, die zu potenziell unsicherem Verhalten führen können. Dies umfasst zwei Dinge. Erstens die Identifikation funktionaler Unzulänglichkeiten, welche sich auf die Erkennung von Situationen, in denen ein System nicht wie vorgesehen oder erwartet

funktioniert, bezieht. Dabei kann es sich um Defekte, Fehlkonfigurationen oder Designmängel handeln, die zu einem unerwünschten Verhalten des Systems führen. Zweitens, die Bewertung von Triggering Conditions, wobei die spezifischen Bedingungen oder Szenarien identifiziert werden, unter denen diese funktionalen Unzulänglichkeiten auftreten können. Dazu gehört das Verständnis der Umstände oder Ereignisse, die zu einem gefährlichen Verhalten führen können. [Int19]

Als nächstes werden Strategien zur Risikominderung entwickelt. Dies kann die Anpassung des Systemdesigns, die Erweiterung der Funktionalitäten oder die Änderung der ODD umfassen. Während des Design- und Entwicklungsprozesses wird das System so gestaltet, dass es die Sicherheitsziele innerhalb seiner ODD erfüllt. Dies beinhaltet die Entwicklung von Funktionen und Kontrollmechanismen, die speziell auf die Herausforderungen und Einschränkungen der ODD zugeschnitten sind. Basierend auf der Gefahrenidentifikation und Risikoanalyse werden Sicherheitsziele entwickelt, die im Kontext der ODD des Systems betrachtet werden, um sicherzustellen, dass diese relevant für die spezifischen Betriebsbedingungen sind. [Int19]

Die Wirksamkeit der Risikominderungsmaßnahmen wird in der Phase der Verifikation und Validierung überprüft. Dies schließt Tests in simulierten und realen Umgebungen ein, um sicherzustellen, dass die Risiken effektiv reduziert wurden. [Int19]

Danach erfolgen die Ermittlung und Bewertung des Restrisikos. Hierbei wird zunächst die Wirksamkeit der zuvor implementierten Risikominderungsmaßnahmen überprüft, um zu bestimmen, inwieweit diese dazu beitragen, die identifizierten Risiken auf ein akzeptables Maß zu reduzieren. Dies geschieht oft durch eine Kombination aus praktischen Tests, Analysen von Simulationsdaten und theoretischen Überlegungen. Anschließend erfolgt die Ermittlung des verbleibenden Restrisikos. Trotz umfassender Sicherheitsmaßnahmen bleibt häufig ein gewisses Maß an Risiko bestehen. Dieses Restrisiko wird genau identifiziert und quantifiziert, um ein klares Bild der verbleibenden Gefahren zu erhalten. Ein wichtiger Aspekt ist die Entscheidung über die Akzeptanz dieses Restrisikos. Hierbei wird bewertet, ob das Restrisiko innerhalb tolerierbarer Grenzen liegt. Diese Bewertung basiert nicht nur auf technischen Daten, sondern berücksichtigt auch Industrienormen, gesetzliche Vorgaben und ethische Überlegungen. Es ist ein Abwägungsprozess, der sicherstellt, dass das System trotz der verbleibenden Risiken sicher genug für den Einsatz ist. [Int19]

Die Dokumentation und Berichterstattung dieses Prozesses sind ebenfalls von großer Bedeutung. Alle Erkenntnisse, Entscheidungen und die Gründe für die Akzeptanz des Restrisikos werden sorgfältig dokumentiert. Abschließend wird oftmals ein Plan für das weitere Monitoring und die Bewertung des Restrisikos im realen Betrieb erstellt. Dieses kontinuierliche Monitoring ist wichtig, um auf Änderungen in der Nutzungsweise oder in der Betriebsumgebung reagieren zu können und stellt sicher, dass das Sicherheitsniveau über den gesamten Lebenszyklus des Systems aufrechterhalten bleibt. [Int19]

Die bisherigen Beschreibungen der SOTIF-Prozesse spiegeln die Herausforderung und die Notwendigkeit wider, die Sicherheit und Zuverlässigkeit des zu untersuchenden Systems innerhalb der operationalen Grenzen, in Form der ODD, zu gewährleisten. Daraus ergeben sich vier grundlegende Use-Cases, bei denen die ODD eine entscheidende Bedeutung aufweist (siehe Abbildung 3.11).

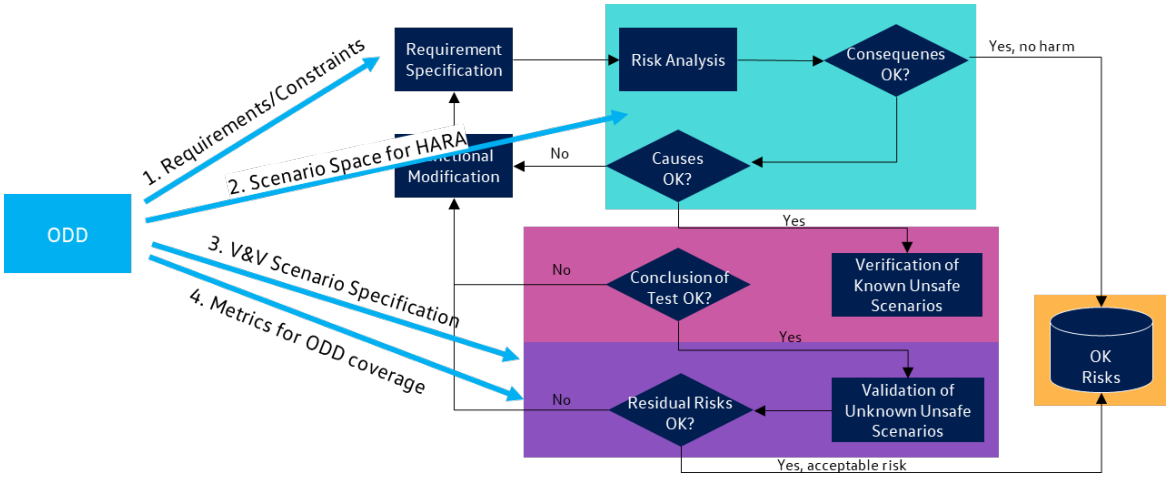

Abbildung 3.11: Einfluss der ODD auf die SOTIF Aktivitäten, nach [Kai21]

Der erste Use-Case innerhalb der SOTIF befasst sich mit der Systemspezifikation. Ähnlich wie auch bei der funktionalen Sicherheit, bildet die Systemspezifikation in Form von Anforderungen und Einschränkungen die Grundlage für alle weiteren Sicherheitsbetrachtungen. Diese hat deshalb eine besondere Bedeutung und erfordert eine detaillierte Berücksichtigung von ODD-spezifischen Betriebsbedingungen. Die zentrale Frage in diesem Kontext ist, wie formale ODD-Beschreibungen zur genauen Definition und Validierung der Anforderungen für AD-Funktionen genutzt werden können. Der zweite Use-Case konzentriert sich auf die Identifikation und Analyse von potenziell gefährlichen Szenarien innerhalb der ODD. Dieser Aspekt ist für die SOTIF von großer Wichtigkeit, da die Erkennung aller möglichen Risikoszenarien innerhalb der ODD essenziell für die Integrität und Sicherheit des Systems ist. Die Herausforderung besteht vor allem darin, den relevanten Szenarienraum für nachgelagerte Betrachtungen zu identifizieren. Im Detail geht es darum, zu beurteilen, welche Szenarien relevant für eine ODD sind und welche nicht (vgl. Abbildung 3.12). Wenn dies mit ausreichender Sicherheit gewährleistet werden kann, können die Sicherheitsaktivitäten auf die relevanten Szenarien beschränkt werden. Damit kann der gezielte Einsatz der begrenzten Ressourcen sichergestellt werden und die Analyse von nicht relevanten Szenarien verhindert werden. [Kai21]

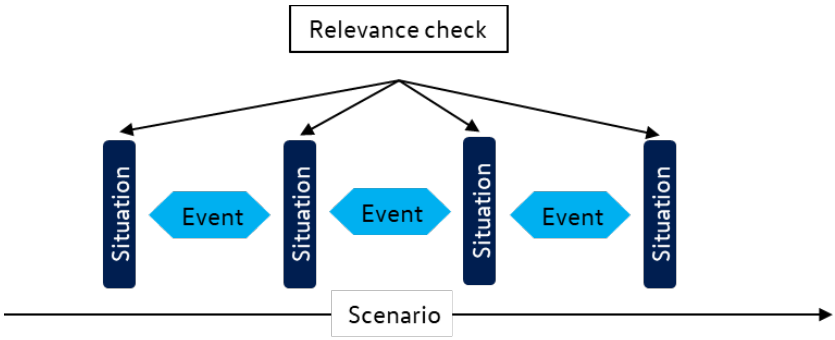

Abbildung 3.12: Relevanzcheck von Situation eines Szenarios, nach [RSR23]

Im dritten Use-Case geht es um die Auswahl und Spezifikation von Szenarien für die Validierung und Verifizierung von AD-Systemen innerhalb der SOTIF, basierend auf der ODD. Die richtige Auswahl und Spezifikation dieser Szenarien sind entscheidend, um sicherzustellen, dass die AD-Systeme unter allen relevanten Bedingungen innerhalb der ODD sicher funktionieren. Die Herausforderung besteht darin, zu untersuchen, wie verschiedene Datenquellen genutzt werden können, um relevante Szenarien für die ODD zu identifizieren und zu spezifizieren. Der vierte Use-Case beinhaltet die Entwicklung von Metriken zur Bewertung, inwieweit die V&V-Aktivitäten die ODD abdecken. Die Bewertung der ODD-

Abdeckung durch die Verifikation und Validierung ist von großer Bedeutung, um mögliche Sicherheitslücken zu identifizieren und die Wirksamkeit der Sicherheitsmaßnahmen zu beurteilen. Dieser Use-Case beinhaltet die Herausforderung, quantitative Methoden zu entwickeln, um zu messen, wie umfassend die ODD durch die V&V-Aktivitäten abgedeckt wird, und welche Bereiche möglicherweise weiterer Aufmerksamkeit bedürfen. [Kai21]

Zusammenfassend sind diese Use-Cases für die ODD im Kontext der SOTIF von entscheidender Bedeutung. Diese adressieren spezifische Herausforderungen bei der Gewährleistung der Sicherheit und Zuverlässigkeit von AD-Systemen innerhalb ihrer operationalen Grenzen und stellen einen wesentlichen Bestandteil für die erfolgreiche Entwicklung und Bewertung von L4-Systemen dar.

3.6. Testprozess

In der Testphase liegt der Fokus auf der Überprüfung der Sicherheit des ADS durch das Testen von Szenarien, die im Rahmen der Sicherheitsaktivitäten entwickelt wurden. Verschiedene Testmethoden kommen zum Einsatz, um diese Szenarien zu evaluieren. Ein entscheidender Aspekt dabei ist die Verwendung der ODD, die dazu dient, festzustellen, ob das ADS innerhalb seiner definierten Betriebsgrenzen agiert. Ein Test gilt als bestanden, wenn nachgewiesen werden kann, dass das ADS durchgehend innerhalb dieser Grenzen funktioniert. Andernfalls wird der Test als nicht bestanden gewertet. Diese Phase ist somit essenziell, um die Sicherheit und Zuverlässigkeit des Systems zu gewährleisten. [OK23] Für die vorliegende Arbeit hat diese Phase eine geringe Bedeutung, da die Lösung sich primär auf die Beschreibung von Szenarien anwenden lassen muss und sich nicht mit deren Abtesten beschäftigt. Für weiterführende Arbeiten ist dies jedoch ein interessanter Aspekt.

3.7. Fahrzeugbetrieb

Der Betrieb eines ADS stellt ähnliche Herausforderungen wie die Testphase dar, mit dem wesentlichen Unterschied, dass das ADS kontinuierlich und in Echtzeit bestimmen muss, ob es sich innerhalb seiner ODD befindet. Diese Anforderung gilt auch für Situationen, die unklar oder unbestimmt sind, was bedeutet. Dass es keine Ausnahmen gibt. Sollte das System erkennen, dass es sich außerhalb der definierten ODD bewegt, ist es zwingend erforderlich, dass es selbstständig Maßnahmen ergreift, um sich in einen Zustand zu versetzen, der das Risiko minimiert. Dies gewährleistet die Sicherheit des Systems und der Umgebung unter allen Umständen. Für die vorliegende Arbeit hat diese Phase eine geringe Bedeutung. [Sal22]

3.8. Gesamtbetrachtung Regulatorik, Business Case und Entwicklungsprozess

Dieses Kapitel fasst die Erkenntnisse der Problemanalyse zusammen und leitet daraus die Anforderungen an eine Lösung ab. Dazu wird eine Gesamtbetrachtung aus regulatorischen Rahmenbedingungen, Business Case und Entwicklungsprozess vorgenommen.

Die regulatorischen Rahmenbedingungen definieren die festen Voraussetzungen, die eine ODD erfüllen muss. Die wirtschaftliche Sicht bietet weitere Erkenntnisse hinsichtlich Relevanz der ODD für die Marktstrategie und die Einführung neuer Dienstleistungen. Sie hilft zu identifizieren, wo und unter welchen Bedingungen ein Service technisch umsetzbar und wirtschaftlich sinnvoll ist. Dies ermöglicht eine gezielte Markterschließung und eine effizientere Nutzung der Ressourcen.

Auf technischer Ebene ermöglicht die ODD eine präzise Definition der Betriebsbedingungen und Grenzen des ADS. Diese klare Abgrenzung ist für die zielgerichtete Systementwicklung und -optimierung entscheidend und dient als Basis für die Erstellung spezifischer Testfälle und Szenarien, was die Verifizierung und Validierung des Systems effizient und effektiv macht. Zudem trägt die ODD

zur Nachvollziehbarkeit und Transparenz im Entwicklungsprozess bei, indem sie als zentrale Informationsquelle und Dokumentationsgrundlage für alle Entwicklungs- und Testaktivitäten dient.

Durch die Kombination dieser wirtschaftlichen und technischen Aspekte wird deutlich, dass die ODD weit mehr als nur ein technisches Werkzeug ist. Vielmehr ist die ODD ein unverzichtbarer Bestandteil der Entwicklung und Implementierung von ADS, indem sie als "Single Point of Knowledge" (SPOK) dient (vgl. Abbildung 3.13). Diese zentrale Rolle der ODD trägt wesentlich zur Sicherheit, Effizienz und zum kommerziellen Erfolg von automatisierten Fahrsystemen bei und unterstreicht die Bedeutung als integralem Bestandteil. [RRS22]

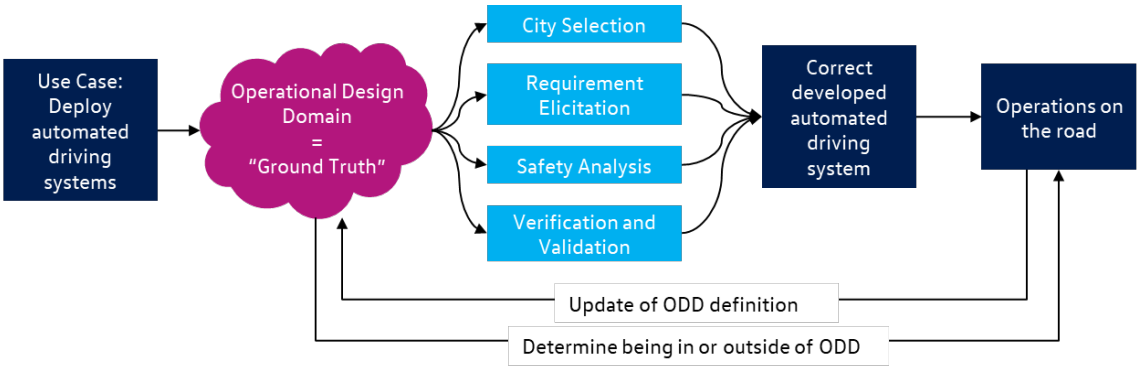

Abbildung 3.13: Darstellung der ODD als Single Point of Knowledge, nach [RRS22]

Nachfolgend wird auf die Erkenntnisse der Problemanalyse eingegangen.

3.8.1. Anforderungen aus den regulatorischen Rahmenbedingungen

Aus regulatorischer Sicht sind Genehmigungsverfahren und die Einhaltung gesetzlicher Anforderungen für die Freigabe und den Betrieb von ADS entscheidend. Dafür ist eine leicht verständliche ODD essenziell, um die Regulierung und Überwachung automatisierter Fahrsysteme effektiv zu gestalten. Klare ODD-Beschreibungen ermöglichen es, die Einhaltung von Sicherheits- und Zulassungsstandards zu prüfen und sind daher relevant für Genehmigungsprozesse. Zudem stärkt eine zugängliche und verständliche ODD das öffentliche Vertrauen, indem sie Transparenz schafft und Missverständnisse reduziert.

> **Anforderung REG 1**: Die ODD-Beschreibung muss in einer leicht verständlichen Sprache vorliegen, sodass diese auch von „Nicht-Experten“ verstanden werden kann.

Für Behörden ist zudem die Nachvollziehbarkeit der Entwicklung von hoher Relevanz. Es muss gewährleistet sein, dass automatisierte Fahrsysteme gemäß den definierten Einsatzbedingungen entwickelt und betrieben werden. Eine klare Verbindung zwischen der ODD, den Entwicklungs- und den Testaktivitäten ermöglicht es den Behörden, die Konformität mit Sicherheitsstandards und gesetzlichen Vorgaben effektiv zu überprüfen.

> **Anforderung REG 2**: Eine Nachvollziehbarkeit zwischen Geschäfts-/Sys/SW/HW-Anforderungen und der ODD-Definition muss herstellbar sein.

> **Anforderung REG 3**: Eine Nachvollziehbarkeit zwischen der ODD und dem Betriebsbereich muss gegeben sein.

Mit der Zeit kann es vorkommen, dass eine freigegebene ODD angepasst werden muss. Dies kann aufgrund neuer regulatorischer Vorschriften oder durch System- bzw. Softwareänderungen auf Seiten des Herstellers der Fall sein. Um fortlaufend die Aktualität und Compliance der ODD sicherzustellen,

soll durch das Hinzufügen neuer Module die ODD schnell an regulatorische oder herstellerseitige Änderungen angepasst werden können, ohne dass eine vollständige Überarbeitung nötig ist. Diese Flexibilität gewährleistet, dass automatisierte Fahrsysteme effizient aktualisiert werden können, um stets den neuesten Anforderungen zu entsprechen.

> **Anforderung REG 4**: Bei Änderungen muss die ODD erweiterbar sein, ohne dass eine vollständige Überarbeitung notwendig ist.

3.8.2. Anforderungen aus dem Business Case

Ähnlich wie Regulierungsbehörden, sieht auch das BCM die Notwendigkeit einer leicht verständlichen Beschreibung der ODD, um die Fähigkeiten des ADS effektiv an potenziellen Kunden zu kommunizieren. Daneben hat die ODD eine hohe Bedeutung für die Bewertung und Definition von Einsatzorten, um einen sicheren, aber vor allem auch wirtschaftlichen Betrieb von ADS zu gewährleisten. Dabei wird die Eignung einer OD für spezifische Anwendungsfälle, wie zum Beispiel MaaS oder TaaS, analysiert [RRS22]. Ein zentraler Punkt ist die Fähigkeit, mehrere Anwendungsfälle unter verschiedenen Umweltbedingungen zu modellieren. Als konkretes Beispiel wird die Analyse von Städten wie New York und Doha für eine MaaS-Anwendung auf der Grundlage einer spezifischen Systembeschreibung genannt. [RRS22]

> **Anforderung BCM 1**: Anhand der OD und der ODD soll bestimmbar sein, auf welchen Straßen das ADS technisch fahren kann.

Ein weiterer entscheidender Aspekt für den Service ist die Betriebsdauer des Systems, die maßgeblich von meteorologischen Parametern wie beispielsweise Regen beeinflusst wird. Für die Planung und Umsetzung von Dienstleistungen ist es daher essenziell, dass diese auf Basis der ODD-Servicezeiten bestimmt werden können.

> **Anforderung BCM 2:** Der räumliche und zeitliche Kontext von Umweltattributen der OD muss beschreibbar sein.

> **Anforderung BCM 3**: Anhand meteorologischer beschriebener Parameter der OD und der Systemfähigkeiten der ODD, müssen Servicezeiten für ein ADS berechnet werden können.

3.8.3. Anforderungen aus der Anforderungsbeschreibung

Die primäre Verantwortung eines Anforderungsingenieurs liegt in der Identifikation relevanter Anforderungen, die durch eine sorgfältige Analyse der OD und des spezifischen Anwendungsfalls abgeleitet werden. Eine präzise Definition der ODD ist dabei unerlässlich, um die zulässigen und unzulässigen Umweltbedingungen sowie deren mögliche Kombinationen festzulegen. Mehlhorn [MDR+24] erweitert diese Betrachtung durch die Unterscheidung zwischen dem Fahrzeugbasisentwickler und dem Entwickler der Fahrsoftware. Während der Basisentwickler die physische Fahrzeugplattform entwickelt, die als Fundament für das ADS dient und grundlegende Merkmale wie die Passagierkapazität und die Reichweite bestimmt, sind für ihn genaue Informationen über Betriebsgrenzen wie die maximale Geschwindigkeit und den Wendekreis von entscheidender Bedeutung. Diese müssen durch eine adäquat definierte ODD vorgegeben werden. Der Fahrsoftware-Entwickler konzentriert sich auf die Softwareentwicklung innerhalb der festgelegten Grenzen der ODD, wobei die spezifischen Anforderungen an das ADS auf Basis der ODD definiert werden.

Um die Betriebsbedingungen für ADS zu definieren, empfiehlt es sich, diese in drei fundamentale Kategorien zu gliedern. Diese Aufteilung dient dazu, die unterschiedlichen Charakteristika jedes Bereichs präzise darzustellen. Durch eine solche Kategorisierung wird die systematische Erfassung und

detaillierte Spezifikation aller relevanten Umwelteinflüsse erleichtert. Die drei relevanten Bereiche sind Szenerie, dynamische Objekte sowie Wetter und Kommunikation (vgl. Abbildung 2.3). [Bri20, Int23] Im Detail beinhalten diese folgende Informationen, die durch eine Beschreibungssprache für die ODD abgebildet werden müssen:

Szenerie: Die Szenerie erfordert eine detaillierte Beschreibung der relevanten Infrastrukturelemente. Eine Beschreibungssprache muss in der Lage sein, die Vielfalt der Umgebungen abzubilden, in denen das ADS operieren könnte, von städtischen Straßen mit ihren Kreuzungen, Zebrastreifen und Verkehrssignalen bis hin zu ländlichen Wegen und Autobahnen.

Dynamische Objekte: Für dynamische Objekte muss die Sprache die Komplexität der Bewegungsmuster, Geschwindigkeiten und möglichen Interaktionen von anderen Fahrzeugen, Fußgängern und Radfahrern präzise beschreiben können.

Wetter und Kommunikation: Die Kategorie Wetter und Kommunikation stellt weitere spezielle Anforderungen an die Beschreibungssprache. Wetterbedingungen wie Regen, Schnee und Nebel haben direkte Auswirkungen auf die Fahrsicherheit und Sichtverhältnisse und müssen daher in der ODD beschrieben werden können. Zudem ist die Fähigkeit zur Beschreibung von Kommunikationsaspekten essenziell, um die Interaktion zwischen dem ADS und anderen Verkehrsteilnehmern oder der Infrastruktur zu ermöglichen.

> **Anforderung REQ1**: Die ODD muss die zulässigen und unzulässigen Betriebsbedingungen (Szenerie, Dynamische Elemente und Wetter) innerhalb der OD definieren.

Für die weitere Spezifikation der ODD, müssen die zulässigen und unzulässigen Betriebsbedingungen beschrieben werden können. Ein praktisches Beispiel ist die Spezifikation einer Parklücke. Hierbei muss für ein bestimmtes System die Parklücke eine Mindestlänge von 4 Metern und eine Breite von mindestens 2,4 Metern aufweisen. Um diese Beschreibung auch für nachfolgende Aktivitäten (z.B. Szenarienerstellung) nutzbar zu machen, muss eine technisch präzise, d. h. formalere Beschreibung, ermöglicht werden. Um das Beispiel zu beschreiben, muss die Lösung folgende Anforderungen erfüllen.

> **Anforderung REQ2**: Die ODD soll die Spezifikation von arithmetischen logischen Ausdrücken unterstützen.
>
> **Anforderung REQ 3**: Die ODD muss die Definition von Einheiten unterstützen, es sei denn, das Element ist einheitenlos.
>
> **Anforderung REQ 4**: Die ODD soll die Nutzung von mathematischen und logischen Operatoren, wie +, -, and, or, >, <, etc. unterstützen.

Des Weiteren kann es vorkommen, dass unterschiedliche Datentypen zur Beschreibung genutzt werden.

> **Anforderung REQ 5**: Die ODD soll die Nutzung unterschiedlicher Datentypen (bspw. enum, boolean, float etc.) ermöglichen.

Das obige Beispiel kann durch eine weitere Bedingung eingeschränkt werden. Das System kann in einer Parklücke mit der Mindestlänge von 4 Metern und einer Breite von mindestens 2,4 Metern parken, es sei denn, es liegt Schnee auf der Straße.

> **Anforderung REQ 6**: Die Sprache muss die Spezifikation von bedingten Aussagen oder eine reduzierte ODD unterstützen. Dies kann durch das Auferlegen von Einschränkungen auf den vollständigen Operationsbereich bestimmter Attribute erreicht werden.

Je nach Anwendungsfall und Betriebsbereich sind unterschiedliche Betriebsbedingungen vorhanden und/oder relevant. Zur Kategorisierung dieser werden Taxonomien eingesetzt, die dann wiederum zur Beschreibung der Betriebsbedingungen genutzt werden.

> **Anforderung REQ 7**: Die Sprache soll die Nutzung von Taxonomien ermöglichen.
>
> **Anforderung REQ 8**: Die Sprache muss konsistent sein mit den in der Taxonomie definierten Kategorien (Keywords, Datentypen und Einheiten).

Während der Entwicklung verändern sich oft die Anforderungen. Es kommen neue hinzu, es werden bestehende gestrichen oder vorhandene werden detailliert. Diese führt dazu, dass eine Taxonomie im Laufe der Entwicklung an Gültigkeit verliert und an neue Gegebenheiten angepasst werden muss. Ein Beispiel hierfür ist die flächendeckende Einführung des E-Rollers, den es so vorher in der Taxonomie vielleicht nicht gab.

> **Anforderung REQ 9**: Die Sprache soll das Hinzufügen neuer Elemente zu einer bestehenden Taxonomie unterstützen.

Die Entwicklung eines ADS ist ein komplexer Prozess, der die simultane Arbeit mehrerer Teams erfordert. Diese Teams bearbeiten parallel unterschiedliche Anforderungen, wobei jeweils verschiedene Aspekte der ODD für ihre spezifischen Implementierungen von Bedeutung sind. Beispielsweise sind Kurvenradien entscheidend für das Design des Fahrwerks und des Wendekreises, während die Auswahl geeigneter Sensoren stark von Wetterbedingungen wie Nebel abhängen kann. Angesichts dieser Vielschichtigkeit ist eine Segmentierung der ODD in spezifische Teilbereiche notwendig. Eine solche Unterteilung ermöglicht es, die unterschiedlichen Fähigkeiten und Anforderungen des ADS präzise zu beschreiben und zuzuordnen. Durch die Aufteilung der ODD in klar definierte Segmente können Teams gezielt an den für ihre Arbeit relevanten Bereichen arbeiten, ohne die Übersicht über das Gesamtsystem zu verlieren.

> **Anforderung REQ 10**: Die ODD-Beschreibung soll die systematische Modularisierung der ODD in Teilbereiche unterstützen, um unterschiedliche Fähigkeiten des AD-Systems getrennt zu beschreiben.

Durch diese Modularisierung wird der Entwicklungsprozess beschleunigt, da bereits definierte ODD-Module für neue Projekte wiederverwendet werden können, ohne dass eine Neukonzeption notwendig ist. Die Möglichkeit, komplexe Bedingungen und Szenarien durch die intuitive Kombination von Grundbausteinen genau zu spezifizieren, vereinfacht nicht nur die Erstellung und Anpassung der ODD an spezielle Erfordernisse, sondern fördert auch die Skalierbarkeit und Flexibilität der verwendeten Beschreibungssprache. Diese modulare Herangehensweise bildet somit eine wesentliche Grundlage für die effiziente Entwicklung und Implementierung von ADS, indem sie eine flexible, skalierbare und präzise Modellierung der Betriebsdomäne ermöglicht.

> **Anforderung REQ 11:** Die Sprache soll das Kombinieren und Wiederverwenden von ODD-Modulen ermöglichen. Dafür sollen Teilmengen einer ODD benannt werden können.
>
> **Anforderung REQ 12**: Module einer ODD sollen mit logischen Operatoren (And, Or) verknüpft werden können.

Neben der Definition von Anforderungen für die ODD, ist der Erstellungsprozess dieser Anforderungen von wesentlicher Bedeutung. Ein entscheidender Teil dieses Prozesses ist die Berücksichtigung der OD und der CODs, aus denen die OD besteht (vgl. Kapitel 2.1.2). Der Abgleich der ODD mit OD bzw. COD

gewährleistet, dass das Fahrzeug in diesen sicher operieren kann. Hierfür ist es notwendig, die relevanten Parameter der OD/COD und deren Bereiche gründlich zu analysieren und zu verstehen.

Anforderung REQ 13: Die OD und COD müssen repräsentiert werden können.

Anforderung REQ 14: Die in der OD und COD enthaltenen Parameter müssen mit Hilfe mathematischer Funktionen (z. B. max(), min(), distance()) ausgewertet werden können.

Anforderung REQ 15: Die OD/COD muss mit der ODD abgeglichen werden können. Dafür muss für jede ODD-Bedingung bestimmt werden können, ob sich diese innerhalb oder außerhalb der OD/COD befindet. Dafür soll eine binäre Auswertung mit True/False genutzt werden.

Darüber hinaus muss die ODD es ermöglichen, eine Rückverfolgbarkeit bzw. ein Mapping zwischen den Situationen des Betriebsbereichs und der ODD herzustellen, um eine Relevanz der Anforderungen zu belegen.

Anforderung REQ 16: Rückverfolgbarkeit zwischen ODD und relevanten OD/CODs.

Nach der Festlegung der Anforderungen spielen die Betriebsbedingungen der ODD eine kontinuierliche Rolle im Entwicklungsprozess. Insbesondere für die Entwicklung von Szenarien und Testfällen ist es essenziell, dass diese systematisch, anhand der ODD-Beschreibung überprüft, werden können. Dies gewährleistet, dass die Testergebnisse nachvollziehbar und verlässlich sind. Für die effektive Erstellung und Auswertung von Szenarien und Testfällen ist daher eine maschinenlesbare und technisch exakte Beschreibung der ODD unerlässlich (vgl. Kapitel 3.5).

Anforderung REQ 17. Die ODD-Spezifikation muss maschinenlesbar sein.

3.8.4. Anforderungen aus der Sicherheitsbetrachtung

Auf Basis der Sicherheitsbetrachtung haben die beiden Normen ISO 26262, für die funktionale Sicherheit, und die ISO 21448, für die Sicherheit der intendierten Funktionalität, eine besondere Bedeutung. Bei beiden Normen steht im Fokus, nicht nur theoretische Sicherheitskonzepte zu entwickeln, sondern auch die Bestätigung ihrer Wirksamkeit nachzuweisen und zu argumentieren. Dies erfordert den Einsatz spezifischer Nachweismethoden, die die Sicherheit des Systems gegenüber der ODD beweisen. Dafür ist entscheidend, eine nachvollziehbare und konsistente Systementwicklung über alle Entwicklungsschritte, angefangen bei der ODD, aufzuzeigen (vgl. Anforderung REG 2 und REG 3).

Anforderung SAF 1: Die ODD muss konsistent und widerspruchsfrei sein.

Daneben rückt die Frage in den Fokus, was die ODD abbilden muss, um die Sicherheitsanalyse effektiv zu unterstützen. Ein Schlüsselelement dabei ist die Integration von Häufigkeit und Wahrscheinlichkeit, die es ermöglicht, die Unsicherheit bezüglich des Auftretens spezifischer ODD-Elemente zu quantifizieren. Es ist unstrittig, dass nicht alle ODD-Elemente innerhalb der OD mit gleicher Wahrscheinlichkeit eintreten. Zum Beispiel tritt meist Nebel weniger häufig auf als Sonnenschein oder Regen. Mit der Angabe der Auftretenswahrscheinlichkeiten unterstützt dies wesentlich die Sicherheits- und Risikobewertung. Es hilft, das Risiko zu bewerten und Entscheidungen über die Akzeptanz des Risikos zu treffen. Zudem ermöglicht es eine zielgerichtete Zuweisung von Test- und Simulationsressourcen.

Anforderung SAF 2: Die ODD muss die Beschreibung von Häufigkeiten und Wahrscheinlichkeiten unterstützen.

Des Weiteren ist anzunehmen, dass nicht alle potenziellen Risikosituationen stets vollständig vermieden werden können. Ein kategorischer Ausschluss jeder Situation, die ein nicht nullwertiges Risiko birgt, wäre nicht praktikabel. Beispielsweise wäre es unlogisch, Autobahnen vollständig zu meiden, nur weil es eine geringe Chance gibt, aufgrund von Unfällen, Staus oder anderen Ursachen, auf Personen zu treffen. Daher muss die Beschreibungssprache der ODD in der Lage sein, eindeutige in-oder-out-Entscheidungen für Situationen zu treffen, die unklar oder mit Unsicherheiten behaftet sind.

> **Anforderung SAF 3:** Die ODD muss auch für fuzzy oder unklare Situationen eine eindeutige in-oder-out Entscheidung treffen.

Neben unklaren oder fuzzy Situationen können ebenfalls unvollständige Situationen auftreten. Dies kann beispielsweise an Limitationen der Sensortechnologie liegen, die nicht alle Umgebungsvariablen erfassen kann. Aus diesem Fall ergibt sich die Notwendigkeit, Entscheidungen über die Zulässigkeit eines Einsatzbereiches (in-oder-out) für das ADS zu treffen, selbst wenn relevante Informationen fehlen. Die Entscheidungsfindung hängt dabei wesentlich vom jeweiligen ODD-Element und seiner Bedeutung für die Sicherheit des Systems ab. Je nach Sicherheitsrelevanz können die Konsequenzen einer solchen Entscheidung variieren, wobei eine sorgfältige Abwägung der Risiken und der vorhandenen Informationen unerlässlich ist. Es folgt, dass trotz des Fehlens relevanter Informationen eine binäre in-oder-out Entscheidung getroffen werden muss.

> **Anforderung SAF 4**: Die ODD muss auch für unvollständige Situationen eine eindeutige in-oder-out-Entscheidung treffen.
>
> **Anforderung SAF 5**: Die Entscheidung, ob eine Situation in-oder-out ist, muss kontextabhängig festgelegt werden können.

Nach der Fertigstellung der ODD-Beschreibung erfolgt im Sinne der Absicherung die Erstellung von Szenarien. In diesem Zusammenhang ist es von entscheidender Bedeutung, dass die Szenarien direkt aus der ODD abgeleitet werden können. Dieser Prozess gewährleistet, dass spezifische Szenarien, die innerhalb der definierten Grenzen der ODD liegen, systematisch identifiziert und entwickelt werden können. Ebenso ermöglicht es die Identifikation und Entwicklung von Szenarien, die explizit außerhalb der ODD angesiedelt sind. Diese methodische Herangehensweise unterstützt die präzise Analyse und Bewertung des ADS unter variierenden Bedingungen und fördert ein umfassendes Verständnis der Einsatzgrenzen des Systems.

> **Anforderung SAF 6:** Die ODD muss die Ableitung von relevanten Szenarien unterstützen.

Zudem besteht die Notwendigkeit, die Abdeckung der betrachteten Szenarien in Bezug auf die gesamte ODD zu demonstrieren und das verbleibende Risiko von Gefahrenszenarien innerhalb der ODD, die nicht durch das Sicherheitskonzept und den Validierungsansatz abgedeckt wurden, abzuschätzen (vgl. Kapitel 3.5). Um dies zu gewährleisten, wird mehr benötigt als eine einfache qualitative Auflistung von Objekten und Bedingungen. Es werden präzise Informationen zu spezifischen Aspekten der ODD benötigt. Dies erstreckt sich von einfachen Ja/Nein-Fragen, ob ein bestimmtes Szenario innerhalb der ODD liegt, bis hin zu komplexeren Beurteilungen, die sowohl qualitative als auch quantitative Bewertungen umfassen. Als Beispiel kann hier die prozentuale Abdeckung der ODD durch Testfälle genannt werden.

Die zu entwickelnde Sprache muss daher die Definition spezifischer Metriken ermöglichen, um verschiedene Eigenschaften und Qualitätsattribute messbar zu machen. Dies schließt Aussagen über die Qualität, Vollständigkeit und den Fortschritt in Bezug auf die ODD-Spezifikation oder deren

Anwendung ein. Solche Metriken sind entscheidend für den Vergleich unterschiedlicher ADS auf Basis ihrer ODDs oder für die Bewertung von Testszenariensätzen, die zur Abdeckung einer gegebenen ODD verwendet werden.

> **Anforderung SAF 7:** Die ODD muss die Definition spezifischer Metriken zur quantitativen und qualitativen Beurteilung ermöglichen.

3.8.5. Anforderung aus dem Testprozess

Das Testen ist der vorrangige Ansatz für die Verifizierung und Validierung, wie von ISO 26262 und ISO/PAS 21448 empfohlen (vgl. Kapitel 3.5) und wird entwicklungsbegleitend durchgeführt. Dafür werden spezifische Testverfahren in den Entwicklungsablauf, sowohl auf der Systemebene als auch auf den Ebenen der Hardware und Software, integriert. Dies dient dazu, zu gewährleisten, dass sowohl das Gesamtsystem als auch die einzelnen Hardware- und Softwarekomponenten den jeweiligen funktionalen und sicherheitstechnischen Anforderungen genügen. In diesem Fall vor allem in Bezug auf die ODD.

Aus Sicht des Testens muss die ODD dafür vor allem die Anforderungen REQ17 und SAF 3-7 unterstützen. Dies gewährleistet eine automatisierte Verarbeitung, Erstellung und Durchführung von Testfällen auf Basis der in den Sicherheitsaktivitäten erstellten Szenarien. Daneben ist für die Durchführung und Bewertung der Tests entscheidend, dass die ODD die Integration von Mechanismen zur systematischen Überprüfung der Übereinstimmung des Fahrzeugverhaltens mit den in der ODD festgelegten Grenzen ermöglicht. Dieser Prozess, der sowohl den Abgleich von Fahrzeugverhalten mit der ODD als auch die Evaluierung der Compliance umfasst, ist entscheidend, um zu gewährleisten, dass das ADS jederzeit innerhalb seiner definierten Betriebsparameter operiert.

> **Anforderung TES 1:** Die ODD muss die Überprüfung des Fahrzeugverhaltens auf Basis verschiedener Dateninputs (Simulation, Realfahrt, etc.) ermöglichen.

3.8.6. Anforderungen aus dem Fahrzeugbetrieb

Während des Betriebs des ADS muss es kontinuierlich sichergestellt sein, dass das ADS sich innerhalb der festgelegten und freigegebenen Grenzen befindet. Dabei ist es wichtig, dass die ODD-Spezifikation in Echtzeit verarbeitet und gegen gemessene Sensordaten überprüft werden kann (vgl. Anforderungen REQ17, TES1). Beispielsweise muss das ADS beim Befahren einer Baustelle, die außerhalb seiner definierten Grenze liegt, entsprechend reagieren. Dafür muss jederzeit eine binäre in-oder-out-Entscheidung bezüglich der ODD getroffen werden können (vgl. Anforderungen SAF3-5, TES1).

3.8.7. Übergreifende Anforderungen

Aus den bisherigen Analysen ergeben sich zudem weitere Anforderung an die Performance und Skalierbarkeit der Nutzung der ODD.

> **Anforderung ÜBG 1:** Die Auswertung soll hinsichtlich der Anzahl der Formelvariablen skalierbar sein.

> **Anforderung ÜBG 1:** Die Auswertung soll hinsichtlich der Anzahl der betrachteten CODs skalierbar sein.

Neben den bisher genannten Stakeholdern und Anforderungen können weitere Anforderungen aus [Asa21, MDR+24, RRS22] entnommen werden.

4. Erörterung der Forschungsfragen

Dieses Kapitel widmet sich der Erörterung der Forschungsfragen und der Präsentation des Lösungskonzeptes. Dafür greift das Kapitel 4.1 die in Kapitel 3 formulierten Anforderungen auf und fasst diese zu einem übergeordneten Problem zusammen. Für diesen Zweck werden fünf Anforderungscluster identifiziert und beschrieben, die eine Lösung erfüllen muss. Anhand dieser Analyse wird in Kapitel 4.2 der Stand der Wissenschaft erläutert. Darauf aufbauend definiert Kapitel 4.3 definiert die Forschungsfragen und Kapitel 4.4 schließt mit der Vorstellung des Lösungskonzeptes.

4.1. Übergeordnetes Problem

Die ODD beschreibt per Definition die Umgebungsbedingungen, die ein ADS bewältigen sollte und für die es konzipiert ist. Diese Bedingungen umfassen eine Vielzahl von Variablen, die von Wetterbedingungen bis hin zu Straßenarten und sogar Verkehrsteilnehmern reichen. Die Komplexität und Vielfalt dieser Umgebungsbedingungen variieren erheblich und unterscheiden sich je nach beabsichtigten Einsatzbereich des ADS. In städtischen Gebieten beispielsweise muss das ADS in der Lage sein, sich durch dichten Verkehr, unvorhersehbare Fußgänger und eine Vielzahl von Signalen und Straßenmarkierungen zu manövrieren. Im Gegensatz dazu könnten Autobahnumgebungen zwar weniger variabel sein, dafür aber möglicherweise höhere Geschwindigkeiten und andere Arten von Hindernissen beinhalten.

Angesichts dieser Herausforderungen wurden in der Literatur verschiedene Ansätze zur Strukturierung und Klassifizierung dieser Bedingungen vorgeschlagen. Es existieren vielfältige Taxonomien [Bri20, Cza18, Nat20, Sae21], die darauf abzielen, diese komplexen Umgebungen in einer systematischen, wenn auch abstrakten Weise, zu beschreiben. Diese Taxonomien dienen als Werkzeuge zur Erstellung einer detaillierten und umfassenden ODD, indem sie Kriterien und Kategorien bereitstellen, die das breite Spektrum realer Fahrbedingungen abdecken. Sie ermöglichen es Entwicklern, kritische Bedingungen zu identifizieren und zu analysieren, die das ADS in verschiedenen Szenarien bewältigen muss.

Auf Basis einer solchen Taxonomie, können je nach spezifischem Anwendungsfall die relevanten Taxonomieelemente für die ODD ausgewählt und detailliert spezifiziert werden. Dieser Prozess ermöglicht eine maßgeschneiderte ODD, die die spezifischen Anforderungen und Herausforderungen eines bestimmten Einsatzszenarios widerspiegelt. Unternehmen wie Waymo [Cal22b] und Cruise [Cal22a] nutzen diesen Ansatz, um ODDs zu erstellen. Diese sind listenhafte Aufzählungen und enthalten Bedingungen, die das ADS bewältigen kann, und Bedingungen, die außerhalb Fähigkeiten liegen (vgl. Abbildung 4.1). Diese Form der ODD-Beschreibung bietet eine Grundlage für eine verständliche Darstellung der Fähigkeiten und Grenzen des Systems.

Operational Design Domain v1
Zulässige Straßentypen sind:
- *Autobahnen*
- *Spielstraßen*

Zulässige Regenkategorien sind:
- *leichter Regen bis 2,5 mm/h*
- *moderater Regen bis 7,6mm/h*

Abbildung 4.1: Beispielhafte ODD auf Basis einer Liste, nach [Cal22a, Cal22b]

Allerdings weist dieser Ansatz auch signifikante Einschränkungen auf, da Umweltbedingungen isoliert und statisch betrachtet werden. Diese Betrachtung stimmt nicht mit der dynamischen Natur echter Fahrbedingungen überein. In der Realität beeinflussen sich verschiedene Bedingungen gegenseitig und

bilden ein komplexes Netzwerk von Herausforderungen, das durch ein solches Modell nur unzureichend erfasst wird. Beispielsweise kann moderater Regen auf Autobahnen ein hohes Risiko darstellen, während in Bereichen mit niedriger Geschwindigkeit dieser als weniger riskant gilt.

Weiterhin führt die vereinfachte Modellierung zu mehrdeutigen und unvollständigen Systemspezifikationen. So könnte ein ADS beispielsweise in Bereichen mit niedriger Geschwindigkeit auch bei starkem Regen operieren. Auf Autobahnen hingegen würde die zulässige Regenmenge aufgrund der höheren Geschwindigkeiten deutlich geringer sein. Da solche Unterscheidungen im Modell nicht abgebildet werden können, sind diese Differenzen auch in den Systemspezifikationen nicht klar erkennbar. Es ist daher notwendig, eine Beschreibungssprache für die ODD zu entwickeln, die die Wechselwirkungen und Dynamiken realer Betriebsbedingungen besser modellieren kann. Eine mächtigere Beschreibungssystematik ist notwendig, um die Dynamik und Komplexität echter Fahrbedingungen adäquater zu repräsentieren.

Um eine solche Beschreibungssprache für die ODD zu entwickeln, gilt es, die Anforderungen, die sich aus der beabsichtigten Nutzung ergeben, festzulegen. Dazu wurden die Anforderungen aus Kapitel 3.8 zu Clustern zusammengefasst.

Tabelle 4-1: Bildung von Anforderungsclustern

Anforderungscluster	Anforderung
Leichte Verständlichkeit und technische Präzision (RQ1.1)	REG1; REQ 17 SAF6
Repräsentation unterschiedlicher Informationsbedarfe (RQ1.1):	BCM2, REQ1, 2, 3, 4, 5, 6, 9, 13, 14 SAF2, 7
Modularität und Wiederverwendbarkeit ohne zusätzliche Komplexität (RQ 1.2):	REG4, 10, 11, 12
Quantitative und qualitative Bewertung der ODD (anhand von Inputdaten) (RQ2)	REG3; 4 BCM1, 3; REQ7, 8, 9, 14, 15, 16; SAF3, 4, 5, 7 TES1
Prozessintegration (RQ3):	REG2, 4 REQ16 SAF1, 6 ÜBG1, 2 TES1

Diese Cluster sind im Einzelnen:

- **Leichte Verständlichkeit und technische Präzision**: Einerseits ist für eine effektive Kommunikation mit Behörden und der Öffentlichkeit eine leicht verständliche Darstellung der ODD in natürlicher Sprache erforderlich. Eine abstrakte Beschreibung der Systemfähigkeiten erweist sich dabei als ausreichend. Andererseits erfordern Verifikations- und Validierungsprozesse eine präzise und eindeutige Sprache, die entweder semi-formal oder formal sein muss. Mit dieser müssen beispielsweise reale Szenarien modellierbar sein oder, komplexe Kombinationen von Betriebsbedingungen mit Hilfe logischer Operatoren erzeugbar sein.

- **Repräsentation unterschiedlicher Informationsbedarfe:** Bei der Gestaltung der ODD ist es entscheidend, dass diese die für jeden Stakeholder notwendigen Informationen abbilden kann, ohne dabei an Nutzbarkeit und Verständlichkeit für andere Stakeholder einzubüßen. Dazu gehört nicht nur die Repräsentation von unterschiedlichen Betriebsbedingungen, sondern beispielsweise auch die Abbildung unterschiedlicher Datentypen.
- **Modularität und Wiederverwendbarkeit ohne zusätzliche Komplexität:** Die Sprache muss modular aufgebaut sein, sodass einzelne Komponenten oder Module in verschiedenen Kontexten wiederverwendet werden können. Gleichzeitig sollte dies ohne Einführung unnötiger Komplexität erfolgen, um die Verständlichkeit und Handhabbarkeit zu gewährleisten. Diese Modularität erhöht die Flexibilität, indem sie einfache Anpassungen und Erweiterungen ermöglicht, ohne das Gesamtsystem zu beeinträchtigen. Sie unterstützt zudem die effiziente Zusammenarbeit in interdisziplinären Teams, die an unterschiedlichen Modulen arbeiten können. Darüber hinaus trägt die Aufteilung in kleinere Einheiten dazu bei, die Gesamtkomplexität des Systems zu reduzieren, wodurch Fehleranfälligkeit verringert, und Nutzerfreundlichkeit verbessert wird.
- **Quantitative und qualitative Bewertung der ODD (anhand von Inputdaten) –** Im Entwicklungsprozess stehen Stakeholder vor der Herausforderung, spezifische Aspekte der ODD zu verschiedenen Zeitpunkten zu bewerten und zu verstehen. Die Anforderungen an diese Bewertungen variieren stark und reichen von einfachen Ja/Nein-Entscheidungen, wie etwa der Frage, ob ein bestimmtes Szenario innerhalb der Grenzen der ODD liegt, bis hin zu komplexeren Beurteilungen. Letztere umfassen sowohl qualitative Einschätzungen – beispielsweise, wie gut die Abdeckung einer ODD durch eine Reihe von Szenarien oder Testfällen ist – als auch quantitative Bewertungen, wie die Bestimmung zulässiger Straßennetzwerke. Um diese komplexen Anforderungen zu erfüllen, ist es unerlässlich, große Datenmengen zu integrieren und zu verarbeiten, die aus diversen Quellen stammen. Dazu zählen beispielsweise Kartendaten, Wetterdaten, Realfahrtdaten oder unterschiedlichen Szenarien. Die Fähigkeit, diese umfangreichen und vielfältigen Informationen effektiv zu nutzen, ist entscheidend für die präzise Entwicklung und Bewertung der ODD.
- **Prozessintegration:** Die ODD dient als Modell, das als gemeinsame Grundlage für alle Stakeholder dient. Da zahlreiche Teams aus unterschiedlichen Bereichen – von der Geschäftsplanung bis hin zur technischen Entwicklung und dem Testen – an diesem Modell arbeiten, ist es unerlässlich, dass die verwendete Sprache kompatibel mit den verschiedenen Prozessen und Systemen ist. Die Sprache muss so gestaltet sein, dass sie nicht nur für technische Anwendungen, wie Entwicklung und Tests, sondern auch für Business-Case-Analysen geeignet ist. Dies stellt sicher, dass Informationen und Daten nahtlos zwischen verschiedenen Abteilungen und Phasen des Entwicklungsprozesses ausgetauscht werden können.

Nach aktuellem Kenntnisstand gibt es bisher keine Lösung, die alle beschriebenen Anforderungen zufriedenstellend adressiert. Dennoch gibt es in der Forschung bereits Ansätze oder Lösungen zu Teilaspekten, die für die weitere Ausarbeitung berücksichtigt werden sollten. Nachfolgend wird auf die Wichtigsten eingegangen.

4.2. Related Work

Dieses Kapitel zeigt auf, das bisher noch keine Lösungen existieren, die die abgeleiteten Anforderungsclustern ganzheitlich erfüllen. Dabei werden sowohl Forschungen in verwandten Bereichen wie der Szenarienmodellierung und Ontologie betrachtet als auch die Nutzung der ODD im Automobilbereich und in anderen Industrien näher beleuchtet. Zudem wird ausführlich auf ODD-

Taxonomien und die verschiedenen Möglichkeiten zur Beschreibung einer ODD eingegangen. Abschließend werden die repräsentativen Forschungen anhand einer einfachen Bewertungsmatrix entsprechend der Anforderungscluster (vgl. Kapitel 4.1) bewertet und damit die Forschungslücke aufgezeigt.

Szenarienbasiert

Das PEGASUS-Projekt hat einen szenariobasierten Test- und Entwicklungszyklus für die ODD implementiert, der wichtige Faktoren wie Verkehrsinfrastruktur, Umweltbedingungen und den Verkehrsfluss für Autobahn-Autonomiefahrten berücksichtigt. Diese Herangehensweise ermöglicht es, spezifische Szenarien zu identifizieren, die im Designprozess des ADS adressiert werden müssen. [Peg19] Ein signifikanter Vorteil dieses Ansatzes liegt in seinem ganzheitlichen Charakter, der speziell für den Einsatz auf Autobahnen entwickelt wurde. Dies gewährleistet, dass eine umfassende Palette an Faktoren, die die Sicherheit und Effizienz von autonomen Fahrten beeinflussen, berücksichtigt wird. Allerdings weist der Ansatz auch eine entscheidende Limitation auf: Es fehlt eine formale Spezifikation, die die Anforderungen der ODD in einer Weise erfüllt, die sowohl datengetriebene Ansätze als auch Sicherheitsaspekte integriert. Dies kann zu Herausforderungen führen, wenn es darum geht, präzise und verlässliche Richtlinien für die Entwicklung und Validierung von ADS in unterschiedlichen Fahrumgebungen und -szenarien zu erstellen.

Andere Forschungen nutzen ein Taxonomie-Framework für die szenarienbasierte Entwicklung und dem Testen von ADS. Mithilfe von Unified Modeling Language (UML)-Diagrammen wird ein Grundvokabular eingeführt, um Beziehungen zwischen Begriffen zu visualisieren. Der Ansatz betont die Notwendigkeit der Ausrichtung auf bestehende Standards, die Anpassung der Begriffe an szenarienbasierte Methoden und die Erweiterbarkeit des Frameworks für zukünftige Ergänzungen und Klarstellungen. [SMM21] Ein weiterer Ansatz betrachtet die Evaluierung der Performance autonomer Fahrzeuge in unterschiedlichen ODDs. Dazu wird ein Bayes'sches Hierarchiemodell genutzt, welches die ADS-Leistung über verschiedene ODDs hinweg zu bewertet. Der Ansatz optimiert das Szenarien-Sampling mittels submodularer Optimierung, mit dem Ziel, den Informationsgewinn zu maximieren und ein effizientes Stoppkriterium zu etablieren. [CBT21] Obwohl die Sprache des Papers teilweise schwer verständlich ist und keine klare Prozessintegration aufgezeigt wird, stellt die submodulare Optimierung einen signifikanten Mehrwert dar.

Weitere Forschungsvorhaben nutzen für die ODD für das Szenarien Sampling. Hierbei wird beschrieben, wie das Szenarien-Sampling zur Validierung und Quantifizierung der Sicherheit eines ADS innerhalb seiner definierten ODD genutzt werden kann. Dafür wird eine Invariantenmengen-Perspektive vorgeschlagen, um die Herausforderungen in der Sicherheitsbewertung über verschiedene Betriebsszenarien hinweg zu adressieren. [WRO+T21] Trotz der komplexen Sprache bietet das Framework einen formalen Ansatz zur Validierung und Quantifizierung von ODDs.

Ontologische Modelle

Die Verkehrsszenarienmodellierung nutzt aufgrund ihrer Ausdrucksstärke zunehmend ontologische Modelle. Diese ermöglichen es, kontextuelle Entitäten in einer formalen und expliziten Weise für Verkehrskomponenten zu konzeptualisieren. Hinsichtlich der Beschreibung von Umweltelementen ergeben sich damit Gemeinsamkeiten zur ODD, wodurch potenzielle Lösungen eventuell übertragbar sind.

Der Autor Regele [Reg08] nutzt Ontologien für ein hierarchisches Weltmodell, das den Entscheidungsprozess in autonomen Fahrsystemen unterstützt. Der Ansatz trennt die niedrigstufige Trajektorienplanung von der übergeordneten Verkehrskoordination, indem er ein abstraktes

Weltmodell auf Basis einer topologischen Fahrspursegmentierung verwendet. Dieses Modell repräsentiert semantische Beziehungen innerhalb von Verkehrsszenarien und vereinfacht so die Entwicklung und Bewertung automatisierter Fahrzeuge. Des Weiteren wird darauf eingegangen, wie eine auf Fahrzeug-zu-Fahrzeug-Kommunikation basierende Zusammenarbeit in komplexen Verkehrssituationen funktioniert und wie ein ontologiebasiertes Verkehrsmodell die Steuerungssysteme autonomer Fahrzeuge optimiert. [Reg08]

Ein weiterer Ansatz präsentiert ein modulares Framework zur Modellierung von Verkehrsszenarien für autonome Fahrzeuge mithilfe von Ontologien. Dafür wird eine semantische Darstellung von Verkehrsszenen entwickelt, die auch Verkehrsregeln einbezieht und so die situative Wahrnehmung und Entscheidungsfindung automatisierter Fahrzeuge verbessert. Diese ermöglicht eine modulare Software, die sich nahtlos an verschiedene nationale Verkehrsregeln anpassen kann und zwischen unterschiedlichen Regeln und Kontexten umschaltet. [BHR+17]

Ein ähnliches Vorgehen wird in einem anderen Ansatz genutzt, indem Verkehrsszenen mithilfe von Ontologien generiert werden. Es wird ein Schichtenmodell vorgestellt, welches die Straßeninfrastruktur, Verkehrselemente und Umweltbedingungen abbildet und so die automatisierte Erstellung von Verkehrsszenen in natürlicher Sprache erleichtert. Insgesamt soll dieses Vorgehen eine systematische und konsistente Erfassung komplexer Fahrsituationen für Tests und Simulationen ermöglichen. [BMM18]

ODD-Erstellung und Nutzung

Andere Forschungsarbeiten betrachten die Erstellung und Nutzung einer ODD. Der Autor Sun [SDC+21] diskutiert hierbei die Bedeutung der ODD bei der Festlegung der von funktionalen Grenzen für ADS. Darüber hinaus werden zentrale Herausforderungen in der Definition und Überwachung der ODD identifiziert, einschließlich der Handhabung ihrer Komplexität, der Überprüfung der ADS-Funktionen und des Echtzeit-Monitorings. Zur Bewältigung dieser Herausforderungen wird ein Rahmenwerk zur „ODD-Akklimatisierung“ vorgeschlagen, das einen zielorientierten Ansatz verfolgt. Dieser Ansatz nutzt etablierte Methoden wie HARA, HAZOP und FTA, um bekannte Risiken zu adressieren und die Leistung des ADS systematisch zu testen. [SDC+21] Obwohl der Ansatz über ein gutes ganzheitliches Konzept verfügt, ist die verwendete Beschreibungssprache sehr komplex.

Ein anderer Ansatz unterstreicht die Relevanz der ODD für die Gewährleistung der Sicherheit von ADS. Es wird betont, dass eine umfassende ODD notwendig ist, um „Edge Cases“ zu identifizieren und den sicheren Betrieb der AVs unter verschiedenen Bedingungen zu gewährleisten. Dabei liegt die Herausforderung in der Schaffung von Standards für ODDs, die sich an unterschiedliche Betriebsumgebungen anpassen und eine gewisse Flexibilität in ihrer Definition erlauben. [Ber19]

Lee [LNG20+] dagegen schlägt eine Methodik zur Definition der ODD eines ADS auf Basis statistischer Daten und Risikotoleranz vor. Der Ansatz beinhaltet die Sammlung geografischer Daten, die Bewertung der Umweltbedingungen und die Risikobewertung der ADS-Leistung, um Gebiete zu identifizieren, die für den autonomen Betrieb sicher sind. [LNG+20]

Die Forschung von Smart [SSJ21] thematisiert das Fehlen einer standardisierten Methode zur Definition und Bewertung der ODD in autonomen Fahrzeugen. Die Autoren stellen einen ganzheitlichen Sicherheitsbewertungsprozess vor, der Wissensgraphen verwendet, um Risiken innerhalb der ODD zu quantifizieren und zu evaluieren. Der Prozess umfasst die Planung und Implementierung, das Management und den Betrieb sowie die kontinuierliche Überwachung und Evaluierung. Durch den Einsatz von Wissensgraphen wird ein datengesteuerter Ansatz zur Identifizierung potenzieller Gefahren, zur Szenariengruppierung und zur Risikobewertung ermöglicht.

Dies kann nachfolgend für eine strukturierte und transparente Methode zur Sicherheitsbewertung für verschiedenen Einsatzszenarien genutzt werden. [SSJ21] Auch wenn der Ansatz eine technisch anspruchsvolle Lösung darstellt, bringt die modulare Nutzung von Wissensgraphen und die Integration in den Bewertungsprozess der ODD jedoch einen großen Vorteil mit sich.

Ito [ito21] beleuchtet für seine Forschung die Bedeutung der ODD bei der Gewährleistung der Sicherheit automatisierter Fahrsysteme. Dabei werden verschiedene Standards und Leitlinien, wie SAE J3016, BSI PAS 1883 und das AVSC-Framework, untersucht, um Methoden zur ODD-Beschreibung zu identifizieren. Obwohl aktuelle Methoden bereits strukturierte Kategorisierungen bieten, zeigt diese Veröffentlichung, dass Aspekte wie das Management von ODD-Verstößen, noch weiterentwickelt werden müssen. [Ito21]

Auch die Forschung von Gyllenhammer [GJW+20] konzentriert sich auf den Zusammenhang von ODD und Sicherheit. Die ODD wird hier genutzt, um den Geltungsbereich des Systems zu definieren und so den Umfang der Sicherheitsnachweise und die erforderliche Verifizierung einzuschränken. Das vorgestellte System zur Kategorisierung der Betriebsbedingungen eines Anwendungsfalls schlägt vor, die ODD entsprechend dieser Struktur zu formulieren, um eine effiziente Zuordnung zu potenziellen Anwendungsfällen zu ermöglichen. Die modulare Modellierung der ODD erleichtert sowohl die kontinuierliche Bereitstellung der ADS-Funktionen als auch den Prozess des Sicherheitsnachweises. [GJW+20] Obwohl dieser Ansatz umfangreich ist und vielfältige Anwendungsmöglichkeiten für das Safety Engineering aufzeigt, fehlt es an einer formalen ODD-Beschreibung. Auch datengetriebene Ansätze werden nicht berücksichtigt oder integriert

Das Paper „Fuzzy Interpretation of Operational Design Domains in Autonomous Driving“ schlägt einen unscharfen Ansatz zur Bewältigung wechselnder Betriebsbedingungen autonomer Fahrzeuge vor, indem die ODD in kleinere Fragmente, sogenannte µODDs, unterteilt wird. Ein Fuzzy Longitudinal Motion Controller (FLMC) passt das Fahrzeugverhalten an wechselnde Bedingungen an, wobei der Fokus auf Szenarien des Autobahnfahrens liegt. [SWT+22] Obwohl der Ansatz modular aufgebaut ist, indem er die ODD in Teile zerlegt, liegt der Fokus auf dem Test und Betrieb eines ADS.

Cho [Cho20] dagegen konzentriert sich auf die Entwicklung einer risikobasierten Methodik zur Definition der ODD für Fahrer-Automations-Integrationssysteme. Er identifiziert einen bedingten Hyperraum zur Darstellung der Bedingungen, die die Zuverlässigkeit des Systems beeinflussen und verwendet ein risikobasiertes Framework, um die Grenze zu bestimmen, innerhalb derer das System sicher operiert. Zudem werden das ODD-Management und die Herausforderungen in automatisierten Systemen. [Cho20]

ODD-Nutzung in anderen Domänen

Auch in anderen Industriebereichen wird das Konzept der ODD für die Definition von Betriebsbedingungen mittlerweile genutzt. So beschäftigt sich die Forschung von Meng [MTW21] mit der Beschreibung einer ODD für einen automatischen Hochgeschwindigkeitszug. Auf Basis dieser ODD wird ein Modell für das Szenario der Verknüpfungssteuerung von Zug- und Bahnsteigtüren entwickelt. Dieses Szenario wird mithilfe von UPPAAL modelliert, einem Tool, das besonders für solch komplexe technische Modelle geeignet ist. Aus dem entworfenen Szenario werden anschließend die Eingaben und die erwarteten Ergebnisse des ATO-Systems abgeleitet. Diese Inputs und Outputs dienen wiederum dazu, die funktionalen Anforderungen an das ATO-System zu definieren. [MTW21] UPPAAL bietet hierbei den Vorteil einer modularen Struktur, setzt jedoch ein spezifisches technisches Fachwissen voraus.

Auch Tonk [TBB+21] wendet das Konzept der ODD auf die Zugentwicklung an. Hierbei wird das ODD-Konzept auf den Bereich der ferngesteuerten Zugführung übertragen. Dabei wird der Fokus daraufgelegt, die ODD durch kontinuierliche Risikobewertung und -reduktion einzuschränken und zu präzisieren, um die Sicherheit des Fernbetriebs von Zügen zu verbessern. [TBB+21]

Eine weitere Forschungsarbeit widmet sich der Definition einer ODD für Schiffsnavigationssysteme, wobei sowohl regulatorische als auch technische Beschränkungen berücksichtigt werden. Zudem wird untersucht, wie die Autonomie eines Schiffs durch ODD und Navigationsspezifikationen genauer definiert werden kann. [UH21]

ODD-Taxonomien

Neben diesen Ansätzen untersuchen viele Forscher die Modellierung und Analyse von Fahrszenarien und Anwendungsfällen für ADS, um entsprechende ODDs zu formalisieren. Czarnecki [Cza18] definiert beispielsweise eine Taxonomie für ODDs autonomer Fahrzeuge, die von verschiedenen Faktoren wie dem Fahrzeugmodell, dem Betriebsumweltmodell der Straße und dem Betriebsweltmodell abhängen. [Cza18] Ein großer Vorteil dieser Taxonomien liegt in der guten Strukturierung des Anwendungsraums. Allerdings bieten diese neben der Strukturierung der Problemdomäne keine weiteren Lösungen, die das Ganze in Form einer Sprache für die Modellierung einer ODD nutzbar machen.

ODD-Beschreibungssprache

Die nachfolgenden Veröffentlichungen beinhalten dagegen Lösungen, mit dessen Hilfe eine ODD modelliert werden kann. So schlägt das Paper „A Knowledge-based Approach for the Automatic Construction of Skill Graphs for Online Monitoring“ eine ontologiebasierte Methode zur Erstellung von Skill-Graphen vor. Diese Graphen repräsentieren die Fähigkeiten, die für automatisierte Fahrzeugverhaltensweisen in bestimmten ODDs erforderlich sind. Der Ansatz zielt darauf ab, die Herausforderung der manuellen Erstellung von Skill-Graphen zu bewältigen, indem der Prozess mithilfe von Expertenwissen automatisiert wird. Die Methode nutzt eine Wissensbasis, um Basis-Skill-Graphen für unterschiedliche Fahrzeugverhaltensweisen zu definieren und passt diese automatisch an die Elemente in der ODD des Fahrzeugs an. Dadurch sollen Inkonsistenzen und Fehler in den Skill-Graphen reduziert und Anpassungen vereinfacht werden, sofern sich die ODD weiterentwickelt. [JMB+21]

Shakeri [Sha24] nutzt einen formalen Ansatz zur Definition der OD und der ODD. Er geht auf Unklarheiten in den aktuellen Definitionen ein und stellt passende mathematische Modelle bereit, die eindeutige Beziehungen zwischen OD und ODD herstellen. Die Autoren argumentieren, dass ein Mangel an klaren Definitionen zu Missverständnissen, erhöhten Sicherheitsrisiken und Verzögerungen in Entwicklungsprozessen führen kann. Durch die Einführung formaler Darstellungen soll die Entwicklung von ODD-Spezifikationen erleichtert und ein einheitliches Verständnis unter verschiedenen Beteiligten sichergestellt werden. [Sha24] Das Paper verwendet dabei eine formale Sprache, um ODD und OD zu definieren. Diese Herangehensweise bietet zwar technische Präzision, ist aber für nicht-technische Stakeholder nicht nachzuvollziehen. Zudem wird ein sehr starker Fokus auf formale Spezifikationsmethoden gelegt, was zu einem detaillierten, aber komplexen Inhalt führt. [Sha24]

Myers [MS20] diskutiert in der Veröffentlichung „Design Considerations for ODD Ontology“ die Notwendigkeit einer standardisierten Sprache zur Definition von ODDs für vernetzte und autonome Fahrzeuge (CAVs). Dafür identifiziert er die Anforderungen an eine solche Sprache, wobei der Schwerpunkt auf maschineller Lesbarkeit, eindeutigen Definitionen sowie der Einbeziehung geografischer und umweltbezogener Variablen liegt. Zudem beschreibt Myers [MS20] die

Herausforderungen, die bei der Darstellung komplexer Bedingungen und der Verwaltung von Änderungen in ODD-Definitionen auftreten können. Als Gegenmaßnahme wird die Verwendung von hierarchischen Strukturen und Versionskontrollen vorgeschlagen, um Klarheit und Anpassungsfähigkeit zu gewährleisten. [MS20] Der Ansatz unterstützt zwar die Verwendung hierarchischer Strukturen, was die Modularität und Prozessintegration begünstigt, bietet aber keine leicht verständliche Sprache. Zudem fehlt ein Konzept zur quantitativen und qualitativen Bewertung der ODD.

Im Rahmen der Standardisierung hat ASAM im Projekt OpenODD ein Konzept erarbeitet, das sich mit der Entwicklung eines einheitlichen Formats zur Beschreibung der ODD für ADS befasst. Der implementierte zweistufige Abstraktionsansatz besteht aus einer strukturierten Darstellung in natürlicher Sprache und einer formalen Repräsentation. Der Ansatz bietet die Möglichkeit, komplexe ODD-Bedingungen mit Hilfe einer definierten Domänenontologie zu beschreiben und so eine umfassende Semantik zu erreichen. Dabei hat der Ansatz das Ziel, ODD-bezogene Aktivitäten während des gesamten Entwicklungszyklus von ADS zu unterstützen, einschließlich Spezifikation sowie Verifikation und Validierung, und dabei eine vielfältige Gruppe von Stakeholdern abzudecken. [IZK21+, Sch21b]

Ein bedeutender Vorteil dieses Ansatzes ist die Bereitstellung einer formalen Sprache, die als Grundlage für die Beschreibung der ODD dient und sich auf alle Stakeholder konzentriert. Allerdings gibt es auch einige Nachteile. Zum einen beinhaltet der Ansatz zwei verschiedene Sprachen, die durch einen Konverter verbunden sind. Zum anderen fehlt ein Methode, die diesen Ansatz für die Entwicklung nutzbar macht und die Integration in den Entwicklungsprozess aufzeigt. Auch die Nutzbarkeit der Sprache in der Praxis wird nicht evaluiert, was Fragen zur praktischen Anwendbarkeit und Modularität offenlässt.

Zusammenfassende Betrachtung

Trotz der beschriebenen Fortschritte befindet sich die Forschung zur Spezifikation und Nutzung einer ODD im Entwicklungskontext noch in einem frühen Stadium. Viele Studien konzentrieren sich darauf, verschiedene Aspekte zu kategorisieren, die die Leistung eines ADS beeinträchtigen könnten, und bemühen sich, die ODD entsprechend zu konstruieren. Andere Ansätze schaffen es, eine formale Sprache vorzuschlagen und die Anwendbarkeit für einzelne Teilbereiche der Entwicklung zu demonstrieren. Es wird jedoch deutlich, dass bisher keine Lösung existiert, die alle oben genannten Anforderungen ganzheitlich und integriert erfüllt. Übersichtlich kann dies anhand der Bewertung der Anforderungscluster aufgezeigt werden (vgl.

Tabelle 4-2). Die Bewertung wird dabei anhand einer einfachen Bewertungsmatrix vorgenommen (0= Anforderung nicht erfüllt, 1 = Anforderung teilweise erfüllt, Anpassung notwendig, 2 = Anforderung vollständig erfüllt).

Tabelle 4-2: Bewertung des Standes der Wissenschaft anhand der Anforderungscluster

	Verständlichkeit und Präzision	Informations-bedarfe	Modularität	Quantitative und qualitative Bewertung	Prozess-integration
[MTW]	0	1	2	1	1
[SWT+22]	0	1	2	1	1
[CBT21]	1	1	2	2	1

[WRO+21]	1	1	2	2	1
[Ber19]	0	1	0	0	1
[LNG+20]	0	1	1	2	1
[UH21]	1	1	0	0	1
[SDC+21]	0	2	1	2	2
[TBB+21]	0	0	0	1	1
[IZK+21]	2	2	0	0	1
[SIZ+21]	2	2	0	1	1
[SSJ21]	1	1	2	2	2
[Ito21]	1	2	0	1	1
[GJW+20]	1	1	2	2	2
[Reg08]	1	2	2	0	1
[SSM21]	1	2	2	1	2
[JMB+21]	1	2	2	1	1
[Cho20]	0	0	1	1	1
[BMM18]	1	1	1	0	0
[BHR+17	0	1	1	0	1
[Sha20]	1	2	2	2	2
[MS20]	1	2	2	1	1

4.3. Forschungsfragen

Aus dem übergeordneten Problem und dem Stand der Wissenschaft ergeben sich die Forschungsfragen, die im Rahmen dieser Arbeit untersucht und beantwortet werden sollen:

- Forschungsfrage 1: Wie kann eine Sprache entwickelt werden, die eine präzise Beschreibung und Repräsentation von unterschiedlichen Betriebsbedingungen eines ADS ermöglicht, indem sie Betriebsbedingungen formal spezifiziert und diese modular organisiert? Wie können zusätzlich Modellierungspatterns abgeleitet werden, die die Anwendbarkeit der Sprache vereinfachen?
- Forschungsfrage 2: Wie lässt sich eine toolgestützte Umsetzung des Lösungskonzeptes gestalten, dass nicht nur den Abgleich des vorgesehenen Betriebsbereiches des ADS mit der ODD unterstützt, sondern auch relevante Daten für die weitere Entwicklung des Systems liefert sowie einen skalierbaren Ansatz bietet?
- Forschungsfrage 3: Wie können die Nutzbarkeit und Effektivität der entwickelten Lösungen zur präzisen Beschreibung und Strukturierung von Betriebsbedingungen eines ADS sowie deren

Entwurf und Nutzbarkeit im Entwicklungsprozess durch ausgewählte Anwendungsfälle demonstriert werden?

4.4. Lösungskonzept

Aus der beschriebenen Problematik, dem Stand der Forschung und den abgeleiteten Forschungsfragen, leitet sich ein Lösungskonzept ab. Zur Erfüllung der beschriebenen Anforderungen, besteht dies aus drei Teilbereichen, die bei der Definition des Lösungskonzeptes gleichermaßen erfüllt sein müssen (siehe Abbildung 4.2).

Model-Driven Engineering: Die Notwendigkeit einer klaren und präzisen Sprache für die Beschreibung der ODD wird durch das übergeordnete Problem unterstrichen, dass die Komplexität und Vielfalt der Umgebungsbedingungen für autonome Fahrsysteme hervorhebt. Die Forschungsfragen fordern die Entwicklung einer Sprache, die Betriebsbedingungen formal in Form von Regeln spezifiziert und diese Regeln modular organisiert. Model-Driven Engineering bietet einen Rahmen für die Entwicklung eines formalen Modells, das als gemeinsame Grundlage für alle Stakeholder dient. Es ermöglicht die Schaffung einer konsistenten und transparenten Darstellung der ODD, die sowohl die Anforderungen an Verständlichkeit als auch technische Präzision erfüllt. [KCG+20]

Big Data Engineering: Die Bedeutung der Integration und Nutzung umfangreicher Datenmengen wird durch das übergeordnete Problem betont, dass die Notwendigkeit einer realitätsnahen Abbildung der ODD unterstreicht. Die Forschungsfragen fordern eine Methode, die relevante Inputs für die Entwicklung liefert und eine gemeinsame Wissensbasis für alle relevanten Stakeholder schafft. Big Data Engineering ermöglicht dies durch die automatisierte Verarbeitung großer Datenmengen, die Informationen über das Wetter, die Infrastruktur und den Verkehr beinhalten. Die Datenmenge kann dabei schnell Terabytegröße je Stadt erreichen. Diese Informationen werden anschließend mit den Spezifikationen der ODD abgeglichen, um beispielsweise die besagten Straßennetze für das Servicegebiet zu identifizieren. Eine effektive Integration und Nutzung dieser Daten für den Abgleich und die Entwicklung einer ODD sind wesentlicher Teil der Lösung. [ZKH20]

Safety Engineering: Die quantitative und qualitative Bewertung der ODD ist ein entscheidender Schritt im Entwicklungsprozess automatisierter Fahrsysteme und dient nicht nur der Bewertung der Abdeckung der ODD, sondern auch der Risikoabschätzung. Die Gesellschaft und vorgeschlagene Regelungen verlangen, dass autonome Fahrer mindestens so sicher sind wie menschliche Fahrer. Eine solche Erwartung zu erfüllen, erfordert die Nutzung von quantitativ datengetriebenen Analysen im Verifikations- und Validierungsprozess. Im Gegensatz zum traditionellen Safety Engineering, das sich auf Fehlermodi konzentriert, die mit Sicherheitszielen ohne Berücksichtigung von Betriebsdaten verbunden sind, erfordert die Sicherheit des autonomen Fahrens das Sammeln und Analysieren von Beweisen aus Daten (z. B. Realfahrdaten oder Kartendaten). Damit kann das Verständnis für spezifisches Systemverhalten verbessert oder der Fokus der Absicherung präzisiert werden. Eine solche Analyse muss quantitative Risikoabschätzungen unterstützen, um den Vergleich mit der Sicherheitsbilanz menschlicher Fahrer zu erleichtern. Die Basis hierfür bildet eine formal beschriebene ODD, die als Anforderung eine solche Analyse unterstützen können muss. [KW24]

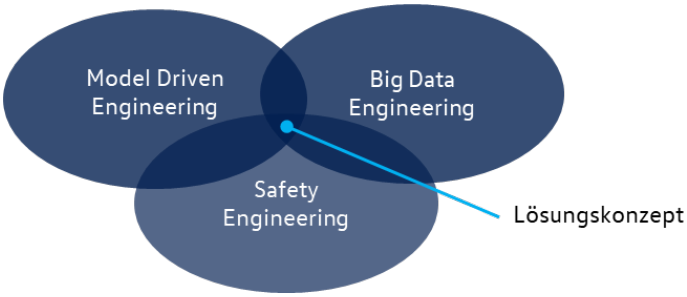

Abbildung 4.2: ODD-Spezifikation in der Schnittmenge verschiedener Teilbereiche

Aus den beschriebenen Teilbereichen lässt sich schließen, was die Lösung berücksichtigen muss. Das Ziel ist es, eine Lösung zu entwickeln, die auf Basis der Sprache komplexe Datensätze effektiv verwaltet und interpretiert, um die Entwicklung von ADS voranzutreiben. Dies wird durch den Einsatz einer spezifischen Taxonomie und einer modular aufgebauten ODD erreicht (vgl. Abbildung 4.3). Diese Struktur hilft dabei, eine Vielzahl von Fahrsituationen zu differenzieren und zu kategorisieren, was die Überprüfung und Validierung der Systemleistung wesentlich verbessert. Der nachfolgende Abschnitt bietet einen detaillierten Einblick in die einzelnen Komponenten der Lösung.

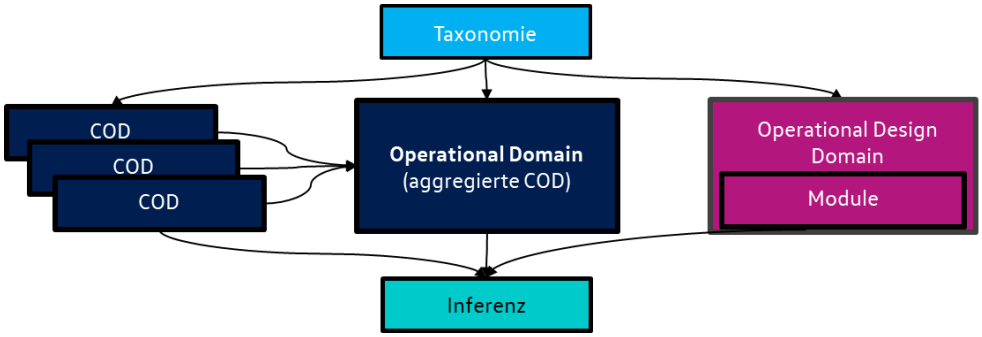

Abbildung 4.3: Darstellung des Lösungskonzeptes

Taxonomie – Die Taxonomie stellt eine Hierarchie von Konzepten dar, die dazu dient, Variablen und Werte zu definieren, welche unterschiedliche Umweltbedingungen repräsentieren. Diese bildet die Grundlage für die Beschreibung von Situationsdaten und ODD-Modulen und ist entscheidend für eine präzise Analyse und den Vergleich verschiedener Umweltbedingungen. Ein Beispiel hierfür ist die Taxonomie von Straßeninfrastruktur (vgl. Quellcode 4-1). Hierarchisch geordnet, sind die Konzepte für „straßentyp“, "fahrstreifen_anzahl“ und „standstreifen“ als untergeordnete Konzepte des übergeordneten Konzepts angelegt. Die spezifischen Kategorien und Datentypen innerhalb dieser Hierarchie erlauben die genaue Definition von Merkmalen wie Autobahn und Hauptstraße.

```
TAXONOMIE:
  straßeninfrastruktur:
    straßentyp:
      - autobahn:
      - hauptstraße
    fahrstreifen_anzahl: integer
    standstreifen: boolean
```

Quellcode 4-1: Beispieltaxonomie - taxonomie_straßeninfrastruktur.yml

Zusätzlich zu den grundlegenden Umweltbedingungen, können weitere Taxonomiekonzepte entwickelt werden, um bspw. Messgrößen zu definieren. Beispielsweise kann dem Konzept

„Straßentyp“ für die Kategorie „Autobahn“ eine Maßeinheit wie Länge zugeordnet werden. Diese gibt an, wie viele Kilometer Autobahn in den Daten (COD) vorhanden oder zulässig sind (ODD).

Current Operational Domain – Eine COD beschreibt relevante Umweltbedingungen zu einem bestimmten Zeitpunkt an einem bestimmten Ort. Damit stellt diese eine Teilmenge der Umweltbedingungen an einem spezifischen Ort und zu einem spezifischen Zeitpunkt dar. Die Definition einer COD umfasst zwei Hauptkomponenten:

- Taxonomie: eine gemeinsam genutzte Taxonomie mit der Modulspezifikation, um eine Vergleichbarkeit herzustellen
- Tabellarische Datensammlung: eine Sammlung von Schlüssel-Wert-Paaren, die den Inhalt der COD beschreiben. CODs können auch tabellarisch repräsentiert werden.

Eine COD könnte beispielsweise eine Autobahn mit drei regulären Fahrstreifen und einem Standstreifen beschreiben (vgl. Quellcode 4-2).

```
s00102:
  - TEMPORAL_EXTENT: "2023-06-01 08:12:53.784"
    SPATIAL_EXTENT: "45.024 10.261"
    straßentyp: autobahn
    fahrstreifen_anzahl: 3
    strandstreifen: true
```

Quellcode 4-2: Beispiel COD s00102

Operational Domain (OD) – Eine Operational Domain (OD) beschreibt ein Gebiet über einen bestimmten Zeitraum hinweg und stellt die Aggregation verschiedener CODs dar. Beispielsweise könnte ein OD ein gesamtes Stadtgebiet abdecken. Die OD kann auch tabellarisch abgebildet werden. Im Folgenden ist ein Beispiel für ein OD dargestellt, das zeitliche und räumliche Ausdehnungen sowie spezifische Wetterbedingungen für einen bestimmten Zeitraum in einer Stadt abdeckt (vgl. Quellcode 4-3).

```
OD:
  - TEMPORAL_EXTENT:      ["2023-06-01 08:12:53.784","2023-06-02 18:34:04.262"]
    SPATIAL_EXTENT:       "45.024 10.261"
    REGENMENGE;mm/hr:     [0.000 .. 6.214]
    REGENMENGE;enum:      [kein_regen, leichter_regen, moderater_regen]
```

Quellcode 4-3: Beispiel einer OD

Operational Design Domain (ODD) – Die Spezifikation einer ODD setzt sich aus drei Hauptkomponenten zusammen: Taxonomie, Module und ODD.

- Taxonomie - Die gemeinsam genutzte Taxonomie zur präzisen Definition und Kategorisierung von CODs, die für das ADS relevant sind.
- Module - Die Module sind wiederverwendbare Beschreibungen, die dazu dienen, Situationen zu spezifizieren, die entweder zulässig oder unzulässig sind. Aufgrund der großen Anzahl von Situationen werden Bedingungen verwendet, die mittels propositionaler Logik spezifiziert sind, um die Listen der zulässigen oder unzulässigen Situationen darzustellen. Module sind in verschiedene Typen gegliedert (Use-Case Modul, domänenspezifische Module und Systemgrenzmodule), um verschiedene Anwendungszwecke zu ermöglichen.
- ODD – Die ODD bündelt die Module zu einer einzigen kohärenten Spezifikation.

Als Beispiel kann ein Modul (vgl. Quellcode 4-4) genutzt werden, das auf der Taxonomie „taxonomie_straßeninfrastruktur.yml“ (vgl. Quellcode 4-1) basiert und welches die Liste aller CODs umfasst, die die folgenden Bedingungen erfüllen:

- Der Straßentyp („straßentyp“) ist Autobahn („autobahn“) oder Hauptstraße („hauptstraße“).
- Die Anzahl der Fahrstreifen („fahrstreifen_anzahl“) ist größer als zwei.
- Es gibt einen Standstreifen („standstreifen“).

Die COD s00102 (vgl. Quellcode 4-2) erfüllt alle diese Bedingungen und ist somit in der Liste der von Modul inkludierten Situationen enthalten.

```
IMPORT:
  - taxonomie_straßeninfrastruktur.yml

MODULES:
  modul_1:
    TYPE: dsm
    INCLUDE_AND:
      straßentyp:
        - autobahn
        - hauptstraße
      fahrstreifen_anzahl: >2
      standstreifen: true
```

Quellcode 4-4: Modul auf Basis von taxonomie_straßeninfrastruktur.yml

Diese strukturierte Vorgehensweise ermöglicht es, komplexe und vielfältige CODs systematisch zu erfassen und zu verwalten und die Betriebsbedingungen präzise zu beschreiben.

Inferenzmaschine - Die Inferenzmaschine nutzt die mit Hilfe der Taxonomie beschriebenen ODD-Module, um zu entscheiden, welche CODs innerhalb der festgelegten Grenzen des Systems liegen. Diese Fähigkeit ist essenziell, um sicherzustellen, dass das System reale Bedingungen effektiv und sicher bewältigen kann.

Je nach Herkunft der Daten liefert diese Analyse wichtige Einblicke in die operativen Grenzen und Fähigkeiten des ADS. Diese Analysen sind entscheidend, um die Konformität des Systems mit seinem vorgesehenen Einsatzbereich zu überprüfen und zu garantieren, dass das Fahrzeug unter verschiedenen Bedingungen wie erwartet funktioniert. Die Vielfalt der COD und die Notwendigkeit, umfangreiche Datensätze zu verarbeiten, erfordern eine hoch skalierbare und flexible Lösung. Unterschiedliche Anwendungsfälle, wie Autobahnfahrten oder das Manövrieren auf Parkplätzen, verlangen spezifische Daten, die wiederum einzigartige Anforderungen an die Relevanzprüfung stellen. Der hier entwickelte Ansatz ermöglicht eine Lösung, die alle diese Anforderungen erfüllt.

Die nachfolgende Beschreibung gliedert sich entsprechend der Forschungsfragen (vgl. Abbildung 4.4). Die Grundlage hierfür bildet eine formale Sprache, die in Kapitel 5 näher erläutert wird. Darauf aufbauend, werden Modellierungspatterns in Kapitel 6 abgeleitet, die die Anwendbarkeit der Sprache erleichtern. Kapitel 7 implementiert die entwickelte Lösung anhand zweier unterschiedlicher Werkzeuge und Kapitel 8 demonstriert den Nutzen anhand drei verschiedener Anwendungsfälle.

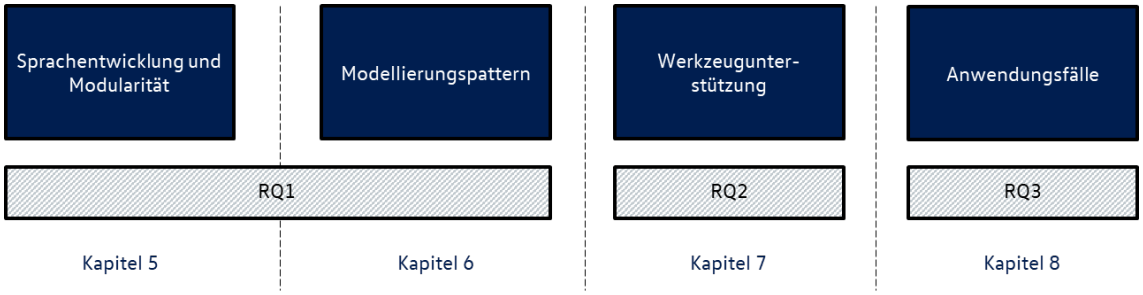

Abbildung 4.4: Verknüpfung von Lösung, Kapitelstruktur und Forschungsfragen

5. Sprachentwicklung

In diesem Kapitel wird die Sprache zur Definition von Taxonomie, COD, Modulen und ODD auf Basis des zugrunde liegenden Lösungskonzepts eingeführt. Die Sprache zielt darauf ab, eine leicht verständliche, aber technisch präzise Beschreibung komplexer Betriebsbedingungen zu ermöglichen. Dies wird durch die Verwendung von YAML als grundlegendes Datenformat unterstützt. Dieses bietet eine klare und einfache Syntax, während die darauf aufbauende Sprache gleichzeitig die Flexibilität zur Darstellung diverser Datentypen und Betriebsbedingungen gewährleistet. Dafür erläutert Kapitel 5.1 zunächst die grundlegende Syntax von YAML, um die Basis für die sprachlichen Definitionen zu schaffen. Darauf aufbauend werden in Kapitel 5.2 die relevanten Basiselemente der Sprache definiert, die über die gesamte Struktur hinweg Anwendung finden.

Die weitere Definition der Sprache fokussiert auf Taxonomie, COD, Module und ODD. Ein zentraler Bestandteil der Lösung ist hierbei die Nutzung einer austauschbaren und erweiterbaren Taxonomie, die die Grundlage für die Definitionen der ODD bildet und eine effiziente Repräsentation verschiedener Informationsbedarfe ermöglicht. Kapitel 5.3 beschreibt die Syntax und Semantik dieser Taxonomie im Detail. Im Anschluss wird in Kapitel 5.4 die Struktur einer COD definiert, welche als Sammlung von Inputdaten die Grundlage für den Abgleich mit der ODD entsprechend des Lösungskonzeptes bildet.

Abschließend widmet sich Kapitel 5.5 der Syntax und Semantik der ODD und ihrer Module. Diese Module sind ein wesentlicher Bestandteil des Lösungskonzepts, da sie die flexible und modulare Definition von ODDs ermöglichen, indem sie auf kleinere Einheiten referenzieren und diese zu größeren Strukturen zusammenfügen. Die Modularität der Lösung wird insbesondere durch die Einführung unterschiedlicher Modultypen erreicht, was im Detail in Kapitel 5.5.2 beschrieben wird.

Die kombinierte Struktur von Taxonomie, COD und Modulen gewährleistet eine konsistente und skalierbare Lösung zur präzisen Definition und Verwaltung der Betriebsbedingungen eines ADS.

5.1. YAML als grundlegende Syntax

Bei der Auswahl einer grundlegenden Syntax für die Modellierungssprache wird YAML (YAML Ain't Markup Language) als besonders geeignetes Datenformat erachtet [Yam21]. Die Verwendung der Syntax von YAML bietet mehrere Vorteile:

- Lesbarkeit: YAML ist für seine hohe Lesbarkeit bekannt und wurde so entworfen, dass es auch für Menschen einfach zu verstehen ist. Dies erleichtert die Zusammenarbeit und das Review von spezifizierten ODDs zwischen verschiedenen und innerhalb von Teams.
- Einfachheit: YAML hat eine einfache Syntax, die es erleichtert, komplexe Datenstrukturen wie Listen, Hashes (Dictionaries) und Skalare ohne umständliche Markierungen oder Klammern darzustellen.
- Flexibilität: YAML unterstützt die Darstellung von hierarchischen Datenstrukturen, was besonders nützlich ist, um vielschichtige Zusammenhänge innerhalb einer ODD zu modellieren.
- Interoperabilität: YAML-Dokumente können leicht in Datenstrukturen von Programmiersprachen wie Python und JavaScript umgewandelt werden, was die Integration in bestehende Tools und Prozesse erleichtert.

Daneben gibt es jedoch auch mehrere Limitationen, die vor der weiteren Verwendung beachtet werden müssen. YAML zeichnet sich durch eine übersichtliche Syntax aus, die das Darstellen von Datenstrukturen wie Skalaren (z. B. Strings, Zahlen), Listen und assoziativen Arrays (Dictionaries) in einer für Menschen leicht lesbaren Form ermöglicht. Als semi-formale Sprache hat YAML jedoch

Grenzen, da es weder eine strenge Typisierung noch eine formale Semantik mit präzisen Verhaltensmodellen bietet. Dies resultiert in spezifischen Limitationen:

- fehlende formale Semantik: Eine präzise, mathematisch fundierte Bedeutung von Sprachelementen und Konstrukten fehlt, was insbesondere bei der Typisierung und Typkonversionen relevant wäre. Da für die Modellierung der ODD aber eine eigene Semantik definiert werden soll, ist dies nicht weiter von Bedeutung.
- strenge Typisierung: Die explizite Deklaration von Variablentypen und die Notwendigkeit bewusster Typkonversionen sind nicht vorgesehen, was die Fehleranfälligkeit durch unbeabsichtigte Typkonversionen erhöhen kann.
- Fehleranfälligkeit der Syntax: Die Einrückungssyntax, obwohl für die Lesbarkeit bekannt, kann insbesondere in umfangreichen oder komplexen Dokumenten zu Fehlern führen.
- Mangel an Formalität: Direkte Mechanismen zur Inhaltsvalidierung gegen ein Schema fehlen.

Trotz dieser Einschränkungen bleibt YAML eine geeignete Wahl, wenn leicht verständliche Inhalte mit einem gewissen Maß an technischer Präzision gefordert sind. Für komplexere Validierungen oder eine striktere Formalisierung kann es dennoch ratsam sein, YAML mit anderen Werkzeugen oder Sprachen zu kombinieren, die robustere Validierungs- und Spezifikationsfähigkeiten bieten (vgl. Kapitel 0).

Neben YAML gibt es mehrere Alternativen, die sich für die beschriebenen Anforderungen (vgl. Kapitel 3), gerade in Bezug auf Einfachheit und leichte Verständlichkeit, jedoch nur bedingt eignen. Die bekanntesten sind:

- **JSON** (JavaScript Object Notation): JSON ist eine textbasierte Datenformatierungssprache, die für ihre Kompaktheit und Geschwindigkeit in der Verarbeitung bekannt ist. Sie bietet eine strengere Struktur als YAML, was dazu führt, dass komplexe Datenstrukturen weniger lesbar werden. Vorteilhaft ist jedoch, dass viele Programmiersprachen eine gute Unterstützung für JSON bereitstellen. [ECM20]
- **XML** (eXtensible Markup Language): Eine Markup-Sprache, die eine sehr detaillierte Datenmodellierung ermöglicht, einschließlich Attributen und verschachtelten Elementen. XML ist mächtig, aber seine Verbosität kann die Lesbarkeit beeinträchtigen. [W3c08]
- **Protobuf** (Protocol Buffers): Protobuf wurde von Google entwickelt und ist ein Mechanismus zur Serialisierung strukturierter Daten, der besonders für die Anwendung in Kommunikationsprotokollen und Datenlagersystemen geeignet ist. Protobuf ist effizient und schnell, erfordert jedoch im Voraus die Definition von Schemata. [Goo23]
- **ASN.1** (Abstract Syntax Notation One): Ein Standard, der eine formale Notation zur Beschreibung von Datenstrukturen bietet. Es wird häufig in der Telekommunikation und Informatik eingesetzt, erfordert aber spezielle Tools zur Handhabung. [Itu21]

Nachdem im vorherigen Abschnitt ein grundlegender Überblick über YAML sowie dessen Stärken und Schwächen gegeben wurde, widmet sich dieser Abschnitt nun einer detaillierteren Betrachtung der Syntax. Anhand konkreter Beispiele wird beschrieben, wie verschiedene Datenstrukturen repräsentiert und in YAML-Dokumenten effektiv genutzt werden können. Dabei wird nur auf die Details eingegangen, die für die nachfolgende Entwicklung und Nutzung der Sprache eine Relevanz besitzen (vollständige Dokumentation der Funktionalitäten von YAML unter [Yam21]).

Struktur durch Einrücken – Maßgeblich für die Syntax in YAML ist, dass die Struktur durch Einrückungen vorgegeben wird. Dies macht die Struktur auf den ersten Blick klar und leicht verständlich (vgl. Quellcode 5-1). [Yam21]

```yaml
person:
   name: Max Mustermann
        alter: 30:
```

Quellcode 5-1:– Struktur durch Einrücken in YAML

Schlüssel-Wert-Paare: YAML nutzt zur Definition sogenannte Schlüssel-Wert-Paare, die eine einzelne Dateneinheit unter einem einzigen Schlüssel abspeichern. Schlüssel und Wert werden dabei durch einen Doppelpunkt getrennt (vgl. Quellcode 5-2). [Yam21]

```yaml
farbe: blau
größe: groß
```

Quellcode 5-2: Schlüssel-Wert-Paare in YAML

Listen und Maps: YAML unterstützt Listen (Arrays) und Maps (assoziative Arrays oder Dictionaries). Listen werden durch Bindestriche angezeigt, während Maps durch Schlüssel-Wert-Paare dargestellt werden (vgl. Quellcode 5-3 und 5-4). [Yam21]

```yaml
hobbies:
   - Lesen
   - Wandern
   - Fotografieren
```

Quellcode 5-3: Darstellung von Listen in YAML

```yaml
adresse:
   straße: Hauptstraße 1
   stadt: Beispielstadt
   plz: 12345
```

Quellcode 5-4: Darstellung von Maps in YAML

Skalare Typen: YAML unterstützt skalare Typen wie Zeichenketten, Booleans (true/false), Ganzzahlen und Fließkommazahlen. Zeichenketten müssen nicht immer in Anführungszeichen gesetzt werden. Eine Ausnahme bilden lediglich Zeichen, die in YAML eine besondere Bedeutung haben (vgl. Quellcode 5-5). [Yam21]

```yaml
name: "Max Mustermann "
verifiziert: true
alter: 30
gewicht: 70.5
```

Quellcode 5-5: Beschreibung von skalaren Typen in YAML

Mehrzeilige Zeichenketten: YAML bietet mehrere Möglichkeiten, mehrzeilige Zeichenketten darzustellen, was besonders nützlich für die Einbindung von Dokumentationen oder großen Konfigurationswerten ist (vgl. Quellcode 5-6). [Yam21]

```yaml
beschreibung: >
   Dies ist eine sehr lange Beschreibung,
   die über mehrere Zeilen geht, aber in der Ausgabe
   in einer einzigen Zeile erscheinen wird.
notiz: |
   Diese Notiz enthält mehrere Zeilen.
   Jede Zeile wird genau so dargestellt,
   wie sie hier eingegeben wurde.
```

Quellcode 5-6: Mehrzeilige Zeichenketten in YAML

Kommentare: Zeilen, die mit einem # beginnen, werden als Kommentare behandelt und von YAML-Parsern ignoriert. Dies ist hilfreich für Anmerkungen oder Erklärungen innerhalb der YAML-Dateien (vgl. Quellcode 5-7). [Yam21]

```
person:
   name: John Doe # Kommentar neben einem Schlüssel-Wert-Paar
   alter: 30 # Das Alter der Person
```

Quellcode 5-7: Kommentare in YAML

Diese Beispiele beschreiben grundlegende Aspekte der YAML-Syntax und zeigen, wie Daten in YAML-Dateien organisiert werden können. Durch die Verwendung dieser Syntaxelemente können komplexe Datenstrukturen auf eine Weise definiert werden, die sowohl für Menschen als auch für Maschinen gut lesbar und interpretierbar ist. Nachfolgend wird diese Syntax als Grundlage für aufbauende Definitionen genutzt.

5.2. Formale Definition grundlegender Strukturelemente

Für die weitere Definition der Sprache werden grundlegende Strukturelemente formal definiert (vgl. Quellcode 5-8). Die Notation der Syntax dieser kontextfreien Sprache erfolgt in EBNF (vgl. Kapitel 2.3).

Aufbau der YAML Struktur Das vorherige Kapitel 5.1 hat YAML als Basis für die grundlegende Strukturierung der Sprache definiert. Daraus ergeben sich Anforderungen an die Syntax, die die Strukturierung und Eigenheiten von YAML berücksichtigen müssen. Zeilenumbrüche sind als Unix-typisches \n oder Windows-kompatibles \r\n festgelegt. Einrückungen erfolgen über Tabulatoren (\t), da YAML hierarchische Strukturen durch Einrückungen darstellt. Die EBNF kann solche kontextabhängigen Strukturen nicht direkt abbilden, weshalb zusätzliche Prüfungen außerhalb des hier definierten Syntaxmodells erforderlich sind.

Zeichen, Zahlen und Operatoren In diesem Segment wird die Konstruktion von Zeichenketten (<string>), die aus Buchstaben, Ziffern und Unterstrichen bestehen können, behandelt. Buchstaben sind auf das lateinische Alphabet beschränkt, Ziffern auf 0–9. Relationale Operatoren umfassen <, >, <=, >= sowie das Minuszeichen -.

Datentypen Dieser Abschnitt erläutert die Definition verschiedener Datentypen. Dazu zählen Ganzzahlen und Fließkommazahlen, die optional ein vorangestelltes Minuszeichen für negative Werte enthalten können. Boolesche Werte sind auf true und false beschränkt. Weiterhin gibt es komplexere Typen wie numerische Bereiche (<num>) und Kategorien (<kat>).

Einzigartige und reservierte Bezeichnungen Die Bezeichner für verschiedene Objekte und Module werden durch einzigartige Bezeichnungen wie <odd_id>, <module_id>, <tax_id>, <tag_id>, <label_id>, und <file_id> definiert. Zusätzlich sind reservierte Bezeichnungen für Maßeinheiten und Messgrößen vorgesehen (<einheit_id>, <measure_id>) vorgesehen.

Importmechanismus Der Importmechanismus wird über den <import>-Befehl definiert, der es ermöglicht, externe YAML-Dateien einzubinden. Dies geschieht durch die Angabe des Befehls IMPORT, gefolgt von Dateinamen, die jeweils mit einem Tabulator eingerückt und mit der Dateiendung ".yml" versehen sind.

```
/*YAML Format*/
<nl> ::= "\n" | "\r\n"
<i> ::= "\t"
```

```
<ws> ::= (" " | "\t")

/* Zeichen, Zahlen und Operatoren */
<letter> ::= [a-zA-Z]
<char> ::= "_" | "/" | "%"
<dot> ::= "."
<nonZeroDigit> ::= [1-9]
<digit> ::= [0-9]
<relationaloperator> ::= "<" | ">" | ">=" | "<="
<hyphen> ::= "-"

/* Bezeichner (Identifier) */
<identifier_start_char> ::= <letter> | <char>
<identifier_part_char> ::= <identifier_start_char> | <digit>
<identifier> ::= <identifier_start_char> (<identifier_part_char>)*

/* Dateinamen-Basis */
<filename_base> ::=<filename_char>+
<filename_char> ::= <letter> | <digit> | <char> | <hyphen> | <dot>

/* Datentypen*/
<float> ::= <hyphen>? ( "0" | <nonZeroDigit> <digit>* ) <dot> <digit>+
<integer> ::= <hyphen>? ( "0" | <nonZeroDigit> <digit>* )
<bool> ::= "true" | "false"
<numerisch> ::= <relationaloperator> ( <float> | <integer>) <ws>+ <einheit_id>
<enum> ::= <hyphen> <ws> <tax_id>
<kategorisch> ::= ( "["< float> <hyphen> <float> "]" <ws>+ <einheit_id> )
        | ( "["< integer> <hyphen> <integer> "] " <ws>+ <einheit_id> )
        | ( "["<identifier> <hyphen> <identifier> "]" )
        | <numerisch>
<data_type> ::= "integer" | "float" | "bool"

/*Einzigartige Bezeichnungen*/
<odd_id> ::= <identifier>
<module_id> ::= <identifier>
<tax_id> ::= <identifier>
<tag_id> ::= <identifier>
<label_id> ::= <identifier>

/*Reservierte Bezeichnungen*/
<einheit_id> ::= <identifier>
<measure_id> ::= <identifier>

/*Import*/
<import> ::= "IMPORT:" <nl> <filename>+
<extension> ::= "yml"
<filename> ::= <i> <filename_base> <dot> <extension> <nl>
```

Quellcode 5-8: Syntax der Basiselemente

Basierend auf den beschriebenen Basiselementen folgt die Startsequenz, die eine Wahl zwischen drei Hauptkonstrukten bietet: Taxonomie, COD oder ODD bzw. Module (vgl. Quellcode 5-9). Die nachfolgenden Kapitel orientieren sich an dieser Struktur und beschreiben diese im Detail.

```
1  <start_files> ::= <start_taxonomie> |<startcod> | <startmodule> | <startodd>
```

Quellcode 5-9: Startsequenz der Sprache

5.3. Taxonomie

Die Taxonomie stellt eine Hierarchie von Konzepten dar, die zur Definition von Konzepten und Werten genutzt wird, welche unterschiedliche Umweltbedingungen repräsentieren. Diese bildet die Grundlage für die Beschreibung von Situationsdaten und ODD-Modulen und ist entscheidend für die präzise Analyse und den Vergleich verschiedener Umweltbedingungen. Unterschiedliche Anwendungsfälle, wie Autobahnfahrten oder das Manövrieren auf Parkplätzen, erfordern spezifische und angepasste Beschreibungen.

Um diese Flexibilität zu gewährleisten, kann die Taxonomie entweder vollständig in einer YAML-Datei definiert oder in Form von Teilkonzepten in separaten YAML-Dateien angelegt werden. Diese Teilkonzepte können durch Referenzieren zu einem größeren Ganzen zusammengeführt werden, was eine flexible und wiederverwendbare Struktur ermöglicht. Die Erklärung der Taxonomie erfolgt in zwei Schritten: der Syntax- und der Semantikdefinition. Die Syntaxdefinition in Kapitel 5.3.1 legt die formalen Regeln fest, während die Semantikdefinition in Kapitel 5.3.2 Bedeutung der Konzepte und ihre Anwendung in unterschiedlichen Kontexten beschreibt. Diese klare Trennung sichert die Konsistenz und Eindeutigkeit der Taxonomie.

5.3.1. Syntax

Die Taxonomie bildet die Grundlage für die Strukturierung und Kategorisierung von Konzepten innerhalb einer bestimmten Domäne. In diesem Zusammenhang wird die zugrundeliegende Syntax, die die hierarchisch organisierte Struktur von Konzepten und deren Eigenschaften ermöglicht, im Detail erläutert. Die Notation der Syntax erfolgt in EBNF (vgl. Kapitel 2).

5.3.1.1. Grundlegende Syntax der Taxonomie

Auf die einzelnen Bestandteile der Syntax der Taxonomie wird nachfolgend eingegangen (vgl. Quellcode 5-10).

Beginn einer Taxonomie: Der Start der Taxonomie wird durch <start_taxonomie> signalisiert, welches einen Import (<import>?) und die Hauptdefinition der Taxonomie (<taxonomie>) beinhaltet. Das Fragezeichen (?) zeigt an, dass der Import optional ist.

Definition einer Taxonomie: Die Struktur einer Taxonomie wird durch das Schlüsselwort „TAXONOMIE:“ eingeleitet und umfasst eine oder mehrere Konzeptdefinitionen (<konzept>+), wobei das Pluszeichen (+) die Notwendigkeit mindestens eines Konzepts definiert. Jedes Konzept beginnt mit einem Zeilenumbruch (<nl>) und einer Einrückung (<i>), gefolgt von einer Bezeichnung (<tax_id>) und einem Doppelpunkt. Anschließend kann ein Konzept durch zusätzliche Unterkonzepte oder durch spezifische Eigenschaftszuweisungen (<eigenschaftszuweisung>) erweitert werden.

Eigenschaftszuweisung: Die Definition der Eigenschaften eines Konzepts erfolgt mittels <eigenschaftszuweisung>, die es erlaubt, dem Konzept diverse Eigenschaften zuzuweisen. Diese können Basisdatentypen (<werte>), kategorische Bereiche (<kategorisch>) und Enumerationen (<enumeration>) umfassen (vgl. Kapitel 5.2). Durch diese strukturierte Vorgehensweise wird eine

detaillierte und präzise Beschreibung jedes Konzepts innerhalb der Taxonomie ermöglicht, was zu einer aussagekräftigen Datenstruktur führt.

Typsystem der Taxonomie: Das Typsystem der Taxonomie umfasst Basisdatentypen, kategorische Bereiche, Enumerationen, Messgrößen und Einheiten.

Basisdatentypen (<werte>): Basisdatentypen sind fundamentale, nicht weiter zerlegbare Datentypen wie Booleans (bool), Ganzzahlen (integer) und Gleitkommazahlen (float). Jeder Typ kann mit einer spezifischen Maßeinheit (<einheit_id>) versehen werden, um die Aussagekraft der Daten zu verstärken.

Kategorischer Bereich (<kategorisch>): Kategorische Bereiche definieren eine Menge von vordefinierten, nicht-numerischen Werten (<kat>), die als diskrete, benannte Kategorien fungieren. Diese Typen sind besonders nützlich zur Darstellung von Eigenschaften, die eine feste Anzahl von Ausprägungen besitzen.

Enumeration (<enumeration>): Enumerationen definieren eine explizite Menge von konstanten Werten. Jeder Wert in einer Enumeration repräsentiert eine mögliche Instanz des Typs und wird in einer strukturierten Liste dargestellt.

Messgröße (<measures>): Messgrößen sind spezielle Typen, die quantitative Daten repräsentieren und durch eine eindeutige Identifikation (tax_id) gekennzeichnet sind. Jede Messgröße ist mit einer numerischen Ausprägung (float oder integer) und einer festgelegten Maßeinheit (<einheit_id>) assoziiert.

Einheiten (<einheit_id>): Die Definition von Einheiten spielt eine entscheidende Rolle, insbesondere wenn es um die präzise und korrekte Darstellung von numerischen Werten oder Messgrößen geht. Einheiten ermöglichen es, dass der Kontext interpretiert werden kann und die Bedeutung klar und eindeutig ist.

```
1  /*Start Taxonomie*/
2  <start_taxonomie> ::=  <import>? <taxonomie>
3
4  /*Taxonomie*/
5  <taxonomie> ::= "TAXONOMIE:" <nl> <inhalt>+
6  <inhalt> ::= <konzept>+
                | ( <konzept> <eigenschaftszuweisung>)+
                | <kategorien>+
                | <enum_list>+
                | <measures>+
7  <konzept> ::= <nl> <i> <tax_id> ":"
9
10 /*boolsche Werte, integer oder float */
11 <eigenschaftszuweisung> ::= <ws>+ <data_type> ( <ws>+ <einheit_id> )?
12
13 /*kategorien */
14 <kategorien> ::= <kategorisch>
15
16 /*enumerationen*/
17 <enum_list> ::= <konzept> (<nl> <i> <enum>)+
18
19 /*measures*/
```

```
<measures> ::= (<nl> <i> <tax_id> <dot> <measure_id>) ":" <ws>+ ((<float> | <integer>) <ws>+ <einheit_id>)
```

Quellcode 5-10: Syntaxnotation der Taxonomie

Um eine nahtlose Verbindung zwischen der allgemeinen Erläuterung der Taxonomiesyntax und ihrer praktischen Anwendung herzustellen, werden mehrere Beispiele aufgezeigt.

5.3.1.2. Basisdatentypen

Das erste Beispiel beschreibt Umweltbedingungen, welche verschiedene Wetterphänomene umfassen (vgl. Quellcode 5-11). Innerhalb der Kategorie „wetter“ werden spezifische Untergruppen wie „wind“ und „niederschlag“ definiert. Die Windgeschwindigkeit wird dabei als Integer-Wert festgelegt, der eine Geschwindigkeit repräsentiert. Der Niederschlag hingegen wird weiter unterteilt in „Regenmenge“, dargestellt als Fließkommazahl und gemessen als Niederschlagsrate, sowie „Regenintensitätstyp“, eine Enumeration, die verschiedene Arten von Regenintensitäten klassifiziert. Ergänzend ermöglicht das Konzept „Eisregen_vorhanden“ die Angabe, ob Eisregen auftritt. Dies wird ausgedrückt durch einen Booleschen Wert (bool).

```
TAXONOMIE:
  umweltbedingungen:
    wetter:
      wind:
        windgeschwindigkeit: integer geschwindigkeit
      niederschlag:
        regenmenge: float niederschlagsrate
        regenintensitätstyp:
          - dynamisch
          - konvektiv
          - orographisch
        eisregen_vorhanden: bool
```

Quellcode 5-11: Beispiel Taxonomie Beispiel bool, integer oder float

5.3.1.3. Kategorische Aufzählungen

Ein weiteres Beispiel demonstriert die Möglichkeit der kategorischen Aufzählungen. Diese können in Form von Bereichen (Ranges) genutzt werden, um verschiedene Stufen oder Kategorien innerhalb einer Taxonomie zu definieren. Dieses Prinzip wird im Beispiel zur Klassifikation der Regenintensität angewandt, bei der „leichter Regen“ und „mäßiger Regen“ jeweils durch spezifische Niederschlagsmengen definiert werden (vgl. Codeblock 5-12). Jeder Kategorie wird ein spezifischer Bereich der Regenmenge zugeordnet, gemessen in Millimeter pro Stunde (mm/h).

- Leichter Regen: Die Kategorie „leichter Regen“ umfasst Niederschlagsmengen, die weniger als 2,5 mm/h betragen. Dies wird durch die Notation < 2.5 mm/h ausgedrückt, was bedeutet, dass alle Regenmengen unter diesem Wert als „leichter Regen“ klassifiziert werden.
- Moderater Regen: Für „moderaten Regen“ wird ein geschlossener Bereich von 2,5 mm/h bis 7,6 mm/h definiert, dargestellt durch [2.5 .. 7.6] mm/h. Dies bedeutet, dass alle Regenmengen, die in diesen Bereich fallen, als „moderater Regen“ eingestuft werden.

```
TAXONOMIE:
  regenintensität:
    leichter_regen:
```

```
    regenmenge: < 2.5 mm/h
  moderater_regen:
    regenmenge: [2.5 - 7.6] mm/h
```

Quellcode 5-12:Taxonomiebeispiel Regenkategorien

5.3.1.4. Enumerationen

Das nächste Beispiel verdeutlicht den Einsatz von Enumerationen innerhalb einer Taxonomie, um eine gezielte Liste von Kategorien für bestimmte Konzepte festzulegen (vgl. Quellcode 5-13). Dabei fokussiert es sich auf das Konzept der „befestigten Straße“ (befestigte_straße), für das eine präzise Zusammenstellung spezifischer Straßenbezeichnungen (RQ28, RQ31, RQ36, RQ43.5) eingeführt wird. Jede dieser Bezeichnungen stellt eine anerkannte Eigenschaft innerhalb der Taxonomie dar und dient der gezielten Klassifikation.

Indem spezifische Straßenbezeichnungen aufgeführt werden, wird klar definiert, welche Identifikatoren im Rahmen des „befestigte Straße“-Konzepts Anwendung finden. Das Beispiel illustriert weiter, wie die Enumeration „befestigte_straße“, die diverse Typen von Straßenoberflächenbeschreibungen umfasst, zur präzisen Definition weiterer Straßentypen wie Autobahnen (motorway), lokaler Straßen (local_road) und Bundesautobahnen (bundesautobahn) herangezogen werden kann. Jeder dieser Straßentypen wird mit einer oder mehreren zutreffenden Oberflächenbezeichnungen verknüpft. Dies ermöglicht eine detaillierte Spezifizierung ihrer Merkmale.

```
TAXONOMIE:
  befestigte_straße:
    - RQ28
    - RQ31
    - RQ36
    - RQ43.5
  straßentyp:
    schnellstraße:
      befestigte_straße: [RQ31 - RQ36]
    stadtstraße:
      befestigte_straße:: RQ28
    bundesautobahn:
      befestigte_straße: RQ43.5
```

Quellcode 5-13: Taxonomiebeispiel kategorisch und enumeration

5.3.1.5. Einheiten

Die sorgfältige Definition und Verwaltung spezifischer Maßeinheiten ist entscheidend für die Erhöhung der Typsicherheit und die Verbesserung der Datenvalidierung. Für die Umsetzung sind drei Schritte entscheidend:

1. Definition der Basiseinheit: Die Definition einer Basiseinheit etabliert eine universelle Vergleichsbasis für interne Berechnungen und unterstützt die klare Kommunikation zwischen verschiedenen Modulen des Systems.
2. Zusätzliche zulässige Einheiten: Ergänzend zur Basiseinheit, sind zusätzliche Einheiten notwendig. Jede dieser Einheiten ist exakt definiert und kann verlässlich in die Basiseinheit umgerechnet werden. Dies ermöglicht eine flexible Datenverarbeitung unter Beibehaltung hoher Genauigkeit. Beispielhaft ist dies für die Umrechnung der Temperatur von der Basiseinheit Celsius auf Fahrenheit oder Kelvin gezeigt (vgl. Quellcode 5-14).

3. Implementierung von Umrechnungsfunktionen: Zur Unterstützung der verschiedenen Einheiten werden spezielle Umrechnungsfunktionen genutzt, die eine Konvertierung zwischen den Einheiten ermöglichen.

Die genaue Umsetzung dieser Bestandteile ist nicht Teil der Lösung.

```
Konvertierung:
  Temperatur:
    C:
      F:
        Skalierung: 1.8
        Offset: 32
      K:
        Skalierung: 1.0
        Offset: 273.1
```

Quellcode 5-14: Taxonomie – Umrechnung von Einheiten

5.3.1.6. Messgrößen

Die Syntax für die Definition einer Taxonomie ermöglicht es, Konzepte direkt mit Messgrößen zu verknüpfen (vgl. Quellcode 5-15). Hierfür können spezielle Schlüsselwörter wie „Durchschnitt“, „Minimum“, „Maximum“, „Mittelwert“ und „Perzentil“ herangezogen werden, die für Aggregationsfunktionen vorgesehen sind und in einer separaten Referenzliste detailliert beschrieben werden können. Als beispielhaftes Konzept dient hierbei der „Fußgänger“ (fußgänger). Wesentliche Eigenschaften wie die Häufigkeit des Auftretens (fußgänger.häufigkeit) und das Vertrauensniveau (fußgänger.vertrauensniveau) werden mithilfe der Messgrößen „Risiko“ und „Prozent (%)“ quantifiziert. Durch die Verknüpfung von spezifischen Messgrößen mit aggregierten Datenwerten wird eine detaillierte Darstellung von relevanten Informationen ermöglicht.

```
  fußgänger:
    fußgänger.häufigkeit: float risiko
    fußgänger.vertrauensniveau: float %
```

Quellcode 5-15:Taxonomiebeispiel Messgrößen

5.3.2. Semantik Taxonomie

Die hierarchische Struktur der Taxonomie kann auch als Baumstruktur dargestellt werden. Zur Erklärung der Semantik dieser Struktur wird der Ansatz einfacher Pfad-Bäume genutzt. Dieser Ansatz verzichtet auf komplexe Konzepte wie objektorientierte Programmierung, Vererbung, Komposition oder komplexe Beziehungen zwischen Konzepten, die typischerweise in Ontologien verwendet werden. Stattdessen steht eine klare und einfache Darstellung durch Pfad-Bäume im Vordergrund. [CLR+09].

Zusätzlich zu den Pfad-Bäumen spielt der Flattening-Prozess eine entscheidende Rolle. Flattening ist der Prozess, bei dem die hierarchische Struktur der Taxonomie in eine flache Liste von Pfaden umgewandelt wird. Jeder Pfad repräsentiert die Abfolge von Schlüsseln, die zu einem bestimmten Knoten führen. Dies ermöglicht eine eindeutige Identifizierung und Überprüfung der Einzigartigkeit von Knoten über die gesamte Taxonomie hinweg, insbesondere wenn mehrere YAML-Dateien zusammengeführt werden.

Zur Beschreibung dieser Struktur werden Begrifflichkeiten verwendet, die in der Darstellung von Baumstrukturen üblich sind. Dazu zählen die Begriffe Wurzel, Knoten, Blattknoten und Werte. [CLR+09]

Wurzel - Die Wurzel ist der oberste Knoten des Baums, von dem alle anderen Knoten ausgehen. In der Taxonomie wird die Wurzel analog als Konzept bezeichnet und kann in YAML als Schlüssel verstanden werden. Im Beispiel aus Quellcode 5-11 stellt das Konzept „umweltbedingungen“ die Wurzel dar. Es ist der oberste Knoten, von dem die gesamte Struktur abgeleitet wird. [CLR+09]

Knoten - Ein Knoten ist ein Element im Baum, das andere Knoten oder Werte enthalten kann. Jeder Knoten kann entweder ein Elternknoten (mit untergeordneten Knoten) oder ein Blattknoten (ohne untergeordnete Knoten) sein. Diese Knoten werden ebenfalls als Konzepte bezeichnet und stellen in YAML sogenannte Schlüssel dar (vgl. Kapitel 5.1). [CLR+09] Im Beispiel (vgl. Quellcode 5-11) ist „umweltbedingungen“ der Wurzelknoten, von dem Knoten wie „wetter“, „wind“, „niederschlag“, „windgeschwindigkeit“, „regenmenge“, „regenintensitätstyp“ und „eisregen_vorhanden“ abgeleitet sind.

Blattknoten - Ein Blattknoten ist ein Knoten, der keine untergeordneten Knoten hat. Er stellt das Ende eines Pfades im Baum dar und enthält typischerweise einen Wert. Blattknoten werden ebenfalls als Konzepte bezeichnet und entsprechen in YAML den Schlüsseln, die Werte zugeordnet haben. [CLR+09] Im Beispiel (vgl. Quellcode 5-11) sind „windgeschwindigkeit“, „regenmenge“, „regenintensitätstyp“ und „eisregen_vorhanden“ die Blattknoten.

Werte - Werte sind einem Blattknoten zugeordnet und definieren die genaue Bedeutung oder den Inhalt des Knotens. Diese werden in der Syntax als Eigenschaftszuweisungen eingeführt und stellen in YAML den zugehörigen Wert dar. Sie können beispielsweise Basisdatentypen oder Enumerationen umfassen. [CLR+09] Im Beispiel (vgl. Quellcode 5-11) beschreiben die Werte „dynamisch“, „konvektiv“ oder „orographisch“ die verschiedenen Typen von Regenintensität, die dem Blattknoten „regenintensitätstyp“ zugeordnet sind.

Diese Begriffe tragen dazu bei, die Struktur des Baums klar zu definieren und verdeutlichen, wie die verschiedenen Elemente miteinander verbunden sind und welche Rolle diese in der Taxonomie spielen. Der Flattening-Prozess ergänzt die Pfad-Bäume, indem er eine flache Darstellung der Baumstruktur bietet. Dadurch wird eine effektive Überprüfung und Verwaltung der Taxonomie über mehrere Dateien hinweg ermöglicht.

In den folgenden Kapiteln werden die zentralen Aspekte dieser Semantik behandelt. Zunächst wird die Einzigartigkeit der Konzepte in Kapitel 5.3.2.1 erläutert, wobei die Notwendigkeit betont wird, dass alle Konzepte eindeutig benannt sein müssen, um Überschneidungen und Verwechslungen zu vermeiden. Ebenfalls wird auf die Konsistenz der Taxonomie eingegangen, um sicherzustellen, dass die Struktur logisch kohärent ist und semantische Inkonsistenzen vermieden werden. Das Kapitel 5.3.2.2 zur referentiellen Integrität beschreibt die Bedeutung einer korrekten Auflösung und Validierung aller externen Referenzen innerhalb der Taxonomie. Weiterhin wird das Typsystem in Kapitel 5.3.2.3 behandelt. Im Anschluss daran wird in Kapitel 5.3.2.4 die Interpretation importierter Daten beschrieben. Dies beinhaltet die Integration von importierten Daten in die bestehende Taxonomiestruktur und stellt eine modulare sowie konsistente Gestaltung sicher.

5.3.2.1. Einzigartigkeit

Die Einzigartigkeit von Bezeichnungen spielt eine zentrale Rolle in der semantischen Struktur von Taxonomien, um die Eindeutigkeit der definierten Konzepte und ihrer Beziehungen zu gewährleisten. In jeder Taxonomie müssen alle Konzepte eindeutig benannt werden, um Überschneidungen und Missverständnisse zu vermeiden. Es existieren außerdem reservierte Schlüsselwörter, die innerhalb des Kontextes der Taxonomie oder des übergeordneten Systems bereits eine spezifische Bedeutung haben (z. B. „TAXONOMIE“ oder „Einheiten“). Diese dürfen nicht als Bezeichnungen für neue Konzepte

oder Schlüsselwörter verwendet werden, um semantische Konflikte zu vermeiden. Die Einzigartigkeit der Konzepte wird in mehreren Schritten überprüft.

Schritt 1 - Einzigartigkeit innerhalb eines einzelnen Taxonomie-Files: Zunächst wird die Einzigartigkeit der Konzepte innerhalb eines einzelnen Files überprüft. Dabei werden zwei unterschiedliche Fälle unterschieden:

- Fall 1: Keine kategorische Aufzählung

In diesem Szenario dürfen Konzepte innerhalb eines Files nicht identische Bezeichnungen tragen. Ein Beispiel für eine Verletzung dieser Regel wäre, wenn das Konzept „Regenintensitätslevel“ zweimal innerhalb derselben Taxonomie verwendet wird (vgl. Quellcode 5-16). Obwohl „leichter Regen“ in diesem Beispiel ebenfalls doppelt vorkommt, stellt dies keine Verletzung der Einzigartigkeit dar, da es sich um eine Eigenschaft (hier eine Enumeration) und nicht um ein Konzept handelt.

```
TAXONOMIE:
  wetter::
    regenkategorie
      regenintensitätslevel:
        - leichter_regen
        - moderater_regen
      regenintesitätslevel:
        - leichter_regen
```

Quellcode 5-16:Negativbeispiel Einzigartigkeit Taxonomie

- Fall 2: kategorische Aufzählung

Eine Ausnahme von der oben genannten Regel tritt auf, wenn es sich um eine kategorische Aufzählung handelt. Kategorische Aufzählungen sind dadurch gekennzeichnet, dass numerische Werte verwendet werden, um spezifische Kategorien zu definieren. Diese numerischen Werte repräsentieren unterschiedliche Kategorien innerhalb einer Taxonomie. Wenn jedoch nur zulässige Basisdatentypen wie integer, float oder boolean verwendet werden, ohne dass numerische Kategorien zugeordnet werden, liegt keine kategorische Aufzählung vor (vgl. Kapitel 5.2.1.2). Im Fall der kategorischen Aufzählung wird die Einzigartigkeit der Konzepte durch die Zuweisung unterschiedlicher numerischer Werte gewährleistet, die spezifische Kategorien definieren. In diesem besonderen Fall ist es zulässig, dass die Blattknoten eine mehrfache Verwendung von Konzepten zur Definition von Kategorien vorweisen. Das nachstehende Beispiel zeigt eine korrekt definierte Taxonomie, in der die Regenmenge unter verschiedenen übergeordneten Konzepten (leichter_regen, moderater_regen) verwendet wird, um unterschiedliche Kategorien von Regenintesitäten zu definieren (vgl. Quellcode 5-17). Die Einzigartigkeit der Konzepte wird durch die eindeutigen numerischen Bereiche < 2.5 mm/h und [2.5 .. 7.6] mm/h sichergestellt:

```
TAXONOMIE:
  regenintensitätslevel:
    leichter_regen:
      regenmenge: < 2.5 mm/h
    moderater_regen:
      regenmenge: [2.5 .. 7.6] mm/h
```

Quellcode 5-17:Einzigartigkeit Taxonomie – kategorische Aufzählung

Schritt 2 - Überprüfung der semantischen Konsistenz inkl. Einheiten in Kategorien: Die Überprüfung der semantischen Konsistenz, insbesondere bei kategorischen Aufzählungen, ist entscheidend, um Missverständnisse und Fehlinterpretationen der Daten zu vermeiden. Semantische Inkonsistenzen treten auf, wenn die Bedeutung von Datenwerten logisch nicht schlüssig oder widersprüchlich ist. Das folgende Beispiel zeigt eine solche Inkonsistenz aufgrund der Überschneidung von Wertebereichen (vgl. Quellcode 5-18). Der Wertebereich für „leichter Regen“ ist definiert als weniger als 2,5 mm/h, während „moderater Regen“ bereits bei 2 mm/h beginnt. Dies ist niedriger als der Grenzwert für „leichter Regen“ und führt daher zu einer semantischen Inkonsistenz.

```
TAXONOMIE:
  regenintensitätslevel:
    leichter_regen:
      regenmenge: < 2.5 mm/h
    moderater_regen:
      regenmenge: [2 ... 7.6] mm/h
```

Quellcode 5-18:Überprüfung semantische Konsistenz

Schritt 3 - Flattening: Neben der Überprüfung der Einzigartigkeit innerhalb eines Taxonomie-Files muss diese auch über mehrere Files hinweg sichergestellt werden (vgl. Kapitel 5.3.2.4). Hierfür wird die YAML-Struktur in eine Pfadstruktur überführt, ein Prozess, der als „Flattening“ bezeichnet wird. Dabei wird die hierarchische Struktur eines YAML-Dokuments in eine Liste von Pfaden umgewandelt. Jeder Pfad stellt die Abfolge von Schlüsseln dar, die zu einem bestimmten Knoten führen. Durch das Flattening wird eine klare Übersicht über alle vorhandenen Knoten und deren Pfade geschaffen. [MS04]

Schritt 4 - Einzigartigkeit der Konzepte über mehrere Files: Die Überprüfung der Einzigartigkeit von Knoten beginnt mit der Pfad-basierten Überprüfung. Nach dem Flattening-Prozess wird sichergestellt, dass jeder Knoten im Baum durch einen einzigartigen Pfad identifiziert wird. Dieser Pfad besteht aus einer Abfolge von Schlüsseln, die von der Wurzel zum Blattknoten führen. Jeder Knoten innerhalb eines YAML-Dokuments muss über die gesamte Baumstruktur hinweg einzigartig sein, um semantische Konflikte zu vermeiden. Ein Positivbeispiel illustriert diesen Ansatz (vgl. Quellcode 5-19).

```
TAXONOMIE:
  regen:
    regenintensitätslevel:
      leichter_regen
        regenmenge: < 2.5 mm/h
      moderater_regen:
        regenmenge: [2.5 … 7.6] mm/h
```

Quellcode 5-19: Positivbeispiel Einzigartigkeit Konzepte - regen_taxonomie_vpe.yml

Durch den Import der „regen_taxonomie_vpe.yml“ (vgl. Quellcode 5-19) und der Erweiterung um die Kategorie starker_Regen, entsteht ein einzigartiger Pfad, der keinen Konflikt mit den bestehenden Pfaden verursacht (vgl. Quellcode 5-20).

```
Import:
  regen_taxonomie_vpe.yml
Taxonomie:
  regen:
```

```
    regenintensitätslevel:
      starker_Regen:
        regenmenge: >= 7.6 mm/h
```

Quellcode 5-20: Erweiterung des Positivbeispiels

Die resultierenden Pfade nach dem Flattening sind:

- TAXONOMIE/regen/regenintensitätslevel/leichter_regen/regenmenge
- TAXONOMIE/regen/regenintensitätslevel/moderater_regen/regenmenge
- TAXONOMIE/regen/regenintensitätslevel/starker_regen/regenmenge

In diesem Positivbeispiel hat jeder Knoten in der Baumstruktur einen eindeutigen Pfad, wodurch die Einzigartigkeit der Konzepte gewährleistet wird. Im Negativbeispiel führt die doppelte Verwendung des Knotens "moderater_Regen" zu einer Überschneidung der Pfade, was die Einzigartigkeit verletzt und potenziell zu semantischen Konflikten führt (vgl. Quellcode 5-21).

```
Import:
  regen_taxonomie_vpe.yml
Taxonomie:
  regen:
    regenintensitätslevel:
      moderater_Regen:
        regenmenge: >= 7.6 mm/h
```

Quellcode 5-21: Negativbeispiel Einzigartigkeit Konzepte über mehrere Files

Pfade:

- TAXONOMIE/regen/regenintensitätslevel/leichter_regen/regenmenge
- TAXONOMIE/regen/regenintensitätslevel/moderater_regen/regenmenge (aus regen_taxonomie_v01.yml)
- TAXONOMIE/regen/regenintensitätslevel/moderater_regen/regenmenge (neuer Eintrag)

In diesem Negativbeispiel führt die doppelte Verwendung des Knotens „moderater_Regen“ zu einer Überschneidung der Pfade, was die Einzigartigkeit verletzt und zu semantischen Konflikten führt. Durch die Verwendung von Pfadbäumen und der Pfad-basierten Überprüfung kann sichergestellt werden, dass jeder Knoten in der Baumstruktur eindeutig und konsistent ist. Dies verhindert semantische Konflikte und stellt sicher, dass alle Konzepte in der Taxonomie klar und eindeutig definiert sind.

5.3.2.2. Referentielle Integrität

Die referentielle Integrität spielt eine zentrale Rolle bei der Nutzung von Importfunktionen von YAML-basierten Taxonomien. Diese Funktionalität gewährleistet, dass alle externen Referenzen, die innerhalb einer Taxonomie importiert werden, korrekt aufgelöst und validiert werden können. Die Einhaltung der referentiellen Integrität wird durch folgende Kriterien sichergestellt:

- 100%-ige Übereinstimmung: Referenzierte Dateinamen im <Import>-Statement müssen genau mit den Dateinamen übereinstimmen, die in der verfügbaren Ressourcenbibliothek vorhanden sind. Dies garantiert die korrekte Einbindung der beabsichtigten und existierenden Dateien.

- Existenz: Jede Datei, die im Import angegeben wird, muss tatsächlich in der spezifizierten Bibliothek existieren. Fehlende Dateien verursachen Fehler, da sie die referentielle Integrität unterbrechen.

Zwei Beispielanwendungen verdeutlichen das Konzept:

- Positives Beispiel: Der Import der Datei „wetter_taxonomie_v01.yml“ repräsentiert ein erfolgreiches Szenario, in dem die referenzierte Datei in der Bibliothek existiert und deren Einbindung die referentielle Integrität wahrt (vgl. Quellcode 5-22).
- Negatives Beispiel: Versucht man hingegen, die Datei „infrastruktur_taxonomie_v01.yml“ zu importieren, die nicht in der Bibliothek vorhanden ist, verletzt dies die referentielle Integrität und führt zu einem Fehler. Dieses Beispiel unterstreicht die Bedeutung der genauen Übereinstimmung und Verfügbarkeit der Dateien innerhalb des Systems.

```
#Verfügbare Bibliotheken:
# - wetter_taxonomie_v01.yml
# Positivbeispiel
Import:
  wetter_taxonomie_v01.yml
# Negativbeispiel
Import:
  infrastruktur_taxonomie.yml
```

Quellcode 5-22:Taxonomiebeispiele referentielle Integrität

Für eine erfolgreiche Durchführung ist die strikte Übereinstimmung zwischen der im Importstatement angegebenen Datei und den in der Bibliothek verfügbaren Dateien erforderlich. Dies stellt sicher, dass nur explizit definierte und damit beabsichtigte Inhalte in das Dokument integriert werden. Für Einheiten und Messgrößen gilt dies dagegen nicht, da diese nicht extra über die Import Funktion eingebunden werden müssen. Auch für diese muss jedoch sichergestellt sein, dass die referenzierten Strings zu den erlaubten verwendeten Bezeichnern gehören.

```
# Erlaubte Einheiten
# länge: m, cm, mm
Straßenbreite: float breite
```

Quellcode 5-23: Taxonomie erlaubte Bezeichner

5.3.2.3. Aufbau des Typsystems

Die Gewährleistung von Typensicherheit und die Validierung von Werten sowie Einheiten sind kritische Elemente beim Aufbau einer Taxonomie. Es muss eindeutig bestimmt werden können, welche Typen und Einheiten für spezifische Elemente zulässig sind. Dies umfasst das Erstellen spezifischer Richtlinien, die die Akzeptanz von numerischen Werten, Textstrings oder anderen Datentypen regeln, sowie die Überprüfung der Einheiten gemäß einem etablierten Standard oder einer vordefinierten Liste. Um dies effektiv zu gewährleisten, ist es entscheidend, ein robustes Typsystem zu entwickeln. Dieses System muss klar definierte Typen und strenge Regeln für deren Verwendung in nachfolgenden Systemkomponenten sicherstellen. Es erfordert eine präzise Definition von Typen und Einheiten sowie deren semantisch korrekte Anwendung, aufbauend auf der definierten Syntax (vgl. Kapitel 5.3.1) und Mechanismen zur Validierung dieser Aspekte.

Typsystem – Das hier beschriebene Typsystem basiert auf einer statischen und starken Typisierung. Die statische Typisierung stellt sicher, dass Typen bereits zur Konfigurations- oder Kompilierzeit überprüft werden, wodurch potenzielle Typfehler frühzeitig erkannt werden können. Durch die starke

Typisierung wird die strikte Einhaltung der definierten Typen erzwungen, sodass Konvertierungen oder Typmischungen nur unter klar definierten Bedingungen möglich sind. Dies minimiert das Risiko von Typinkonsistenzen und sorgt für eine hohe Sicherheit und Zuverlässigkeit in der Datenverarbeitung. Zusätzlich wird eine explizite Typdeklaration verwendet, bei der alle Variablen und Werte klar definierten Typen zugeordnet werden müssen, um eine präzise und nachvollziehbare Typverwendung sicherzustellen. [Pie02]

Typüberprüfungen – Typüberprüfungen sehen namentliche und strukturelle Äquivalenz vor und sind entscheidend für die Konsistenz der Typverwendung in der Taxonomie und den nachfolgenden Modulen:

- Namentliche Äquivalenz: Dies bedeutet, dass Typen als äquivalent angesehen werden, wenn sie denselben Namen haben. Die konsistente Verwendung der definierten Namen wie „windgeschwindigkeit“ gewährleistet, dass die gleiche Definition aus der Taxonomie übernommen wird.
- Strukturelle Äquivalenz: Diese liegt vor, wenn Typzuweisungen, wie die Zuordnung von Zahlenwerten zu integer oder float, strukturell mit den Typdefinitionen übereinstimmen. [Pie02]

Typkonsistenz zwischen Taxonomie und Modulen – Typkonsistenz ist entscheidend, um sicherzustellen, dass die innerhalb der Taxonomie definierten Typen und Strukturen konsistent über verschiedene Module und Systemkomponenten hinweg angewendet werden:

- Implementierung: Um Typkonsistenz zu gewährleisten, sind die Typdefinitionen zentral verwaltet und müssen über ein gemeinsames Repository oder eine Schnittstelle zugänglich gemacht werden. Die zentrale Definition der Einheiten und Typen sowie die Validierung von Modulen anhand dieser zentralen Vorgaben ist ein gängiger Ansatz in der Softwareentwicklung und Systemarchitektur, der in verschiedenen Arbeiten detailliert beschrieben wird [BW11, Coo89, Kle17].
- Durchsetzung: Es muss sichergestellt werden, dass alle Systemkomponenten, die die Taxonomie nutzen, die Typdefinitionen einhalten.

Anhand eines Beispiels wird dies weiter veranschaulicht (vgl. Quellcode 5-24). In der Taxonomie wird die „windgeschwindigkeit“ als „float“ definiert und „geschwindigkeit“ weist auf die physikalische Einheit hin, in der diese Zahl gemessen wird (z. B. m/s oder km/h). Diese Definition wird dann in Modulen verwendet, um zulässige oder unzulässige Betriebsbedingungen festzulegen. Die Konsistenz zwischen Taxonomie und Moduldefinition wird in Kapitel 5.5.4.3 detailliert beschrieben.

```
1 wind:
2   windgeschwindigkeit: float geschwindigkeit
```

Quellcode 5-24:Taxonomie Typkonsistenz

Die Konsistenz der Typdefinitionen und -strukturen über die Taxonomie und die darauf aufbauenden Module hinaus ist eine grundlegende Anforderung, um die Integrität und Effizienz des Gesamtsystems zu gewährleisten. Durch die zentrale Verwaltung in der Taxonomie und strenge Durchsetzung der Typkonsistenz können Fehlerquellen minimiert und die Entwicklung sowie Wartung von Systemkomponenten erheblich vereinfacht werden.

5.3.2.4. Interpretation der importierten Daten

Das Ziel des Imports besteht darin, bestehende Taxonomiekonzepte nahtlos zu integrieren und gezielt zu erweitern. Dabei wird die Interpretation importierter Daten beschrieben, um sicherzustellen, dass

diese Daten aus „Import“-Anweisungen reibungslos in die übergeordnete Taxonomie eingebettet werden. Dieser Prozess umfasst typischerweise die Verarbeitung und das Zuordnen von Schlüssel-Wert-Paaren aus YAML-Dateien in die bestehende Struktur der Taxonomie. Das Zusammensetzen einer Taxonomie aus mehreren Dateien lässt sich mittels der „IMPORT“-Anweisung realisieren. Um dies zu veranschaulichen, wird das nachfolgende Beispiel einer Basistaxonomie „import_file1.yml“ (vgl. Quellcode 5-25 unten) betrachtet.

```
TAXONOMIE:
  umweltbedingungen:
    regen:
      regenintensität:
        regenmenge: float regenrate
```

Quellcode 5-25:Interpretation importierter Daten – import_file1.yml

Diese Basistaxonomie kann durch eine zweite Datei erweitert werden, die „import_file1.yml“ importiert (vgl. Quellcode 5-26).

```
IMPORT:
  import_ file1.yml
TAXONOMIE::
  umweltbedingungen:
    regen:
      regenintensität:
        regenintensitätstyp:
          - dynamisch
          - konvektiv
          - orographisch
```

Quellcode 5-26: Interpretation importierter Daten - import_file 2.yml

Nach dem Auflösen der Referenzen ergibt die durch “import_file2.yml“ (vgl. Quellcode 5-26) theoretisch folgende kombinierte Beschreibung, die aus der Integration der Inhalte beider Dateien resultiert (vgl. Quellcode 5-27).

```
TAXONOMIE:
  umweltbedingungen:
    regen:
      regenintensität:
        regenmenge: float precipitation_rate
        regenintensitätstyp:
          - dynamisch
          - konvektiv
          - orographisch
```

Quellcode 5-27: Aufgelöste Referenzen von import_file 2.yml

Durch den Einsatz der Importanweisungen wird die Wiederverwendung bestehender Taxonomiedefinitionen ermöglicht und gleichzeitig die Struktur und Konsistenz der übergeordneten Taxonomie gewahrt. Dies fördert eine modulare und effiziente Gestaltung von Taxonomien, indem die Erweiterung an spezifische Bedürfnisse ohne Redundanz ermöglicht werden. Ein potenzielles Risiko besteht jedoch darin, dass Referenzen, auf die verwiesen wird, nachträglich geändert werden könnten. Dies wirft die Frage auf, ob solche Änderungen noch im Sinne des Nutzers sind und die beabsichtigte Integrität der Taxonomie gewahrt bleibt. Dieses Risiko wird in der vorliegenden Doktorarbeit nicht

abschließend beantwortet, sondern als ein wichtiger Aspekt für zukünftige Untersuchungen hervorgehoben.

5.4. Current Operational Domain

Per Definition beschreibt eine COD ein spezifisches Set von Umgebungsbedingungen an einem spezifischen Ort und zu einer spezifischen Zeit (vgl. Kapitel 2.1.2). Diese Definition impliziert die Notwendigkeit, dass die Darstellung einer COD essenzielle Informationen, wie den zeitlichen und geografischen Kontext, klar abbilden muss. CODs können entweder in tabellarischer Form oder als YAML strukturiert werden. Üblicherweise wird die tabellarische Darstellung bevorzugt, da sie die effiziente Speicherung, Verwaltung und das Abrufen strukturierter Daten ermöglicht. Im weiteren Verlauf dieser Arbeit werden beide Darstellungsformen verwendet.

Tabellarische Darstellung: Für die tabellarische Darstellung gilt:

- Jede Zeile in der COD-Tabelle repräsentiert eine Messung für einen spezifischen zeitlichen Umfang (TEMPORAL_EXTENT) und räumlichen Umfang (SPATIAL_EXTENT).
- Eine einzige Spalte für den TEMPORAL_EXTENT ist vorgesehen, die für alle Zeilen Nicht-Null-Werte enthält.
- Eine einzige Spalte für den SPATIAL_EXTENT ist ebenfalls vorgesehen, die für alle Zeilen Nicht-Null-Werte enthält.

Jede weitere Spalte, außer den zeitlichen und räumlichen Umfängen, repräsentiert ein Konzept aus der Taxonomie. Für dieses gilt:

- Numerische Werte: Jedes numerische Feld ist einem einheitlichen Einheitstyp zugeordnet und als Teil des Spaltenkopfes spezifiziert, beispielsweise „regenmenge;mm/hr“. Wenn möglich, werden spezifische Werte angegeben.
- Kategoriale Werte: Jeder zugewiesene Wert basiert auf einem taxonomischen Konzept ohne zugeordnete Einheiten. Wo möglich, wird ein einzelner Wert zugewiesen.
- Boolesche Werte: Diese werden durch kategoriale Spalten dargestellt, wobei die Werte auf „true“ und „false“ beschränkt sind.
- Objektzählspalten: Eine ganze Zahl gibt eine Zählung an, wie „fußgänger.anzahl;anzahl“, und repräsentiert einen spezifischen erfassten Wert.

Umgang mit fehlenden Werten

- Fehlende Werte, die auftreten, wenn Sensoren Daten zu unterschiedlichen Zeitpunkten liefern, bleiben entsprechend leer. Sie unterscheiden sich von Standardwerten und werden als unbekannt betrachtet, ohne dass alle fehlenden Werte explizit mit „unknown“ markiert werden müssen. Dies kann verschiedene Ursachen haben, wie beispielsweise, dass Sensoren ihre Daten zu unterschiedlichen Zeitpunkten liefern oder die Daten einfach nicht verfügbar sind. Diese fehlende Werte sind jedoch gültig, die Interpretation dieser fehlenden Werte hängt sowohl von der Semantik als auch von der verwendeten Werkzeugkette ab.

Als Beispiel könnte eine Tabelle zur Repräsentation verschiedener COD folgendermaßen aussehen (vgl. Tabelle 5-1):

Tabelle 5-1: Darstellung einer COD in tabellarischer Form

#	Temporal Extent	Spatial Extent	Regenmenge [mm/h]	Regenlevel [enum]	Fußgänger [bool]	Fußgänger [Anzahl]
1	2023-06-01 08:12:53.784	45.024 10.261	6.214	moderat		
2	2023-06-01 08:12:53.784	45.024 10.261			wahr	1
3	2023-06-01 09:12:44.754	45.024 10.261	1.783	leicht		
4	2023-06-01 08:10:20.784	45.024 10.261			falsch	0
5	2023-06-01 11:20:53.354	45.024 10.261	0.000	Kein Regen		
6	2023-06-01 18:12:53.784	45.024 10.261			wahr	2

YAML – Darstellung: Die Struktur der COD kann ebenfalls, wie die Taxonomie und die ODD sowie Module, mittels YAML beschrieben werden. Nachfolgend wird das obige tabellarische Beispiel übersetzt in YAML gezeigt (vgl. Quellcode 5-28).

```
COD:
  - TEMPORAL_EXTENT:      "2023-06-01 08:12:53.784"
    SPATIAL_EXTENT:       "45.024 10.261"
    REGENMENGE;mm/hr:     6.614
    REGENINTENSITÄTSLEVEL;enum      moderater_regen
  - TEMPORAL_EXTENT:      "2023-06-01 08:12:54.149"
    SPATIAL_EXTENT:       "45.024 10.261"
    FUßGÄNGER;bool:   true
    FUßGÄNGER.ANZAHL; anzahl: 1
  - TEMPORAL_EXTENT:      "2023-06-02 11:42:21.913"
    SPATIAL_EXTENT:       "45.024 10.261"
    REGENMENGE;mm/hr:     1.783
    REGENINTENSITÄTSLEVEL;enum:     leichter_regen
  - TEMPORAL_EXTENT:      "2023-06-02 11:42:22.427"
    SPATIAL_EXTENT:       "45.024 10.261"
    FUßGÄNGER;bool:   false
    FUßGÄNGER. ANZAHL; anzahl: 0
  - TEMPORAL_EXTENT:      "2023-06-02 23:09:02.376"
    SPATIAL_EXTENT:       "45.024 10.261"
    REGENMENGE mm/hr:     0.000
    REGENINTENSITÄTSLEVEL;enum:     kein_regen
  - TEMPORAL_EXTENT:      "2023-06-02 23:09:02.50"
    SPATIAL_EXTENT:       "45.024 10.261"
    FUßGÄNGER;bool:   true
    FUßGÄNGER. ANZAHL;anzahl: 2
  - TEMPORAL_EXTENT:      "2023-06-02 18:33:57.681"
    SPATIAL_EXTENT:       "45.024 10.261"
    REGENMENGE;mm/hr:     0.000
    REGENINTENSITÄTSLEVEL;enum:     kein_regen
  - TEMPORAL_EXTENT:      "2023-06-02 18:34:04.262"
    SPATIAL_EXTENT:       "45.024 10.261"
```

Quellcode 5-28: COD Repräsentation in YAML

Die Taxonomie, die diese Beispiele unterstützt, würde dafür folgendermaßen aussehen (vgl. Quellcode 5-29).

```
TAXONOMY:
  umweltbedingungen:
    wetter:
      regen:
        regenmenge: float regenrate
        regenintensitätslevel:
          leichter_regen:
            regenmenge: < 2.5 mm/h
          moderater_regen:
            regenmenge: [2.5 .. 7.6] mm/h
        vru:
          fußgänger: boolean
          fußgänger.anzahl: integer anzahl
```

Quellcode 5-29:Taxonomiebeispiel für COD Repräsentation

Ergänzend sei darauf hingewiesen, dass in der Praxis CODs aus unterschiedlichen Datenquellen generiert werden können, die sich sowohl geografisch auch zeitlich überlappen können. Aus diesem Grund ist es notwendig, mehrere Dateien, die CODs in überlappenden geografischen Bereichen und Zeitintervallen abbilden, zu konsolidieren. Obwohl eine fusionierte COD-Datei verschiedene Spalten oder Felder enthalten kann, muss die fusionierte COD einen einheitlichen Satz von Feldern aufweisen. Diese Thematik der Datenkonsolidierung wird jedoch nicht im Rahmen dieser Arbeit behandelt.

5.5. ODD und Module

Um die Modularität der ODD zu erreichen, wird diese in unterschiedliche Module gegliedert. Zusammen bilden diese Module eine kohärente Spezifikation der ODD. In diesem Kapitel werden daher sowohl die einzelnen Module als auch die ODD behandelt. Für die Spezifikation von zulässigen und unzulässigen Betriebsbedingungen wird eine Logik benötigt, die diese Bewertung unterstützt. Der Auswahlprozess der passenden Logik wird in Kapitel 5.5.1 detailliert dargestellt. Anschließend beschreibt Kapitel 5.5.2 die Modularisierung der ODD und erläutert die verschiedenen Modultypen. Die darauffolgenden Kapitel konzentrieren sich auf die Syntax in Kapitel 5.5.3 und die Semantik in Kapitel 5.5.4.

5.5.1. Auswahl der zugrundeliegenden Logik

In der theoretischen Informatik und Softwareentwicklung spielen Logiken eine zentrale Rolle bei der Modellierung, Analyse und Verifikation komplexer Systeme. Diese bieten die Möglichkeit, Prämissen und Schlussfolgerungen auf präzise und mathematisch fundierte Weise zu formulieren und zu überprüfen. Durch deren Anwendung lassen sich Eigenschaften oder Systemverhalten mit hoher Genauigkeit beschreiben und analysieren. Die Auswahl der passenden Logik ist daher entscheidend für die Modellierung einer ODD, um die spezifischen Anforderungen zu erfüllen. Dieses Kapitel widmet sich der Erläuterung der Unterschiede zwischen verschiedenen Logiken und demonstriert deren Anwendung anhand eines Beispiels. Ziel ist es, die Stärken und Grenzen verschiedener Logiken im Kontext der ODD-Modellierung zu beleuchten. Das gewählte Beispiel beschreibt die zulässigen Betriebsbedingungen für ein System auf der Autobahn:

- Das System kann auf der Autobahn fahren.
- Das System kann auf der Autobahn fahren, wenn eine Baustelle vorhanden ist.

Dabei wird eine zulässige Autobahn charakterisiert durch:

- Weiße Fahrbahnmarkierung, die Anzahl der Fahrspuren ist kleiner gleich 3 und der Straßentyp ist Autobahn.

Eine zulässige Baustelle wird definiert anhand:

- Gelbe Fahrbahnmarkierung und die Anzahl der Fahrspuren ist kleiner gleich 2.

Das Beispiel wird anhand der drei grundlegenden Logiken Aussagenlogik, Beschreibungslogik und typisierte Prädikatenlogik erster Stufe (vgl. Kapitel 2.2) beschrieben. Durch den Vergleich desselben Beispiels aus der Perspektive der drei Logiken wird es möglich, ihre spezifischen Vor- und Nachteile im Kontext der Modellierung einer ODD zu verstehen. Am Ende des Kapitels wird ein Vergleich zwischen den Logiken und den Anforderungen aus der Problemanalyse gezogen, um die am besten geeignete Logik für den dargestellten Anwendungsfall zu identifizieren.

5.5.1.1. Aussagenlogik

Die Aussagenlogik konzentriert sich auf die Verknüpfung und Bewertung von Aussagen durch logische Operatoren zur Bestimmung von deren binären Wahrheitswert (vgl. Kapitel 0). Für das gegebene Beispiel können die Aussagen in folgende Variablen übersetzt werden, die jeweils als wahr oder falsch interpretiert werden können:

- F: Zulässige Bedingungen
- A: Straßentyp Autobahn
- W: Weiße Fahrbahnmarkierung
- G: Gelbe Fahrbahnmarkierung
- L2: maximale 2 Fahrspuren
- L3: maximal 3 Fahrspuren

Damit lässt sich das Beispiel, dass das System auf der Autobahn fahren kann, wenn kein Baustellenabschnitt vorhanden ist oder wenn eine Baustelle mit gelber Fahrbahnmarkierung und maximal 2 Fahrspuren vorhanden, aussagenlogisch wie folgt darstellen:

$$F \leftrightarrow \big(A \ \wedge ((W \wedge L3) \vee (G \wedge L2))\big)$$

5.5.1.2. Beschreibungslogik

Die Beschreibungslogik bietet einen formalen Rahmen zur Repräsentation und zum Schlussfolgern über Wissen in spezifischen Domänen (vgl. Kapitel 2.2.2). Dazu wird Wissen in den zwei Hauptkomponenten der T-Box und der A-Box organisiert. Diese Strukturierung ermöglicht eine klare Unterscheidung zwischen generellem Wissen über Konzepte und Beziehungen in einer Domäne (T-Box) und spezifischen Informationen über Individuen und deren Beziehungen (A-Box). Im Kontext des Beispiels bezüglich der Autobahn, lassen sich die Anwendung der Beschreibungslogik und die Einteilung in T-Box und A-Box wie folgt darstellen:

T-Box – Definition von Konzepten und Beziehungen: Die T-Box umfasst Definitionen von Konzepten, Rollen und Axiomen. Für die Konzepte gilt:

- Autobahn: Dieses Konzept repräsentiert Straßen, die als Autobahnen klassifiziert sind.
- WeißeMarkierung: ein Konzept für Straßen, die weiße Fahrbahnmarkierungen haben.
- GelbeMarkierung: ein Konzept für Straßen, die gelbe Fahrbahnmarkierungen haben.
- Max3Spuren: ein Konzept, das Straßen beschreibt, die maximal drei Fahrspuren aufweisen.
- Max2Spuren: ein Konzept für Straßen mit maximal zwei Fahrspuren.

Für die Rollen gilt:

- hatMarkierung: eine Rolle, die eine Straße mit der Art ihrer Fahrbahnmarkierung verbindet. Diese Rolle bezieht sich auf das Attribut der Fahrbahnmarkierung einer Straße.
    - Domäne: Straße
    - Bereich: {WeißeMarkierung, GelbeMarkierung}
- hatSpurenanzahl: eine Rolle, die eine Straße mit ihrer Spuranzahl verbindet.
    - Domäne: Straße
    - Bereich: {Max3Spuren, Max2Spuren}

Die Definition der Zulässigkeit im Sinne einer ODD kann dann anhand der definierten Konzepte und Rollen folgendermaßen ausgedrückt werden:

- $\text{Fahrbar} \equiv \text{Autobahn} \sqcap \big((\exists \text{hatMarkierung}.\,\text{WeißeMarkierung} \sqcap \exists \text{hatSpurenanzahl}.\,\text{Max3Spuren}) \sqcup (\exists \text{hatMarkierung}.\,\text{GelbeMarkierung} \sqcap \exists \text{hatSpurenanzahl}.\,\text{Max2Spuren})\big)$

A-Box: Spezifische Fakten über Individuen – Die A-Box beinhaltet konkrete Informationen über Individuen (Instanzen von Konzepten) und ihre Beziehungen untereinander. Zum Beispiel könnten spezifische Autobahnen oder Baustellen als Instanzen der entsprechenden Konzepte in der A-Box repräsentiert und durch Rollen miteinander in Beziehung gesetzt werden Diese spezifischen Informationen erlauben es, direkte Schlüsse über die Eigenschaften und Beziehungen einzelner Entitäten zu ziehen. Für dies Beispiel ist das nicht notwendig.

5.5.1.3. Typisierte Prädikatenlogik erster Ordnung

Um das Beispiel in der typisierten Prädikatenlogik erster Ordnung zu modellieren, ist es notwendig, eine Typenhierarchie $\mathbb{T}$ sowie eine passende Signatur Σ zu definieren (vgl. Kapitel 2.2.3). Diese Informationen stammen aus der Taxonomie, welche das relevante Domänenwissen beinhaltet und für die Formalisierung in der typisierten Prädikatenlogik unerlässlich ist. Die Modularisierung der ODD kann in der typisierten Prädikatenlogik durch die Verwendung spezifischer Datentypen aus dieser Hierarchie realisiert werden. Diese Datentypen ermöglichen es, lokale Belegungen festzulegen, die in den jeweiligen Modulen erforderlich sind.

Die Datentyphierarchie enthält den universellen Datentyp $\top$und den leeren Datentyp $\bot$ (vgl. Abbildung 5-1).

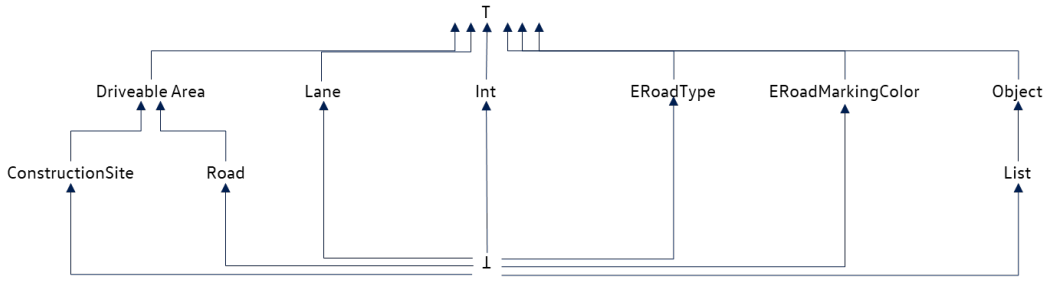

Abbildung 5.1: Darstellung der Datentyphierarchie für das Autobahnbeispiel

Anschließend wird die Typhierarchie $\mathbb{T}$mit der Menge von abstrakten Datentypen $\boldsymbol{\mathcal{T}_A}$, der Menge von dynamischen Datentypen $\boldsymbol{\mathcal{T}_D}$, und mit der Untertyp-Beziehung $\sqsubseteq$ zwischen den Datentypen definiert.

Abstrakte Datentypen:

- $\mathcal{T}\mathrm{A} = \{\top, \mathrm{DrivableArea}, \mathrm{Object}\}$

Dynamische Datentypen:

- $\mathcal{T}\mathrm{D} = \{\mathrm{ConstructionSite}, \mathrm{Road}, \mathrm{Lane}, \mathrm{ERoadMarkingColor}, \mathrm{ERoadType}, \mathrm{Int}, \mathrm{List}\}$
- $\mathrm{ConstructionSite} \sqsubseteq \mathrm{DrivableArea}, \mathrm{Road} \sqsubseteq \mathrm{DrivableArea}, \mathrm{Lane} \sqsubseteq \mathrm{DrivableArea}$
- $\mathrm{ERoadType} \sqsubseteq \top, \mathrm{ERoadMarkingColor} \sqsubseteq \top, \mathrm{Int} \sqsubseteq \top$

Domäne von den Datentyp $ERoadType$:

- $\mathcal{D}ERoadType = \{motorway, countryroad, other\}$

Domäne von den Datentyp $ERoadMarkingColor$:

- $\mathcal{D}ERoadMarkingColor = \{yellow, white, other\}$

Darauf aufbauend, wird die Signatur $\boldsymbol{\Sigma}$ definiert, welche die Variablenmenge $\boldsymbol{V}$, die Funktionsmenge $\boldsymbol{F}$und die Prädikatenmenge P beinhaltet.

Variablen:

- $\mathrm{V} = \{\mathrm{r} : \mathrm{DrivableArea}, \mathrm{l} : \mathrm{Lane}, \mathrm{i}: \mathrm{Int}\}$

Funktionen:

- $\mathrm{F} = \{\mathrm{getRoadType} :\rightarrow \mathrm{ERoadType}, \mathrm{getRoadMarkingColor} :\rightarrow \mathrm{ERoadMarkingColor}, \mathrm{getNumberOfLanes} :\rightarrow \mathrm{Int}, \mathrm{getLane}: \mathrm{Int} \rightarrow \mathrm{Lane}\}$

Prädikate:

- $\mathrm{P} = \{\mathrm{OnMotorway}, \mathrm{OnConstructionSite}, \wedge, \vee, \neg, \leq, <, =, >, \geq\}$
- $\mathrm{OnMotorway}(\mathrm{r}: \mathrm{DrivableArea}) ::= (\mathrm{r}.\mathrm{getRoadType} = \mathrm{motorway}) \wedge (\mathrm{r}.\mathrm{getNumberOfLanes} \leq 3) \wedge (\forall\, \mathrm{i}: \mathrm{Int}\, (\mathrm{i} \leq \mathrm{r}.\mathrm{getNumberOfLanes} \wedge (\mathrm{r}.\mathrm{getLane}(\mathrm{i}).\mathrm{getRoadMarkingColor} = \mathrm{white}))$
- $\mathrm{OnConstructionSite}(\mathrm{r}: \mathrm{DrivableArea}) ::= (\mathrm{r}.\mathrm{getNumberOfLanes} \leq 2) \wedge (\forall\, \mathrm{i} : \mathrm{Int}(\mathrm{i} \leq \mathrm{r}.\mathrm{getNumberOfLanes}) \wedge (\mathrm{r}.\mathrm{getLane}(\mathrm{i}).\mathrm{getRoadMarkingColor} = \mathrm{yellow}))$

Abschließend folgt daraus die formale Spezifikation des Beispiels in der typisierten Prädikatenlogik erster Ordnung:

- $\forall \mathrm{d}: \mathrm{DrivableArea}\, (\mathrm{OnMotorway}(\mathrm{d}) \wedge (\neg \mathrm{OnConstructionSite}(\mathrm{d}) \vee \mathrm{OnConstructionSite}(\mathrm{d})))$

5.5.1.4. Auswahl der geeigneten Logik

Die vorherigen Kapitel haben die drei beschriebenen Logiken anhand eines Beispiels beschrieben. Darauf aufbauend, wird in diesem Kapitel ein Vergleich zwischen diesen Logiken vorgenommen. Dieser stützt sich dabei auf ein Subset der aus der Problemanalyse abgeleiteten Anforderungen, die eine Relevanz für die Logikauswahl zur Modellierung der ODD haben (vgl. Kapitel 3.8). Andere Anforderungen, die beispielsweise die Werkzeugunterstützung adressieren, bleiben unberücksichtigt. Das Ziel ist es, jede Logik hinsichtlich ihrer Eignung für die ODD-Modellierung zu bewerten und eine fundierte Auswahl zu ermöglichen.

Die Bewertung der Logiken erfolgt anhand einer Skala mit den Werten „0“, „1“ und „2“. Diese repräsentieren:

- **0 = Erfüllt die Anforderung nicht**: Die betrachtete Logik bietet keinerlei Unterstützung für die spezifizierte Anforderung.
- **1 = Erfüllt die Anforderung teilweise**: Die Logik bietet eine gewisse Unterstützung, die jedoch Einschränkungen aufweist oder nicht alle Aspekte der Anforderung vollständig abdeckt.
- **2 = Erfüllt die Anforderung vollständig**: Die Logik erfüllt alle Aspekte der spezifizierten Anforderung ohne Einschränkungen.

Die systematische Anwendung der Bewertungsskala bietet eine differenzierte Betrachtung der Stärken und Schwächen jeder Logikform in Bezug auf spezifische Anforderungen und schafft eine objektive Vergleichsbasis, die eine fundierte Entscheidungsfindung unterstützt. Die Auswertung zeigt, dass insbesondere die Aussagenlogik über alle Anforderungen die höchsten Bewertungen erhält, gefolgt von der Beschreibungslogik und zuletzt der typisierten Prädikatenlogik erster Stufe (vgl. Tabelle 5-2). Auf die Details der Bewertung wird nachfolgend eingegangen.

Tabelle 5-2:Vergleich unterschiedlicher Logiken

	Aussagenl.	Beschreibungsl.	Prädikatenl.
Verständlichkeit (REG1)	2	1	0
Arithmetische Ausdrücke (REQ2)	0	0	2
Mathematische Operatoren (REQ4)	0	0	2
Logische Operatoren (REQ4)	2	1	2
Modularisierung (REQ 10,11)	1	2	1
Entscheidbarkeit (REQ15, SAF3, SAF4, SAF5)	2	1	0
Maschinenlesbar (REQ17)	2	2	1
Konsistenz (SAF1)	2	2	1
Wahrscheinlichkeiten (SAF 2)	0	0	0
Performance (ÜBG1)	2	1	0
Summe	**13**	**10**	**9**

Verständlichkeit – Die Verständlichkeit ist ein Schlüsselelement für die breite Anwendbarkeit und adressiert, wie zugänglich die ODD-Beschreibung für "Nicht-Experten" ist. (Anforderung REG1, vgl. Kapitel 3.8)

- Aussagenlogik: Bewertung 2 – Da die Aussagenlogik grundlegende logische Operatoren nutzt und auf einfache, intuitive Weise arbeitet, ist diese relativ leicht verständlich. Dies gilt insbesondere im Vergleich zu komplexeren Logiken.
- Beschreibungslogik: Bewertung 1 – Die Beschreibungslogik ist durch die Nutzung von Ontologien und eine etwas komplexere Syntax weniger intuitiv als die Aussagenlogik, bietet aber eine strukturierte Herangehensweise, die schnell verstanden werden kann.
- Prädikatenlogik erster Stufe: Bewertung 0 – Die Prädikatenlogik erster Stufe ist aufgrund ihrer höheren Komplexität und der Verwendung von Quantoren sowie der Möglichkeit, über individuelle Objekte zu sprechen, wenig intuitiv.

Arithmetisch, logische Ausdrücke – Dies beschreibt die Fähigkeit, arithmetische logische Ausdrücke zu unterstützen (Anforderung REQ2, vgl. Kapitel 3.8).

- Aussagenlogik: Bewertung 0 – Obwohl die Aussagenlogik selbst keine direkte Bewertung arithmetischer Ausdrücke durchführt, können solche Ausdrücke als präpositionale Variablen behandelt werden, deren Wahrheitswerte extern bestimmt werden.

- Beschreibungslogik: Bewertung 0 – Direkte arithmetische Operationen sind nicht Teil der Beschreibungslogik, ähnlich wie bei der Aussagenlogik, und erfordern externe Mechanismen zur Bewertung.
- Prädikatenlogik erster Stufe: Bewertung 2 – Ermöglicht die direkte Modellierung arithmetischer Ausdrücke und deren Vergleich.

Mathematische Operatoren – Die Fähigkeit zur Nutzung mathematischer Operatoren zeigt auf, wie effektiv die ODD mathematische Bedingungen handhaben kann (Anforderung REQ4, vgl. Kapitel 3.8).

- Aussagenlogik: Bewertung 0 – Die Aussagenlogik bietet keine direkte Unterstützung für mathematische Operatoren. Obwohl diese logische Zustände effektiv handhabt, müssen mathematische Ausdrücke außerhalb des Systems bewertet und dann als präpositionale Variablen eingeführt werden.
- Beschreibungslogik: Bewertung 0. Ähnlich wie die Aussagenlogik, bietet die Beschreibungslogik keine direkte Unterstützung für mathematische Operatoren. Ihr Schwerpunkt liegt auf der Modellierung von Beziehungen und Eigenschaften, nicht auf arithmetischen Berechnungen.
- Prädikatenlogik erster Stufe: Bewertung 2. Die Prädikatenlogik erster Stufe ermöglicht die direkte Modellierung und Bewertung mathematischer Ausdrücke und unterstützt vollständig mathematische Operatoren. Dies ermöglicht eine umfassende Ausdrucksfähigkeit, die sowohl logische als auch mathematische Beziehungen umfasst.

Logische Operatoren – Die Einbindung logischer Operatoren bewertet, inwiefern die Logik logische Verknüpfungen zwischen verschiedenen Bedingungen ermöglicht. (Anforderung REQ4, vgl. Kapitel 3.8)

- Aussagenlogik: Bewertung 2 – Die Aussagenlogik unterstützt logische Operatoren vollständig, indem grundlegende logische Operatoren wie UND, ODER und NICHT auf eine einfache und intuitive Weise genutzt werden können.
- Beschreibungslogik: Bewertung 1 – Die Beschreibungslogik unterstützt logische Operatoren und ermöglicht die Anwendung dieser Operatoren innerhalb der Definition von Klassen und Beziehungen. Die Fokussierung liegt jedoch mehr auf den Beziehungen und Eigenschaften zwischen den Entitäten, was zu einer leicht reduzierten Bewertung führt.
- Prädikatenlogik erster Stufe: Bewertung 2 – Diese Logikform unterstützt eine breite Palette von logischen Operatoren und ermöglicht eine detaillierte Ausdrucksfähigkeit durch die Verwendung von Quantoren und spezifischen Prädikaten.

Modularisierung – Die Modularisierung beurteilt die Fähigkeit, wiederverwendbare und kombinierbare Teilbereiche zu definieren (Anforderung REQ 10, 11, vgl. Kapitel 3.8).

- Aussagenlogik: Bewertung 1 - Während die Aussagenlogik selbst keine explizite Struktur für Modularisierung bietet, können unabhängige Aussagen konzeptionell als Module behandelt werden, die in größeren logischen Ausdrücken wiederverwendet werden.
- Beschreibungslogiken: Bewertung 2 - Beschreibungslogiken unterstützen die Modularisierung und Wiederverwendung sehr gut, insbesondere durch die Definition von Klassen und die Vererbung.
- Prädikatenlogik erster Stufe: Bewertung 1 - Modularisierung ist möglich, aber nicht so explizit oder strukturiert wie in der Beschreibungslogik oder in spezifischen Programmiersprachen.

Entscheidbarkeit – Die Entscheidbarkeit prüft, ob für jede Bedingung innerhalb der ODD immer bestimmt werden kann, ob diese erfüllt ist oder nicht (Anforderung REQ15, SAF3, SAF4, SAF5, vgl. Kapitel 3.8).

- Aussagenlogik: Bewertung 2 – Die Aussagenlogik ist entscheidbar, was bedeutet, dass für jede Aussage bestimmt werden kann, ob sie wahr oder falsch ist.
- Beschreibungslogik: Bewertung 1 – Während viele Fragmente der Beschreibungslogik entscheidbar sind, können Komplexität und spezifische Konstrukte die Entscheidbarkeit limitieren.
- Prädikatenlogik erster Stufe: Bewertung 0 – Prädikatenlogik erster Stufe ist nicht immer entscheidbar, was bedeutet, dass nicht für jede Aussage bestimmt werden kann, ob sie wahr oder falsch ist.

Maschinenlesbar – Die Maschinenlesbarkeit stellt sicher, dass die ODD effizient von Computersystemen verarbeitet werden kann (Anforderung REQ17, vgl. Kapitel 3.8).

- Aussagenlogik: Bewertung 2 – Die Aussagenlogik ist maschinenlesbar, da sie auf einem klaren und einfachen Satz von Operatoren und Regeln basiert, die leicht von Computersystemen interpretiert werden können.
- Beschreibungslogiken: Bewertung 2 – Beschreibungslogiken sind gut für maschinelle Verarbeitung und Interpretation ausgelegt, insbesondere in semantischen Webtechnologien.
- Prädikatenlogik erster Stufe: Bewertung 1 – Obwohl maschinenlesbar, kann die Komplexität der Prädikatenlogik die Verarbeitung erschweren.

Konsistenz – Konsistenz und Widerspruchsfreiheit sind entscheidend für die Zuverlässigkeit und Eindeutigkeit der ODD (Anforderung SAF1, vgl. Kapitel 3.8).

- Aussagenlogik: Bewertung 2 – Die Aussagenlogik ermöglicht es, konsistente Systeme zu erstellen, indem sichergestellt wird, dass keine widersprüchlichen Aussagen (d. h. Aussagen, die gleichzeitig wahr und falsch sind) gemacht werden.
- Beschreibungslogik: Bewertung 2 – Die Beschreibungslogik bietet starke Mechanismen zur Sicherstellung der Konsistenz innerhalb der definierten Ontologien.
- Prädikatenlogik erster Stufe: Bewertung 1 -–Konsistenz kann erreicht werden, erfordert aber sorgfältige Formulierung und kann durch die höhere Komplexität erschwert werden.

Wahrscheinlichkeiten – Die Fähigkeit, Häufigkeiten und Wahrscheinlichkeiten zu modellieren, um die Abbildung von Unsicherheiten in der ODD zu ermöglichen (Anforderung SAF 2, vgl. Kapitel 3.8).

- Aussagenlogik: Bewertung 0 – Während die Aussagenlogik selbst keine Wahrscheinlichkeiten direkt modelliert, können Wahrscheinlichkeitsaussagen als externe Bedingungen behandelt werden.
- Beschreibungslogik: Bewertung 0 – Standardmäßig wird keine Unterstützung für die Modellierung von Wahrscheinlichkeiten geboten.
- Prädikatenlogik erster Stufe: Bewertung 0 – Wie die Aussagenlogik, unterstützt die Prädikatenlogik standardmäßig keine Wahrscheinlichkeiten.

Performance – Performance bewertet die Effizienz der ODD-Beschreibung hinsichtlich Skalierbarkeit und Komplexitätsmanagement. (Anforderung ÜBG1, vgl. Kapitel 3.8)

- Aussagenlogik: 2 – Aufgrund ihrer Einfachheit und Entscheidbarkeit kann die Aussagenlogik effizient von Computern verarbeitet werden, was zu guter Performance führt, besonders bei der Auswertung von logischen Ausdrücken.
- Beschreibungslogik: 1 – Die Performance kann variieren und hängt stark von der Komplexität der Ontologie und der verwendeten Inferenzmaschine ab.

- Prädikatenlogik erster Stufe: 0 – Die Komplexität und potenzielle Unentscheidbarkeit können zu Performance-Problemen führen, insbesondere bei umfangreichen oder komplexen Domänen.

Die Bewertung unterstreicht, dass die Aussagenlogik hinsichtlich der vorgegebenen Anforderungen am besten abschneidet, offenbart jedoch Defizite bei der Handhabung arithmetisch-logischer Ausdrücke und der Verarbeitung von Wahrscheinlichkeiten. Eine effektive Strategie zur Überwindung dieser Limitationen bietet der Einsatz externer Erweiterungen, die präpositionale Variablen und spezialisierte Bewertungssysteme einbinden. Für die Verarbeitung arithmetischer und mathematischer Operationen können präpositionale Variablen herangezogen werden, um die Resultate dieser Berechnungen zu repräsentieren. Externe Funktionen übernehmen dann die Auswertung dieser Ausdrücke und ordnen den Variablen adäquate Wahrheitswerte zu. Zur Modellierung von Wahrscheinlichkeiten lassen sich ebenso präpositionale Variablen definieren, deren Wahrheitsgehalte durch externe Wahrscheinlichkeitsanalysen festgelegt werden. Dieser Ansatz ermöglicht es, auch komplexe Bedingungen und Szenarien mittelbar im Kontext der Aussagenlogik abzubilden, indem die Ergebnisse externer Analysen als binäre Werte in das logische Gefüge integriert werden. Für die weitere Ausarbeitung wird die Aussagenlogik genutzt.

5.5.2. Modularisierung der ODD

Die ODD wird in unterschiedliche Modultypen strukturiert, um der Herausforderung der Modularität begegnen. Diese Modularisierung ermöglicht nicht nur die Wiederverwendbarkeit von Komponenten, sondern beeinflusst auch direkt die Syntax und Semantik der zu entwickelnden domänenspezifischen Sprache. Die verschiedenen Modultypen erfordern spezifische sprachliche Konstruktionen, um die Beziehungen, Abhängigkeiten und Hierarchien zwischen den Modulen korrekt abzubilden. Daher ist die Modularisierung mehr als nur ein methodisches Konzept – sie stellt einen integralen Bestandteil der Sprachdefinition dar.

5.5.2.1. Aufbau der ODD

Die ODD stellt die oberste Hierarchieebene dar, innerhalb derer die spezifischen Rahmenbedingungen für den Betrieb des ADS definiert werden. Diese legt fest, welche Situationen und Betriebsbedingungen das ADS bewältigen kann, indem sie eine detaillierte Auflistung von Modulen und zugehörigen Bezeichnungen (Labels) vorschreibt. Die Entwicklung des ADS erfolgt durch mehrere Teams, die sich jeweils auf bestimmte Domänen konzentrieren. [MDR+20, Way20]. Daher ist es von entscheidender Bedeutung, eine monolithische Spezifikation der ODD zu vermeiden. Ähnlich wie in anderen komplexen Systemen, ist die Zerlegung der Spezifikationen in handhabbare Teile essenziell, um eine strukturierte Entwicklung und Anpassung zu fördern. Die einzelnen Module profitieren von etablierten Designprinzipien, die auch in der Softwareentwicklung Anwendung finden (vgl. Kapitel 6).

Für die Organisation der Module und Labels der ODD wird ein gerichteter, azyklischer Graph (Directed Acyclic Graph, DAG) genutzt (vgl. Abbildung 5.2: ODD-Abhängigkeiten als DAG Abbildung 5.2). [CLR+09] Diese Struktur unterstützt die Etablierung klarer und effizienter Abhängigkeitsbeziehungen zwischen den Modulen, die durch grafische Darstellungen visualisiert werden können. In dieser Konfiguration kann jedes Modul auf andere Module verweisen oder von ihnen abhängen. Dadurch wird eine komplexe Vernetzung innerhalb der ODD ermöglicht. Insgesamt bietet die Verwendung eines DAG mehrere Vorteile. Erstens verhindert dieser zyklische Abhängigkeiten, was vorteilhaft ist, da Rückkopplungsschleifen und potenzielle Inkonsistenzen im System leichter vermieden werden können. Zweitens erleichtert der DAG die Etablierung einer klaren Abhängigkeitshierarchie, wodurch das Verständnis der Modulbeziehungen vereinfacht und die separate Bearbeitung von Inhalten gefördert wird. Drittens ermöglichen die Eigenschaften des DAG einfache Aktualisierungen und Erweiterungen

der Module, ohne dass dabei das Gesamtsystem beeinträchtigt wird. Viertens tragen die klare Trennung und Definition der Modulabhängigkeiten zur Minimierung von Fehlern bei. Schließlich fördert die Struktur die Parallelisierung der Entwicklungs- und Testprozesse, indem Teams unabhängig an Modulen arbeiten können, die nicht direkt voneinander abhängig sind.

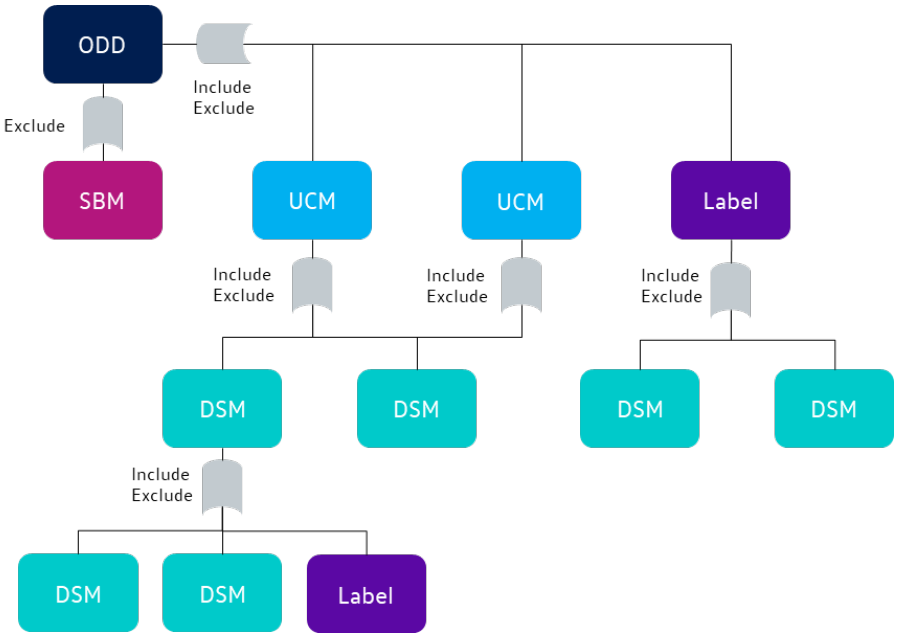

Abbildung 5.2: ODD-Abhängigkeiten als DAG

Die Hierarchie innerhalb der Module ist top-down angelegt, beginnend mit der ODD. Diese dient als Ausgangspunkt für die gesamte Modulstruktur und verweist auf verschiedene Use-Case-Module (UCM), System Boundary Module (SBM) und Labels. Use-Case-Module spezifizieren die anwendungsspezifischen Bedingungen innerhalb der ODD. Jedes Use-Case-Modul beschreibt einen spezifischen Anwendungsfall, wie beispielsweise „Highway-Pilot“ oder „Parkassistent“. Dabei wird festgelegt, unter welchen Bedingungen ein bestimmter Anwendungsfall im System integriert oder ausgeschlossen wird. Die UCMs spezifizieren die Betriebsbedingungen jedoch nicht direkt, sondern integrieren diese durch Domain Specific Module (DSM) und Labels. Durch die Auswahl der gewünschten Kombinationen von Betriebsbedingungen können diese indirekt über UCMs beschrieben werden. Dies fördert das Ziel, die Module so zu gestalten, dass diese isoliert funktionieren und keine negativen Interferenzen auf andere Anwendungsfälle ausüben. [vgl. Kapitel 5.5.2.2]

DSMs sind auf die Beschreibung spezifischer Domänen auf Basis der Taxonomie ausgerichtet. Diese Module legen die genauen Bedingungen für ihre jeweilige Domäne fest, die dann in einem UCM referenziert werden können. DSMs ermöglichen eine detaillierte Spezifikation von Umweltbedingungen, die die Sicherheit und Effizienz des ADS unter verschiedenen Betriebsbedingungen beeinflussen [vgl. Kapitel 5.5.2.3].

SBMs definieren die absoluten Grenzen, innerhalb derer das ADS operieren darf. Diese Module sind entscheidend für die Gewährleistung der Sicherheit des Gesamtsystems, indem sie sicherstellen, dass bestimmte Parameter wie Geschwindigkeiten oder geografische Grenzen nicht überschritten werden. Die SBMS werden direkt auf der Ebene der ODD eingebunden, um eine „globale“ Gültigkeit sicherzustellen [vgl. Kapitel 5.5.2.4].

Labels dienen dazu, Module oder Gruppen von Modulen zu kennzeichnen, die gemeinsame Eigenschaften teilen. Dies ermöglicht es, effizient auf eine Gruppe von Modulen zu verweisen und

diese flexibel in verschiedenen Kontexten innerhalb der ODD zu verwenden. Der Vorteil ist, nicht jedes einzelne Modul muss direkt angesprochen werden. Damit wird auch eine dynamische Anpassung der ODD durch einfache Aktualisierung von Label-Zuweisungen ermöglicht, ohne die zugrunde liegenden Module zu verändern [vgl. Kapitel 5.5.2.5].

In einem kohärenten System bilden die ODD, UCMs, DSMs, SBMs und Labels ein integriertes Netzwerk. Die ODD definiert die Rahmenbedingungen und die Top-Level-Logik des Systems. Innerhalb dieser Rahmenbedingungen spezifizieren Use-Case-Module die Anwendungsbedingungen, während DSMs die detaillierten Betriebsbedingungen bereitstellen. SBMs setzen die sicherheitskritischen Grenzen, und Labels gruppieren diese Module effizient und flexibel. Zusammen bilden diese Komponenten eine modulare und flexible Architektur, die es ermöglicht, das ADS sicher und effektiv zu entwickeln. Die strukturierte und hierarchische Organisation der Module und Labels innerhalb der ODD ermöglicht nicht nur eine klare Definition von Abhängigkeiten, sondern auch die Anpassung und Erweiterung des Systems ohne Beeinträchtigung der Gesamtstruktur.

Ein praktisches Beispiel für die Anwendung der beschriebenen Prinzipien ist eine ODD für ein ADS, das sowohl das Parken im Innen- als auch im Außenbereich umfasst (vgl. Quellcode 5-30). Diese spezifische ODD integriert Module wie „ucm_parken_unzulässig“, „ucm_pudo“ (Pick-Up and Drop-Off) und „ucm_parken_zulässig“. Darüber hinaus bezieht sie ein SBM mit ein, dass eine minimale Parklückenlänge festlegt, indem es zu kleine Parklücken ausschließt. Zusätzlich wird ein Label verwendet, das gefährliche Situationen beim Parken kategorisiert und clustert.

```
…
ODD:
  odd_handle:
    INCLUDE_AND:
      - ucm_parken_unzulässig
      - ucm_pudo
      - ucm_parken_zulässig
    EXCLUDE_OR:
      - sbm_kleine_Parklücken
      - gefährliche_parksituationen
```

Quellcode 5-30: ODD Beispiel für Parken

5.5.2.2. Use Case Module

Ein UCM spezifiziert die Liste der anwendungsspezifischen Bedingungen, die festlegen, wann eine COD in die ODD einbezogen oder ausgeschlossen wird. Dieses Modul sollte so gestaltet sein, dass es keinen Einfluss auf andere Nutzungsfälle hat (vgl. Kapitel 6). Die Definition des UCM zielt darauf ab, die ODD auf eine wartbare Weise zu organisieren, indem es die Bedingungen zwischen verschiedenen Use-Cases separiert. Als Beispiel sei die Notwendigkeit angeführt, die Bedingungen für das Fahren auf der Autobahn von denen für das Fahren in der Stadt zu separieren. Die Bedingungen innerhalb dieses Moduls können Aspekte aus mehreren Domänen kombinieren, wie etwa Wetter, Verkehrsinfrastruktur und dynamische Verkehrsteilnehmer. Das UCM definiert diese Bedingungen nicht direkt, sondern bindet relevante Bedingungen über DSMs oder Labels ein.

Um die Struktur und Funktionsweise eines spezifischen Use-Case-Moduls zu verdeutlichen, wird beispielhaft das Modul „ucm_parken_indoor“ betrachtet, das für das Parken im Innenbereich vorgesehen ist (vgl. Quellcode 5-31). Der Abschnitt INCLUDE_AND listet die Bedingungen auf, die erfüllt sein müssen, damit das Parkmanöver im Innenbereich als zulässig betrachtet wird. Beispielsweise muss eine unterstützende Infrastruktur vorhanden sein, wie diese im DSM

„*dsm_unterstütze_infrastruktur*“ definiert ist. Dies bedeutet, dass bestimmte infrastrukturelle Voraussetzungen erfüllt sein müssen, die den Parkvorgang unterstützen. Der Abschnitt EXCLUDE_OR führt Bedingungen auf, die zum Ausschluss der Situation aus der ODD führen, selbst wenn die anderen Bedingungen erfüllt sind. Dies umfasst eine Liste von nicht unterstützten statischen und dynamischen Objekten (*„dsm_unzulässige_statische_objekte“*, „dsm_*unzulässige _dynamische_ objekte*“) sowie Bedingungen, die nicht unterstützte Geschwindigkeiten beim Parken betreffen (*„dsm_ unzulässige _geschwindigkeiten_parken*“). Zusätzlich wird das Label „gefährliche_parksituationen“ aufgeführt, das Bedingungen gruppiert, die aufgrund ihrer Gefährlichkeit ausgeschlossen werden sollten. In diesem Kontext werden also mehrere Domänen kombiniert, um eine umfassende und sichere Betriebsweise zu gewährleisten.

```
...
MODULES:
  ucm_parken_indoor:
    Type: ucm
    INCLUDE_AND:
      dsm_unterstütze_infrastruktur
    EXCLUDE_OR:
      dsm_unzulässige_statische_objekte
      dsm_unzulässige _dynamische_ objekte
      dsm_unzulässige _geschwindigkeiten_parken
      gefährliche _parksituationen
```

Quellcode 5-31: Beispiel Modularität – Aufbau eines UCM

Diese systematische Strukturierung ermöglicht eine präzise Kontrolle über die Inklusion und Exklusion von spezifischen Bedingungen, wodurch das ADS effektiv und sicher innerhalb seines UseCases und seiner festgelegten ODD operieren kann.

5.5.2.3. Domain Specific Module

Ein DSM etabliert detaillierte Bedingungen für eine bestimmte Domäne basierend auf einer Taxonomie. Die DSMs beschränken sich auf die Spezifikation spezifischer Bedingungen, die dann auf Ebene des UCM integriert oder kombiniert werden können Diese modulare Architektur erlaubt eine komplexe Formulierung von Bedingungen, die über mehrere Domänen hinweg reichen kann. Ein DSM kann dabei in mehreren UCMs sowie in anderen DSMs referenziert werden.

Anhand eines Beispiels soll dies näher erläutert werden (vgl. Quellcode5-32). Es wird ein UCM definiert, das durch die Integration verschiedener DSMs eine umfassende Spezifikation von Betriebsbedingungen ermöglicht. Dazu werden relevante Taxonomiedateien importiert, die die notwendigen Parameter und Werte für die Spezifikation bereitstellen. Innerhalb des kombinierten UCMs werden bestimmte DSMs kombiniert, um eine gesamtheitliche Aussage zu ermöglichen. Die integrierten DSMS haben dabei folgende Bedingungen definiert:

- Die DSM „wetter_modul“ berücksichtigt leichten Schneefall und Windstärken zwischen „stärke00“ und „stärke02“. Extremes Wetter wie schwerer Regen, gewaltiger Regen und Wolkenbrüche werden ausgeschlossen.
- Die DSM „szenerie_modul“ konzentriert sich auf die Sichtbarkeit von Verkehrsschildern.
- Die DSM „konnektivitäts_modul“ definiert geeignete Parameter, damit während es Betriebs sichergestellt ist, dass notwendige Konnektivitätstechnologien wie „netz_3G“ und „netz_4G“ verfügbar sind.

```
IMPORT:
   wetter_taxonomie.yml
   szenerie_ taxonomie.yml
   konnektivität_ taxonomie.yml

MODULES:
   kombinierte_module:
      TYPE: ucm
      INCLUDE_AND:
         - wetter_modul:
         - szenerie_modul:
         - konnektivität_modul:

   wetter_module:
      TYPE: dsm
      INCLUDE_AND:
         schneefall_intensität:
            - leichter_schnee
         windstärke_level:
            - stärke00
            - stärke01
            - stärke02
      EXCLUDE_OR:
         regenintensitätslevel:
            - schwerer_regen
            - gewaltiger_regen
            - wolkenbruch

   szenerie_modul:
      TYPE: dsm
      INCLUDE_AND:
         schild_feature: sichtbar

   konnektivität_modul:
      TYPE: dsm
      INCLUDE_AND:
         konnektivtätstyp:
            - netz_3G
            - netz_4G
```

Quellcode 5-32: Beispiel Modularität - DSM

Durch das Zusammenführen dieser Domänen komplexe und wechselhafte Bedingungen modular beschrieben werden.

5.5.2.4. System Boundary Module

Die SBM definieren die globalen Grenzen eines ADS. Diese globalen Grenzen sind kontextunabhängig und werden daher direkt auf der Ebene der ODD eingebunden, um „global“ gültig zu sein. Jedes SBM beinhaltet lediglich eine einzige Grenzbedingung im INCLUDE-Abschnitt. Für kategoriale Elemente

oder Enumerationen eignet sich eine Struktur mit einem Elternknoten und mehreren Blattknoten (vgl. Kapitel 5.5.3).

Die SBMs sind so konzipiert, dass diese durch eine EXCLUDE OR/AND-Bedingungen innerhalb der ODD referenziert werden. Beispielsweise könnte ein SBM definiert sein, die einen inakzeptable Geschwindigkeitsbereich definiert. Dies bedeutet, dass innerhalb des SBMs im INCLUDE-Abschnitt eine Geschwindigkeitsgrenze inkludiert wird, die das ADS nicht überschreiten darf. Dies führt dazu, dass die ODD dieses spezifische SBM ausschließt, um sicherzustellen, dass die festgelegte Geschwindigkeitsgrenze nicht überschritten werden darf. Solch eine methodische Einbindung der SBMs garantiert, dass definierte Grenzen durch kein anderes Modul überschritten werden können.

Das Beispiel des spezifischen SBMs "sbm_unzulässige_geschwindigkeit" verdeutlicht diesen Ansatz (vgl. Quellcode 5-33). Dieses Modul setzt eine absolute Geschwindigkeitsbegrenzung von 100 km/h als maximal zulässige Geschwindigkeit fest und definiert höhere Geschwindigkeiten als nicht zulässig. Durch die Konfiguration mittels einer EXCLUDE_OR innerhalb der ODD wird sichergestellt, dass das ADS unter keinen Umständen die definierte Geschwindigkeitsgrenze überschreitet, wodurch die Sicherheit des Fahrsystems signifikant erhöht wird. Diese Strategie unterstützt die Sicherheit, indem eine klare und unkomplizierte Methode zur Definition und Überwachung von sicherheitskritischen Systemgrenzen geboten wird.

```
ODD:
  ODDv1:
    EXCLUDE_OR:
      sbm_unzulässige_geschwindigkeit:
MODULES:
  sbm_unzulässige_geschwindigkeit:
    Description: Das ADS kann keine Geschwindigkeiten über 100 km/h händeln.
    TYPE: sbm
    INCLUDE_AND:
      unzulässige_geschwindigkeit: >100 km/h
```

Quellcode 5-33: Beispiel Modularität - SBM

5.5.2.5. Label

Label stellen ein zentrales Instrument für die effiziente Organisation und Wiederverwendung von Modulen innerhalb der ODD dar. Durch das Labeln ist es möglich, multiple Module, die spezifische Bedingungen teilen, disjunktiv zu referenzieren. Dies erweist sich gerade dann als besonders vorteilhaft, wenn zahlreiche Module vorhanden sind und es notwendig ist, alle relevanten Module einzubeziehen, die einen spezifischen Bereich betreffen. Die Verwendung von Labels ermöglicht die flexible Nutzung von Modulen in verschiedenen Kontexten, ohne dass eine Modifikation der Module selbst erforderlich ist. Die entsprechenden Label können in der ODD oder in den Modulen referenziert werden, woraufhin automatisch alle zugehörigen Module referenziert werden.

Ein konkretes Beispiel stellt die Gruppierung von gefährlichen Verkehrssituationen in einem ADS dar (vgl. Quellcode 5-34). Das Label „gefährliche_situation“ wird eingesetzt, um spezifische Module, die gefährliche Situationen darstellen, zusammenzufassen. Dies vereinfacht die Anwendbarkeit, da theoretisch eine unbegrenzte Anzahl von Modulen hinter dem Label stehen kann. Gleichzeitig erfordert dies jedoch auch ein effektives Management der Label und ihrer referenzierten Module. So spezifiziert beispielsweise das Modul „gefährliche_situation_001“, dass gemischte Nutzungsbereiche und Baustellen als potenzielle Gefahrenzonen eingestuft werden. Allerdings wird dieses Modul ausgeschlossen, wenn die Geschwindigkeit des Fahrzeugs unter 15 km/h fällt. Das zweite Modul,

„gefährliche_situation_002“, kennzeichnet Parkspuren als potenzielle Gefahren und schließt bestimmt Geschwindigkeitsbereiche aus. Ein neues Modul „neues_module“ (vgl. Quellcode 5-34) nutzt das Label „gefährliche_situation“, um alle Situationen auszuschließen, die unter das Label fallen. Dieser Ansatz demonstriert, wie Labels innerhalb der ODD dazu beitragen können, das System strukturiert und sicher zu gestalten, indem diese relevante Module entsprechend der definierten Labels gruppiert.

```
Import:
    szenerie_taxonomie.yml
MODULES:
   gefährliche_situation_001:
      TYPE: dsm
      LABEL:
         gefährliche_situation:
      INCLUDE_AND:
         fixierte_zone:
           - gemischter_nutzen
           - baustelle
      EXCLUDE_OR:
         ego_geschwindigkeit_vorwärts: "<15km/h"

   gefährliche _situation_002:
      TYPE: dsm
      LABEL:
         gefährliche_situation
      INCLUDE_AND:
         rechte_spur_typ: parken
      EXCLUDE_OR:
         ego_geschwindigkeit_vorwärts: "<30km/h"

   neues_modul:
      TYPE: dsm
      EXCLUDE_AND:
         gefährliche_situation
```

Quellcode 5-34: Beispiel Modularität - Label

5.5.3. Syntax ODD und Module

Aufbauend auf dem Konzept der Modularität, wird eine Syntax definiert, die diese Anforderungen unterstützt. Die Definition der Syntax erfolgt in EBNF und wird sowohl für die Beschreibung von Modulen als auch für die ODD vorgenommen.

5.5.3.1. Syntax Module

Module ermöglichen eine modularisierte Definition zulässiger und unzulässiger Betriebsbedingungen basierend auf einer Taxonomie. Die Taxonomie dient dabei als Basis für die Strukturierung und Kategorisierung von Konzepten und legt fest, welche Konzepte zur Definition benutzt werden können. Die zugrundeliegende Syntax wird nachfolgend detailliert erläutert (vgl. Quellcode 5-35).

Beginn der Moduldefinition – Die Anweisung <startmodule> leitet die Definition des Moduls ein. Diese besteht aus einem Importbefehl <import>, der Taxonomie und der Modul-Definition.

- <import>: Referenziert externe Taxonomien oder Moduldateien, die bei der Definition genutzt werden können.
- <taxonomie>: Dies ist optional und kann verwendet werden, um referenzierte Taxonomien „inline“ zu erweitern (vgl. Kapitel 5.3.1.1).
- <modules>: Bildet den Ausgangspunkt für die nachfolgende Definition der Modulkonzepte und -eigenschaften.

Modulkonzepte und -eigenschaften – Diese definieren die grundlegende Struktur für Module und legen fest, welche spezifischen Inhalte und Eigenschaften jedes Modul haben kann. Dabei wird syntaktisch zwischen zwei Haupttypen von Modulen unterschieden: dem „Use Case“- und dem „Domänenspezifischen“ Modul sowie dem Modul für „Systemgrenzen“.

Das Schlüsselwort "MODULES:" leitet die Definition der Module ein und markiert den Beginn der Auflistung verschiedener Module. Die Zeile „<module_inhalt> definiert die Struktur eines einzelnen Moduls. Dabei zeigt das Pluszeichen „+“ an, dass mindestens ein Modulinhalt erforderlich ist. Die <module_id> ist dabei ein eindeutiger Bezeichner des Moduls, während <module_properties> die ihm zugeordneten spezifischen Eigenschaften beschreibt.

Die Eigenschaften eines Moduls werden durch <module_properties> spezifiziert. Dabei wird zwischen <moduleproperty1> und <moduleproperty2> unterschieden, wodurch verschiedene Modultypen mit jeweils angepassten Eigenschaften definieren werden können.

Modultyp Use-Case und Domänenspezifisch – Die Syntaxbeschreibung für <moduleproperty1> ist darauf ausgerichtet, verschiedene Attribute für UCM und DSM zu definieren. <moduleproperty1> legt den Typ des Moduls fest. Dieses kann entweder als "UCM" oder als "DSM" klassifiziert werden. Nach dieser Festlegung können verschiedene optionale Attribute hinzugefügt werden:

- Titel: Ein optionaler Titel kann spezifiziert werden (<i> "TITLE: " <string> <nl>), der als einfache Zeichenkette (<string>) dargestellt wird.
- Beschreibung: Eine Beschreibung des Moduls kann ebenfalls optional angegeben werden (<i> "DESCRIPTION: " <string> <nl>).
- Tags: Tags dienen zur weiteren Kennzeichnung und sind optional (<i> "TAGS:" <nl> <i> "- " <tag_id>+ <nl>). Jedes Tag beginnt mit einem Bindestrich und steht in einer neuen Zeile. Tags tragen primär informativen Charakter. Im Gegensatz zu Labeln tragen Tags nur einen informativen Charakter und können beispielsweise die Suchfähigkeit von Modulen unterstützen.
- Labels: Ähnlich wie Tags, bieten Labels eine zusätzliche Möglichkeit zur Kategorisierung und sind in der Syntax optional gehalten (<i> "LABELS: " <nl> <i> "- " <label_id>+ <nl>). Im Gegensatz zu Tags bieten Label aber weitere Möglichkeiten zur gebündelten Referenzierung und Einbindung von Modulen (vgl. Kapitel 5.5.4.6).

Die logischen Operatoren in der Syntax bestimmen, wie Module unter bestimmten Bedingungen einbezogen oder ausgeschlossen werden:

- Inklusionsbedingungen: Der <includetypemodule> definiert die logische Verknüpfung für die Einbeziehung von Modulen und kann eine zusätzliche Option beinhalten, um Bedingungen zu verschachteln ((<include_and> <nl> <includeoropt>?) | (<include_or> <nl> <includeandopt>?)). Die Nutzung dieser zusätzlichen Option ist optional und auf eine Verschachtelung beschränkt.
- Exklusionsbedingungen: Der <excludetypemodule> bestimmt die logische Verknüpfung für den Ausschluss von Modulen und kann ebenfalls eine zusätzliche Option enthalten, um

Bedingungen zu verschachteln ((<exclude_or> <nl> <excludeandopt>?) | (<exclude_and> <nl> <excludeoropt>?)). Auch hier ist die Nutzung der zusätzlichen Option optional und auf eine Verschachtelung beschränkt.

Modultyp Systemgrenze – Dieser Modultyp spezifiziert die Eigenschaften von Modulen, die als Systemgrenze (SBM) klassifiziert sind. Diese Module sind dafür ausgelegt, die Grenzbedingungen festzulegen, die das System nicht überschreiten darf (vgl. Kapitel 5.5.2). In ihrer Struktur ähneln diese den Modultypen UCM oder DSM, haben jedoch einigen Unterschiede. Die Definition dieser Module beinhaltet ebenfalls Titel, Beschreibung, Tags und Labels, setzt aber den Modultyp spezifisch auf "SBM" fest und ist beschränkt auf einen logischen Operator (<logicaloperator2>).

- <moduleproperty2>: Spezifiziert die Parameter für SBM. Bis auf den Typ und den logischen Operator sind die anderen Moduleigenschaften optional.
- <logicaloperator2>: Gibt die zulässige logische Operation für die Definition der Systemgrenze vor.

Logische Ausdrücke – Um festzulegen, wie und unter welchen Bedingungen bestimmte Betriebsbedingungen zulässig oder unzulässig sind, werden logische Ausdrücke verwendet. Diese Ausdrücke sind durch spezifische Schlüsselwörter definiert, die die Grundlage für die Inklusions- und Exklusionsbedingungen bilden.

Der Einsatz der Schlüsselwörter „INCLUDE_AND“ und „INCLUDE_OR“ legt fest, welche Konzepte unter welchen definierten Bedingungen zulässig sind:

- <include_and> initiiert eine Sequenz, in der alle angegebenen Konzepte und deren Eigenschaften gleichzeitig erfüllt sein müssen, um ein Modul einzubeziehen. Diese AND-Verknüpfung erfordert eine strikte Übereinstimmung mit allen spezifizierten Bedingungen.
- <include_or> ermöglicht die Einbeziehung eines Moduls, wenn mindestens eines der angegebenen Konzepte zutrifft. Diese OR-Verknüpfung bietet Flexibilität, indem verschiedene akzeptable Definitionen ermöglicht werden.
- <include_and2>: bildet eine Ausnahme und wird speziell für die Definition der Systemgrenze verwendet, wobei nur eine einzige Konzept-Eigenschaftszuweisung erlaubt ist. Diese spezielle Anwendung erlaubt eine präzise Spezifikation von Systemgrenzen.

Ähnlich wie bei den Inklusionsbedingungen, bestimmen die Schlüsselwörter „EXCLUDE_OR“ und „EXCLUDE_AND“ die Kriterien für den Ausschluss von Konzepten:

- <exclude_or> verwendet eine OR-Verknüpfung, die ein Konzept ausschließt, sobald eines der spezifizierten Kriterien erfüllt ist. Dies erlaubt einen breiten Ausschluss basierend auf mehreren möglichen unerwünschten Bedingungen.
- <exclude_and> erfordert, dass alle spezifizierten Bedingungen gleichzeitig erfüllt sind, um ein Konzept auszuschließen. Diese AND-Verknüpfung ist nützlich, wenn ein Ausschluss nur unter einer vollständigen Reihe spezifischer Bedingungen erfolgen soll.

Jede dieser logischen Sequenzen beginnt mit einem Schlüsselwort, gefolgt von mindestens einem Konzept <konzept> und einer zugehörigen Eigenschaftszuweisung <eigenschaftszuweisung>, die in der importierten Taxonomie festgelegt sind. Diese Konzepte und Eigenschaften werden visuell durch Zeilenumbrüche und Einrückungen hervorgehoben und müssen den Vorgaben der Taxonomie entsprechen.

Logische Ausdrücke (Opt) – logische Operatoren bieten zusätzliche Flexibilität, indem sie die Möglichkeit zur Verschachtelung von Bedingungen für die Einbeziehung oder den Ausschluss von

Konzepten unter spezifischen Bedingungen ermöglichen. Diese erweiterten Operatoren ergänzen die bereits definierten Standardbedingungen und ermöglichen eine präzisere Definition.

Für die erweiterte Definition von zulässigen Konzepten können optionale logische Operatoren verwendet werden, um die Bedingungen zu verschachteln. Solche Verschachtelungen können folgende Formen annehmen:

- Die Kombination von „<include_and>„ gefolgt von „<includeoropt>„ (optional) ermöglicht es, eine AND-Verknüpfung zu definieren und optional eine OR-Verknüpfung hinzuzufügen, falls zusätzliche, alternative Bedingungen berücksichtigt werden sollen.
- Alternativ kann „<include_or>„ gefolgt von „<includeandopt>„ (optional) genutzt werden, um eine OR-Verknüpfung zu etablieren, mit der Möglichkeit, diese durch eine zusätzliche AND-Verknüpfung zu erweitern.

Für den erweiterten Ausschluss von Konzepten kann dies folgendermaßen aussehen:

- Die Verwendung von „<exclude_or>„ gefolgt von „<excludeandopt>„ (optional) erlaubt die Definition einer OR-Verknüpfung, mit der Option, diese durch eine AND-Verknüpfung zu ergänzen. Um ein Modul auszuschließen, müssen dann mehrere spezifische Kriterien gleichzeitig erfüllt sein.
- Umgekehrt kann „<exclude_and>„ gefolgt von „<excludeoropt>„ (optional) verwendet werden, um eine AND-Verknüpfung zu definieren, die durch eine OR-Verknüpfung erweitert werden kann. Sobald eines von mehreren möglichen Kriterien zutrifft, wird ein Modul ausgeschlossen.

Die Nutzung dieser Verschachtelungen ist optional, bietet jedoch zusätzliche Präzision, indem sie komplexere logische Bedingungen beschreibbar macht. Dadurch wird eine genauere Anpassung an spezifische Betriebsbedingungen möglich.

```
1  /*Start Module*/
2  <startmodule> ::= <import> (<taxonomie>)? <modules>
3
4  /*Module*/
5  <modules> ::=  "MODULES:" <module_inhalt>+
6  <module_inhalt> ::= <nl> <i> <module_id> ":" <module_properties>
7  <module_properties> ::= <nl> (<moduleproperty1> | <moduleproperty2>)
8
9  /*UCM + DSM*/
10 <moduleproperty1> ::= (<i> "TYPE: " ("ucm" | "dsm") <nl>) (<i> "TITLE: " <string> <nl>)? (<i>
   "DESCRIPTION: " <string> <nl>)? (<i> "TAGS:" <nl> <i> "- " <tag_id>+ <nl>)? (<i> "LABELS: " <nl> <i> "- "
   <label_id>+ <nl>)? <logicaloperator1>
11 <logicaloperator1> ::= (<includeoptionmodule> | <excludeoptionmodule> | <includeoptionmodule>
   <excludeoptionmodule>)
12 <includeoptionmodule> ::= <i> <includetypemodule>
13 <includetypemodule> ::= (<include_and> <nl> <includeoropt>?) | (<include_or> <nl> <includeandopt>?)
14 <excludeoptionmodule> ::= <i> <excludetypemodule>
15 <excludetypemodule> ::= (<exclude_or> <nl> <excludeandopt>?) | (<exclude_and> <nl> <excludeoropt>?)
16
17 /*SBM*/
18 <moduleproperty2> ::= (<i> "TYPE: " "sbm" <nl>) (<i> "TITLE: " <string> <nl>)? (<i> "DESCRIPTION: "
   <string> <nl>)? (<i> "TAGS:" <nl> <i> "- " <tag_id>+ <nl>)? (<i> "LABELS: " <nl> <i> "- " <label_id>+ <nl>)?
   <logicaloperator2>
19 <logicaloperator2> ::= <i> <include_and2>
```

```
20
21  /*Logical_Expressions*/
22  <logicalexpression> ::= <include_and> | <include_or> | <exclude_or> | <exclude_and>
23  <include_and> ::= "INCLUDE_AND: " (<konzept> <eigenschaftszuweisung>)+
24  <include_or> ::=  "INCLUDE_OR: " (<konzept> <eigenschaftszuweisung>)+
25  <exclude_or> ::= "EXCLUDE_OR: " (<konzept> <eigenschaftszuweisung>)+
26  <exclude_and> ::= "EXCLUDE_AND: " (<konzept> <eigenschaftszuweisung>)+
27  <include_and2> ::= "INCLUDE_AND: " (<konzept> <eigenschaftszuweisung>)
28
29  /*Logical_Expressions_Opt*/
30  <logicalexpressionopt> ::= <includeandopt> | <includeoropt> | <excludeoropt> | <excludeandopt>
31  <includeoropt> ::= <i> "OR:" (<konzept> <eigenschaftszuweisung>)+ <nl>
32  <includeandopt> ::= <i> "AND:" (<konzept> <eigenschaftszuweisung>)+ <nl>
33  <excludeandopt> ::= <i> "AND:" (<konzept> <eigenschaftszuweisung>)+ <nl>
34  <excludeoropt> ::= <i> "OR:" (<konzept> <eigenschaftszuweisung>)+ <nl>
```

Quellcode 5-35: Syntax der Moduldefinition

Um das Verständnis der theoretischen Darstellung der Syntax zu verbessern, wird nachfolgend die praktische Implementierung in Modulen anhand von ausgewählten Beispielen demonstriert.

Moduldefinition Typ DSM – Zuerst wird die Definition eines DSM gezeigt. Dies kann genutzt werden, um spezifische Kriterien für Autobahnauffahrten auf der A2 in Deutschland zu definieren (vgl. Quellcode 5-36). Zulässige Betriebsbedingungen sind ein Kurvenradius über 20 Meter und Fahrbahnen, die als Autobahn klassifiziert sind. Unzulässige Betriebsbedingungen sind Geschwindigkeiten von über 130 km/h und Straßenbreiten von 2,5 Metern oder weniger.

```
1   IMPORT:
2    deutsche_autobahn_szenerie.yml
3   MODULES:
4    autobahnauffahrt:
5      TYPE: dsm
6      TITLE: "Autobahnauffahrten A2 Deutschland"
7      DESCRIPTION: Beschreibung der Autobahnauffahrten A2 in Deutschland zwischen Berlin und Dortmund
8      TAGS:
9       - "projekt highwaypilot"
10     LABELS:
11      - auffahrt
12     INCLUDE_AND:
13      kurvenradius: >20m
14      straßentyp:
15           - autobahn
16     EXCLUDE_OR:
17      geschwindigkeit:  >130km/h
18      straßenbreite: <= 2,5m
```

Quellcode 5-36: Syntaxbeispiel Module - DSM

Verschachtelte Moduldefinition – Des Weiteren bietet die Syntax die Möglichkeit, Bedingungen zu verschachteln, um die Anforderungen für ein Modul, das temporäre Straßenstrukturen und spezifische Kreuzungsmerkmale berücksichtigt, zu definieren (vgl. Quellcode 5-37). Dies ermöglicht eine differenziertere Spezifikation von Bedingungen (vgl. Kapitel 5.5.4).

```
1   Import:
2    deutsche_autobahn_szenerie.yml
3   MODULES:
```

```
    beispielmodul:
      TYPE: dsm
      INCLUDE_AND:
        temporäre_straßenstrukturen: Baustellenumleitungen
        straßentyp: Autobahn
        OR:
          kreuzungsmerkmal: kanalisiert
          kreisverkehr_größe: Doppelkreisverkehr
```

Quellcode 5-37: Syntaxbeispiel verschachtelte Moduldefinition

Moduldefinition Typ SBM – Um Systemgrenzen zu definieren, darf maximal ein Konzept inkludiert werden. In dem gegebenen Beispiel wird eine Systemgrenze spezifiziert, die Steigungen von mehr als 10 % betrifft (vgl. Quellcode 5-38). Diese Spezifikation kann entscheidend für Systeme sein, die in hügeligen oder bergigen Gebieten eingesetzt werden sollen.

```
Import:
 szenerie_global.yml
MODULES:
  beispielmodul:
    TYPE: sbm
    INCLUDE_AND:
      steigung_bergauf: >10 %
```

Quellcode 5-38:Moduldefinition Typ Systemgrenze

5.5.3.2. Syntax ODD

Die ODD wird durch eine Syntax definiert, die eine hohe Ähnlichkeit zu der Syntax der Module aufweist (vgl. Quellcode 5-39). Allerdings werden in der ODD keine Konzepte oder Eigenschaften definiert, sondern direkt die definierten Module oder Label als zulässig oder unzulässig eingebunden.

Start der Definition

Die ODD-Definition beginnt mit einem Startsymbol <startodd>, dass die folgenden Elemente umfasst:

- <import>: Dies wird verwendet, um notwendige externe Dateien oder Module zu importieren, die innerhalb der ODD genutzt werden.
- <odd>: Dieses Element leitet die Definition der Operational Design Domain ein.

Definition der ODD

- "ODD: ": Dieses Schlüsselwort signalisiert den Beginn der ODD-Definition.
- <odd_inhalt>: Enthält die eigentliche Definition der ODD. Diese wird strukturiert durch:
  - <nl>: Ein Zeilenumbruch zur Trennung der Definitionsteile.
  - <i>: Dynamische Einrückung, die zur Strukturierung der Inhalte nach YAML dient.
  - <odd_id>: Eine eindeutige Kennzeichnung der ODD.

Inklusions- und Exklusionsoptionen

Innerhalb der ODD können spezifische Module durch Inklusions- (<includeoptionodd>) und Exklusionsoptionen (<excludeoptionodd>) behandelt werden. <includeoptionodd> und <excludeoptionodd> definieren dabei, ob Module oder Labels, basierend auf definierten Bedingungen, einbezogen oder ausgeschlossen werden sollen.

- "INCLUDE_AND: " oder "INCLUDE_OR: ": Hierbei erfordert INCLUDE_AND das gleichzeitige Zutreffen aller Bedingungen, um ein Modul oder Label einzubeziehen, während INCLUDE_OR die Inklusion erlaubt, sobald eine der Bedingungen erfüllt ist.

- "EXCLUDE_AND: " oder "EXCLUDE_OR: ": Bei EXCLUDE_AND müssen alle Bedingungen gleichzeitig zutreffen, um ein Modul oder Label zu exkludieren. Währenddessen bei EXCLUDE_OR eine Exklusion schon dann erfolgt, sobald eine der Bedingungen erfüllt ist.

<condition_odd> legt dabei die Module oder Labels fest, die für die Inklusion oder Exklusion angewendet werden sollen:

- <condition_odd>: Beinhaltet die eindeutigen Modul-IDs und Label_IDs, die eingebunden werden sollen. Diese verweisen wiederum auf spezifische Betriebsbedingungen, die innerhalb des referenzierten Moduls oder des Labels definiert sind.

```
/*Start ODD/Module*/
<startodd> ::= <import> <odd>

/*ODD*/
<odd> ::= "ODD: " <odd_inhalt> +
<odd_inhalt> ::= <nl> <i> <odd_id> ": " (<includeoptionodd> | <excludeoptionodd> | <includeoptionodd> <excludeoptionodd>)
<includeoptionodd> ::= <nl> <i> <includetype> <condition_odd>+
<excludeoptionodd> ::= <nl> <i> <excludetype> <condition_odd>+
<includetype> ::= "INCLUDE_AND: " | "INCLUDE_OR: "
<excludetype> ::= "EXCLUDE_AND: " | "EXCLUDE_OR: "
<condition_odd> ::= <nl> <i> (<module_id> |<label_id>)
```

Quellcode 5-39: Syntaxnotation der ODD

Das nachfolgende Beispiel veranschaulicht die Einbindung und den Ausschluss spezifischer Module und Labels innerhalb einer ODD (vgl. Quellcode 5-40). Hier werden Module, die eine Autobahnszenerie beschreiben, in die ODD inkludiert, während Labels, die schlechtes Wetter und schlechte Konnektivität repräsentieren, ausgeschlossen werden. Die spezifischen Bedingungen für Einbeziehung und Ausschluss sind direkt in den jeweiligen Modulen definiert. Weitere Einzelheiten zur Interpretation und Anwendung dieser Bedingungen sind in der Semantik erläutert (vgl. Kapitel 5.5.4).

```
IMPORT:
  modul_bibliothek.yml
ODD:
  Odd_autobahnpilot:
    TITLE: ODD für den Autobahnpiloten v10.3
    INCLUDE_OR:
      auffahrten
      abfahrten
      baustellen
      autobahn
    EXCLUDE_OR:
      schlechtes_wetter #label
      schlechte_konnektivität #label
```

Quellcode 5-40: Beispiel ODD für einen Autobahnpiloten

5.5.4. Semantik Module und ODD

Nachfolgend wird auf die Semantik der Module und der ODD eingegangen.

5.5.4.1. Einzigartigkeit

Die Einzigartigkeit von Identifikatoren in einem komplexen System ist für dessen einwandfreie Funktion von grundlegender Bedeutung. Durch die konsistente Zuweisung eindeutiger Bezeichner für ODDs, Module, Taxonomien, Tags und Labels wird sichergestellt, dass jede Komponente im System klar

identifizierbar und referenzierbar ist (vgl. Kapitel Syntax 5.3.1). Ergänzend zu der bereits erklärten Einzigartigkeit bei der Taxonomie, sind folgende weitere Punkte zu beachten:

- ODDs und Module: Die Zuweisung eindeutiger IDs zu ODDs und Modulen ist unerlässlich, um die korrekte Zuordnung und das Management von Bedingungen zu ermöglichen.
- Tags und Labels: Spezifische <tag_id> und <label_id> garantieren eine konsistente und fehlerfreie Anwendung von Klassifizierungen im System.
- Dateien: Die Wichtigkeit von <file_id> für das Importieren und Integrieren externer Ressourcen ist entscheidend. Nur durch eindeutige Datei-IDs wird ein korrektes Importieren und Referenzieren gewährleistet.
- Konzepte innerhalb der Moduldefinition und über referenzierte Module hinweg, dürfen nur einmalig als zulässig oder unzulässig definiert werden.

Es ist von höchster Bedeutung, dass keine Überschneidungen zwischen den genannten Identifikatoren auftreten und dass diese nicht mit reservierten Schlüsselwörtern, festgelegten Einheiten oder Messgrößen im Konflikt stehen. Die Überwachung der Einzigartigkeit erfordert systematische Kontrollen, die Folgendes sicherstellen:

- Validierung neuer IDs: Jede neue ID muss gegen bestehende überprüft werden, um Duplikate auszuschließen.
- Schutz reservierter Schlüsselwörter: Dürfen nur im syntaktisch korrekten Kontext verwendet werden.

Diese Prinzipien werden anhand von Beispielen verdeutlicht (vgl. Quellcode 5-41 und Quellcode 5-42). Ein Beispiel für die Notwendigkeit einzigartiger Identifikatoren zeigt sich beim Import der Datei module1.yml (vgl. Quellcode 5-41) in modul2.yml (vgl. Quellcode 5-42). Beide Dateien enthalten Module mit demselben Namen. Dies führt zu Konflikten. Zum einen innerhalb Datei modul2.yml, indem der Modulname „modul2“ doppelt verwendet wird. Zum anderen zwischen Datei „modul1.yml“ und „modul2.yml“ hinweg, indem die Bezeichner „modul1“ doppelt verwendet werden (vgl. Quellcode 5-42, rot markiert). Das zweite Beispiel zeigt eine weitere Verletzung der Einzigartigkeit, indem genutzte Konzepte, wie in diesem Beispiel die Windgeschwindigkeit, mehrfach im selben Kontext verwendet werden.

```
IMPORT:
…
MODULES:
  modul1:
    …
```

Quellcode 5-41: Beispielmodul Einzigartigkeit modul1.yml

```
IMPORT:
  modul1.yml:
MODULES:
  modul2:
    …
  modul1:
    TYPE: dsm
    INCLUDE_AND:
      windgeschwindigkeit: <20km/h
    EXCLUDE_OR:
      windgeschwindigkeit: <50km/h
```

```
12    modul2:
13      …
```

Quellcode 5-42: Beispielmodul Einzigartigkeit module2.yml

Ein weiteres Beispiel zeigt den Konflikt zwischen genutzten Identifikatoren, die sowohl in der Taxonomie als auch in Modulen verwendet werden. Dazu werden eine beispielhafte Taxonomie für Szenerieelemente (vgl. Quellcode 5-43) sowie ein Modul zur Definition von zulässigen Straßentypen (vgl. Quellcode 5-44) definiert.

```
1  TAXONOMIE:
2    szenerie_elemente:
3      straßentyp:
4        - autobahn
5        - spielstraße
6      straßenoberfläche:
7        - beton
8        - asphalt
```

Quellcode 5-43: Beispieltaxonomie Einzigartigkeit taxonomie.yml

In beiden werden dieselben Bezeichner genutzt, wodurch die geforderte Einzigartigkeit verletzt wird. Das Problem tritt auf, weil Module Bezeichner benutzen, die in der Taxonomie als Konzept verwendet werden (z.B. „straßentyp“ und „autobahn“). Eine klare Trennung zwischen Modul ID und Taxonomieelementen kann somit nicht mehr hergestellt werden.

```
1  IMPORT:
2    taxonomie.yml
3  MODULES:
4    straßentyp:
5      TITLE: M01
6      TYPE: dsm
7      INCLUDE_AND:
8        straßentyp:
9          - spielstraße
10   autobahn:
11     TITLE: M02
12     TYPE: dsm
13     EXCLUDE_OR
14       straßenoberfäche:
15         - asphalt
```

Quellcode 5-44: Beispielmodul Einzigartigkeit module3.yml

5.5.4.2. Referentielle Integrität

Referentielle Integrität ist für die Funktionsweise von Modulen und ODDs unerlässlich, um Konsistenz und Fehlerfreiheit zu gewährleisten. Dies erfordert, dass alle Referenzen in Modulen und ODDs vorhanden sind und korrekt aufgelöst werden können. Dies Vorgehen ist analog zum Vorgehen in der Taxonomie (vgl. Kapitel 5.3.2.2). Bei der Einbindung externer Dateien über Importanweisungen müssen die Dateinamen genau mit denen in einer Ressourcenbibliothek übereinstimmen, um Fehler zu vermeiden. Es ist auch wichtig, dass Bezeichner innerhalb der Module und ODDs den bereits bestehenden Bezeichnern entsprechen, um die Eindeutigkeit und Korrektheit der Verweise sicherzustellen. Anhand des nachfolgenden Beispiels soll dies verdeutlicht werden. Der Aufbau und

die Funktionsweise einer Ressourcenbibliothek sind dabei nicht Gegenstand der Betrachtung. In modul1.yml (vgl. Querverweis 5-45) wird eine Beispieltaxonomie durch ein Import-Statement eingebunden.

```
IMPORT:
  beispieltaxonomie.yml
MODULES:
  modulbasis:
    TYPE: dsm
    LABEL:
      gefährlich
    INCLUDE_AND:
      kurvenradius: <= 30m
      geschwindigkeit: 100km/h
```

Quellcode 5-45: Modulbeispiel Referenzen – modul1.yml

Hier definiert das Modul Eigenschaften wie Kurvenradius und Geschwindigkeit und wird mit dem Label "gefährlich" versehen. „modul2.yml“ (vgl. Quellcode 5-46) importiert „modul1.yml“ und nutzt die bereits definierten Bedingungen, um weitere Definitionen vorzunehmen.

```
IMPORT:
  modul1.yml
MODULES:
  modulimport_möglichkeit1:
    TYPE: dsm
    INCLUDE_AND:
      modulbasis
      …
```

Quellcode 5-46: Modulbeispiel Referenzen – modul2.yml

Schließlich zeigt „modul3.yml“ (vgl. Quellcode 5-47), wie das Label "gefährlich" aus „modul1.yml“ (vgl. Quellcode 5-45) in einem neuen Kontext verwendet werden kann.

```
IMPORT:
  modul1.yml
MODULES:
  modulimport_möglichkeit2:
    TYPE: dsm
    INCLUDE_AND:
      gefährlich
```

Quellcode 5-47:Modulbeispiel Referenzen – modul3.yml

Diese Beispiele demonstrieren die korrekte Übernahme von Bezeichnern und Labels aus importierten Modulen und unterstreichen die Bedeutung präziser und eindeutiger Benennungen. Die korrekte Anwendung der Import-Statements gewährleistet, dass die Module effektiv miteinander interagieren und die gewünschten Eigenschaften in verschiedenen Modulen konsistent reproduziert werden.

5.5.4.3. Typenkorrektheit

Im Kontext von Typsystemen und Taxonomien bezieht sich Typkorrektheit auf die Übereinstimmung der Datentypen in Ausdrücken oder Zuweisungen, um sicherzustellen, dass sie kompatibel sind. Typkompatibilität wird durch das Typsystem einer Taxonomie bestimmt und ist entscheidend für die

Vermeidung von Laufzeitfehlern oder logischen Fehlern in der Datenverarbeitung (vgl. Kapitel 5.3.2.3). Typkorrektheit erfordert, dass die linke und rechte Seite einer Zuweisung oder eines Vergleichs kompatible Typen aufweisen.

Die linke Seite ist in der Regel die Variable oder der Speicherort, der einen Wert aufnehmen soll oder mit einem Wert verglichen wird. Eine Variable ist ein benannter Speicherbereich, der Daten speichern kann. Der Wert einer Variable kann während der Laufzeit eines Programms geändert werden. Variablen werden mit einem spezifischen Typ deklariert, der bestimmt, welche Art von Daten diese speichern können und welche Operationen mit ihnen durchgeführt werden dürfen.

Die rechte Seite ist der Wert oder Ausdruck, der der Variablen zugewiesen oder mit der diese verglichen wird. Ein Literal ist eine direkte Darstellung eines Wertes in Code. Literale beziehen sich auf feste Datenwerte, die direkt im Quellcode eingebettet sind. Ähnlich wie Variablen, haben auch Literale einen Typ, der auf dem Wert basiert, den sie darstellen.

In Typsystemen ist die Unterscheidung zwischen Variablen und Literalen wichtig, weil beide Typinformationen tragen müssen, um die Typsicherheit zu gewährleisten. Das Typsystem verwendet diese Informationen, um zu überprüfen, ob Operationen, die auf Variablen und Literalen ausgeführt werden, typkompatibel sind. Dabei wird die Typkorrektheit überprüft, indem sichergestellt wird, dass die Werte (sowohl von Variablen als auch von Literalen) in den Ausdrücken kompatibel sind. Für die Überprüfung der Typkorrektheit in den Modulen sind namentliche und strukturelle Äquivalenz relevant (vgl. Kapitel 5.3.2.3):

- Namentliche Äquivalenz: Dies bedeutet, dass Typen als äquivalent angesehen werden, wenn sie denselben Namen haben. Die konsistente Verwendung der definierten Namen wie „windgeschwindigkeit“ gewährleistet, dass die gleiche Definition aus der Taxonomie übernommen wird. Diese Form der Äquivalenz stellt sicher, dass die semantische Integrität der Typen über verschiedene Module und Komponenten hinweg gewahrt bleibt.
- Strukturelle Äquivalenz: Diese liegt vor, wenn Typzuweisungen, wie die Zuordnung von Zahlenwerten zu integer oder float, strukturell mit den Typdefinitionen übereinstimmen. Diese Art von Äquivalenz ist wichtig, um die funktionale Konsistenz zu gewährleisten, sodass Typen basierend auf ihrer tatsächlichen Struktur und nicht nur auf ihrem Namen korrekt funktionieren.

Anhand eines Beispiels wird dies weiter veranschaulicht. In der Taxonomie wird die „windgeschwindigkeit“ als float definiert. Zudem weist die „geschwindigkeit“ auf die physikalische Einheit hin, in der diese Zahl gemessen wird (z. B. m/s oder km/h) (vgl. Quellcode 5-48).

- Variable: windgeschwindigkeit
- Datentyp: float
- Einheit: geschwindigkeit (impliziert, z.B. km/h)

```
1 wind:
2   windgeschwindigkeit: float geschwindigkeit
```

Quellcode 5-48: Beispieltaxonomie Typenkorrektheit – taxonomie1.yml

Ein Modul definiert für die windgeschwindigkeit einen Wert von 20,6 km/h (vgl. Quellcode 5-49).

- Variable: windgeschwindigkeit
- Literal für den Wert: 20.6 (Floating Point Literal)
- Literal für die Einheit: km/h (String Literal)

```
...
  windgeschwindigkeit: 20,6 km/h
```

Quellcode 5-49: Beispielmodul Typenkorrektheit – module1.yml

Die Typüberprüfung erfolgt anhand der in der Taxonomie definierten Typen und der tatsächlichen Werte und Einheiten, die in den Modulen oder Anwendungen genutzt werden.

Überprüfung der namentlichen Äquivalenz – In der Taxonomie „taxonomie1.yml“ (Quellcode 5-48) und im Modul „module1.yml“ (Quellcode 5-49) wird der gleiche Name „windgeschwindigkeit“ verwendet. Da beide Instanzen den exakt gleichen Namen für den Datentyp verwenden, stellt dies ein Beispiel für namentliche Äquivalenz dar.

Überprüfung der strukturellen Äquivalenz – Hier betrachtet man die Kompatibilität der Datenstrukturen selbst. „windgeschwindigkeit“ ist als float typisiert, was bedeutet, dass es eine Gleitkommazahl erwartet. Der im Modul verwendete Wert 20,6 km/h entspricht diesem Typ, da 20,6 eine Gleitkommazahl ist und die Einheit km/h mit der erwarteten Einheit „geschwindigkeit“ übereinstimmt.

Überprüfung des Datentyps:

- erwarteter Typ (Taxonomie): float
- tatsächlicher Typ des Werts (Beispiel): 20.6 (Gleitkommazahl, passt zu float)

Überprüfung der Einheit:

- erwartete Einheit (Taxonomie): geschwindigkeit (angenommen km/h)
- tatsächliche Einheit (Beispiel): km/h

Als Ergebnis liegt eine Typkompatibilität vor. Das zweite Beispiel soll den gegenteiligen Fall aufzeigen. Die Taxonomie „taxonomie2.yml“ definiert das Vorhandensein von Hagel (vgl. Quellcode 5-50). Hier wird erwartet, dass hagel_vorhanden ein Boolescher Wert ist, also true oder false.

- Variable: hagel_vorhanden
- Datentyp: bool

```
...
  hagel_vorhanden: boolean
```

Quellcode 5-50: Beispieltaxonomie Typenkorrektheit – taxonomie2.yml

Das Modul „modul2.yml“ definiert für hagel_vorhanden einen Wert von <3cm (vgl. Quellcode 5-51).

- Variable: hagel_vorhanden
- Literal für den Wert: 3 (Integer Literal)
- Literal für die Einheit: cm (String Literal)

```
...
  hagel_vorhanden: <3cm
```

Quellcode 5-51: Beispielmodul Typenkorrektheit – modul2.yml

Das Modul versucht, einen numerischen Wert mit einer Einheit (<3cm) für eine Boolesche Variable zu verwenden. Dies ist ein Beispiel für fehlende Typkompatibilität, da die Typen boolean und numerisch nicht kompatibel sind.

5.5.4.4. Logische Abbildung

Die Syntax zur Beschreibung der ODD und Module nutzt spezielle Schlüsselwörter, um Elemente und ihre Beziehungen zu beschreiben. Um die logische Auswertung dieser Strukturen zu ermöglichen, ist es entscheidend, eine eindeutige Verbindung zwischen den verwendeten Schlüsselwörtern der Modellierungssprache und der Booleschen Aussagenlogik (vgl. Kapitel 5.5.1.1) herzustellen. Diese Verbindung ermöglicht es, dass die durch die Schlüsselwörter definierten Bedingungen formal logisch interpretiert und auf ihre Korrektheit überprüft werden können. Nachfolgend wird darauf eingegangen, wie das grundlegende Konzept der Aussagenlogik auf die entwickelte Modellierungssprache abgebbildet werden kann. Darauf aufbauend erfolgt die semantische Erklärung der Schlüsselwörter in Bezug auf die logischen Operatoren der Aussagenlogik.

5.5.4.4.1. Abbildung auf das grundlegende Konzept der Aussagenlogik

Das grundlegende Konzept lässt sich anhand einer Taxonomie, einem Modul und einer COD erklären. Die Taxonomie „taxonomie1.yml“ definiert „windgeschwindigkeit“ als Fließkommazahl in Kilometern pro Stunde und „straßentyp“ mit den Enumerationen „Autobahn“ und „Landstraße“ (vgl. Quellcode 5-52).

```
taxonomie1.yml
TAXONOMIE:
   use_case1:
      windgeschwindigkeit: float geschwindigkeit
      straßentyp:
       - autobahn
       - landstraße
```

Quellcode 5-52: Beispieltaxonomie Abbildung Aussagenlogik – taxonomie1.yml

Ein Modul „module1.yml“ könnte dann definiert werden, dass eine bestimmte Windgeschwindigkeit und einen spezifischen Straßentyp einschließt. Beispielsweise sind lediglich Autobahnen und eine Mindestwindgeschwindigkeit von 20,6 km/h zulässig (vgl. Quellcode 5-53).

```
IMPORT:
   taxonomie1.yml
MODULES:
   module1:
      TYPE: dsm
      INCLUDE_AND:
         windgeschwindkeit: < 20.6 km/h
         straßentyp:
            - autobahn
```

Quellcode 5-53: Beispielmodul Abbildung Aussagenlogik – modul1.yml

Des Weiteren ist eine COD vorhanden, die eine Windgeschwindigkeit von 17 km/h und den Straßentyp „landstraße“ aufweist (vgl. Tabelle 5-3).

Tabelle 5-3: Beispielhafte COD zu Windgeschwindigkeit und Straßentyp

#	Temporal Extent	Spatial Extent	Windgeschwindigkeit [km/h]	Straßentyp [enum]
1	2024-06-01 08:12:53:784	*48.0232 11.7153*	17	Landstraße

Die formale Überprüfung zeigt dann, dass die tatsächlichen Bedingungen nicht den Anforderungen des Moduls entsprechen, da der Straßentyp nicht den festgelegten Kriterien genügt. Die Überprüfung erfolgt durch den Abgleich der realen Werte mit den Modulspezifikationen. Die verwendeten Konzepte lassen sich bekannten Konzepte der Aussagenlogik zuordnen.

Atome/Atomare Formeln – Atome sind die grundlegendsten Elemente in logischen Ausdrücken, die nicht weiter unterteilbar sind und einen einzelnen, spezifischen Aspekt einer Situation beschreiben. Zum Beispiel kann die Windgeschwindigkeit ≥ 20,6 km/h als Atom betrachtet werden, das eine spezifische Bedingung der Windgeschwindigkeit enthält. Hier wird die gesamte Bedingung, einschließlich der numerischen Vergleichsoperation, als ein einzelnes Atom betrachtet, da es die Basisinformation ohne logische Verknüpfung mit anderen Informationen darstellt.

Formel – Komplexe Formeln setzen sich aus mehreren atomaren Formeln zusammen, die durch logische Operatoren wie UND (AND) und ODER (OR) verbunden sind. Diese Formeln beschreiben umfassendere Bedingungen oder Regeln innerhalb von Modulen oder ODDs. Zum Beispiel lautet eine Formel im Modul: INCLUDE_AND: {windgeschwindigkeit: <20,6 km/h, straßentyp: autobahn}. Diese Formel verwendet den logischen Operator UND, um auszudrücken, dass beide Bedingungen gleichzeitig erfüllt sein müssen.

Belegung – Eine Belegung ist die Zuweisung spezifischer Werte zu den Atomen in einer gegebenen Situation. Zum Beispiel könnte die Belegung für eine COD „windgeschwindigkeit: 17 km/h, straßentyp: landstraße“ sein.

Interpretation: – Die Interpretation nutzt die Belegungen, um die Wahrheitswerte der komplexeren Formeln zu berechnen. Damit gibt diese an, ob eine bestimmte Situation den festgelegten Bedingungen des Moduls entspricht. In unserem Beispiel würde die Interpretation überprüfen, ob die Werte der realen Situation den definierten Bedingungen im Modul entsprechen.

Wahrheitswerte – Wahrheitswerte sind das Ergebnis der Interpretation. Diese zeigen an, ob die Bedingungen der Formel in der COD erfüllt sind oder nicht. Beispielsweise könnte das Modul eine Windgeschwindigkeit von mindestens 20,6 km/h und den Straßentyp Autobahn definieren. Wenn die COD jedoch eine Windgeschwindigkeit von 17 km/h und den Straßentyp Landstraße aufweist, gibt die Interpretation den Wahrheitswert der Formel als falsch zurück. Die spezifizierten Bedingungen sind nicht erfüllt und daher stimmen das Modul und die COD nicht überein. Anhand einer Wahrheitstabelle kann dies übersichtlich dargestellt werden (vgl. Tabelle 5-4)

Tabelle 5-4:Wahrheitstabelle zur Aussagenlogik

Situation	P: Windgeschwindigkeit < 20,6 km/h	Q: straßentyp: - landstraße	Gesamtaussage (P ∧ Q)
COD #01	True	False	False

5.5.4.4.2. Abbildung und Interpretation der logischen Operatoren

Modulbedingungen haben eine propositionale Semantik. Die propositionale Semantik ist ein Zweig der mathematischen Logik, der sich mit dem Studium von Aussagen und ihren logischen Beziehungen befasst. In diesem Rahmen sind Propositionen Aussagen, die entweder wahr oder falsch sind. Die propositionale Semantik befasst sich mit der formalen Darstellung dieser Propositionen und den logischen Operationen, die an ihnen durchgeführt werden können. [Hof18] Zentrale Konzepte der propositionalen Semantik umfassen propositionale Variablen, die als Symbole für wahrheitswertige Aussagen dienen, welche entweder wahr oder falsch sein können. Eine entscheidende Rolle spielen

auch logische Konnektive, die es ermöglichen, aus einfachen Aussagen komplexere zu bilden. Zu den Wichtigsten gehören:

- Konjunktion (∧), die „und“ repräsentiert: p∧q ist wahr, wenn sowohl p als auch q wahr sind.
- Disjunktion (∨), die „oder“ repräsentiert: p∨q ist wahr, wenn mindestens eines von p oder q wahr ist.
- Negation (¬), die „nicht“ repräsentiert: ¬p ist wahr, wenn p falsch ist.

Diese Konzepte bilden die Grundlage, um die Semantik der durch Module definierten Bedingungen zu beschreiben. Sie bestimmen, wie die Wahrheitswerte der einzelnen Komponenten den Gesamtwahrheitswert einer Bedingung beeinflussen. Der Wahrheitswert eines Moduls ergibt sich aus der Kombination der Werte seiner Include- und Exclude-Abschnitte (vgl. Tabelle 5-5). Include-Abschnitt und Exclude-Abschnitt bilden zusammen den Gesamtwahrheitswert des Moduls.

Tabelle 5-5: Wahrheitstabelle Module/ODD

INCLUDE Section	EXCLUDE Section	Ganzes Modul
False	False	False
True	False	True
False	True	False
True	True	False

Diese Prinzipien sind entscheidend, um zu verstehen, wie Bedingungen in Modulen definiert und bewertet werden.

Es existieren zwei Haupttypen von Include-Abschnitten: INCLUDE_AND und INCLUDE_OR, sowie zwei Arten von Exclude-Abschnitten: EXCLUDE_AND und EXCLUDE_OR (vgl. Kapitel 5.5.3). Die Logik hinter diesen Abschnitten ist entscheidend für die Modellierung und Auswertung der Bedingungen, da auf dessen Basis der Wahrheitswert ermittelt wird. Für die Modellierung dieser Bedingungen gelten folgende Regeln:

- Ein INCLUDE_AND-Abschnitt wird als "wahr" bewertet, wenn alle Bedingungen wahr sind. Dies stellt sicher, dass alle spezifizierten Kriterien erfüllt sein müssen.
- Ein INCLUDE_OR-Abschnitt wird als "wahr" bewertet, wenn mindestens eine der Bedingungen wahr ist.
- Ein EXCLUDE_AND-Abschnitt wird als "wahr" bewertet, wenn alle seine Bedingungen falsch sind. Dies entspricht in der Logik einer Negation. Dies bedeutet, dass die Bedingungen zwar erfüllt sein müssen, das Endergebnis jedoch negiert wird.
- Ein EXCLUDE_OR-Abschnitt wird ebenfalls als "wahr" bewertet, wenn mindestens eine seiner Bedingungen falsch ist. Dies stellt ebenfalls eine Negation dar.

Ergänzend gelten für die Wahrheitswerte der Ausdrücke je nach Datentyp weitere Regeln:

- Boolesche Werte können direkt ausgewertet werden.
- Numerische Werte:
    - Ein Ausdruck für eine obere Grenze wird als wahr bewertet, wenn das angegebene Feld kleiner oder gleich dem festgelegten Schwellenwert ist.
    - Ein Ausdruck für eine untere Grenze wird als wahr bewertet, wenn das angegebene Feld größer oder gleich dem festgelegten Schwellenwert ist.
    - Ein Gleichheitsausdruck wird als wahr bewertet, wenn das angegebene Feld dem im Zustand spezifizierten Wert entspricht.

- Ein Bereichsausdruck wird als wahr bewertet, wenn der Wert des angegebenen Feldes innerhalb der durch den Bereich spezifizierten oberen und unteren Grenzen liegt.
- Eine Enumeration wird als wahr bewertet, wenn der Wert des angegebenen Feldes in der Liste der im Ausdruck spezifizierten Werte enthalten ist.
- Eine Kategorie wird als wahr bewertet, wenn der Wert des angegebenen Feldes entweder innerhalb der durch den Bereich spezifizierten oberen und unteren Grenzen der Kategorie liegt ist oder in der Liste der im Ausdruck spezifizierten Werte enthalten ist.

Nachfolgend wird diese Semantik anhand von Beispielen genauer erläutert.

INCLUDE_AND-Semantik: Das Konstrukt “INCLUDE_AND” verknüpft verschiedene Elemente konjunktiv miteinander, was bedeutet, dass alle gelisteten Elemente gleichzeitig erfüllt sein müssen (vgl. Quellcode 5-54). Es wird formalisiert durch die Boolesche Formel (element_1) ∧ (element_2).

```
INCLUDE_AND:
    - element_1
    - element_2
```

Quellcode 5-54: INCLUDE_AND Semantik

Zur Verdeutlichung des Einsatzes von INCLUDE_AND wird ein Beispiel betrachtet (vgl. Quellcode 5-55). Dieses Modul enthält eine einzelne INCLUDE_AND-Sektion. Diese wird als "wahr" bewertet, wenn die Belegung der CODs eine Windgeschwindigkeit von weniger als 40 km/h und eine Niederschlagsrate von weniger als 20 mm/h vorweist.

```
MODULES:
   beispiel_modul_1:
      TYPE: dsm
      INCLUDE_AND:
         windgeschwindigkeit: < 40 km/h
         regenrate: < 20 mm/h
```

Quellcode 5-55: Beispielmodul INCLUDE_AND Semantik

Um die Auswertung des Moduls zu veranschaulichen, werden beispielhafte CODs zum passenden Kontext betrachtet (vgl. Tabelle 5-6).

Tabelle 5-6: Beispielhafte COD für die“ INCLUDE_AND“ Semantik

#	Temporal Extent	Spatial Extent	Windgeschwindigkeit [km/h]	Regenrate [mm/h]
1	2024-06-01 08:12:53:784	*48.0232 11.7153*	10.1	2.31
2	2024-06-01 08:13:53:784	*48.0232 11.7153*	50.7	3.65
3	2024-06-01 08:14:53:784	*48.0232 11.7153*	21.3	23.29

Die Interpretation der Belegung führt dabei zu folgenden Wahrheitswerten (vgl. Tabelle 5-7):

- COD #01 liegt innerhalb der definierten ODD, da sowohl die Windgeschwindigkeit als auch die Niederschlagsrate die Bedingungen erfüllen.
- COD #02 und COD #03 liegen außerhalb der ODD, weil jeweils mindestens eine der Bedingungen nicht erfüllt wird.

Tabelle 5-7: Wahrheitstablle für die „INCLUDE_AND“ Semantik

Situation	P: Windgeschwindigkeit < 40 km/h	Q: Regenrate < 20 mm/h	Gesamtaussage (P ∧ Q)
COD #01	True	True	True
COD #02	False	True	False
COD #03	True	False	False

INCLUDE_OR-Semantik: Das Konzept „INCLUDE_OR“ verbindet verschiedene Elemente auf eine disjunktive Weise, sodass die Erfüllung eines der Elemente genügt, um die Anforderungen zu erfüllen (vgl. Quellcode 5-56). Die Boolesche Darstellung dieses Konzepts lautet (element_1) ∨ (element_2).

```
INCLUDE_OR:
    - element_1
    - element_2
```

Quellcode 5-56: INCLUDE_OR-Semantik

Zur Demonstration der Verwendung von INCLUDE_OR wird ein konkretes Beispiel analysiert (vgl. Quellcode 5-57). In diesem Fall wird spezifiziert, das entweder die Windgeschwindigkeit unter 40 km/h oder die Regenrate unter 20 mm/h liegen muss, damit die Bedingung als "wahr" ausgewertet wird.

```
MODULES:
   beispiel_modul_2:
      TYPE: dsm
      INCLUDE_OR:
         windgeschwindigkeit: < 40 km/h
         regenrate: < 20 mm/h
```

Quellcode 5-57: Beispielmodul INCLUDE_OR-Semantik

Um die Interpretation zu veranschaulichen, werden beispielhafte COD-Daten mit entsprechendem Kontext betrachtet (siehe Tabelle 5-6). Für die resultierenden Auswertungen für die beispielhaften CODs ergibt sich (vgl. Tabelle 5-8):

- COD #01 liegt innerhalb der ODD, da sowohl die Bedingungen für Windgeschwindigkeit < 40 km/h als auch für Regenrate < 20 mm/h erfüllt sind.
- COD #02 liegt innerhalb der ODD, weil die Regenrate < 20 mm/h ist.
- COD #03 liegt innerhalb der ODD, da die Windgeschwindigkeit < 40 km/h ist.

Tabelle 5-8: Wahrheitstabelle für die INCLUDE_OR-Semantik

Situation	P: Windgeschwindigkeit < 40 km/h	Q: Regenrate < 20 mm/h	Gesamtaussage (P ∨ Q)
COD #01	True	True	True
COD #02	False	True	True
COD #03	True	False	True

EXCLUDE_OR-Semantik: Das Konstrukt „EXCLUDE_OR“ verbindet verschiedene Elemente in einer disjunktiven Ausschlussbeziehung. Das bedeutet, dass die Anwesenheit eines der aufgelisteten Elemente zur Nichterfüllung der Gesamtbedingung führt (vgl. Quellcode 5-58). Es wird formalisiert durch die Boolesche Formel ¬((element_1) ∨ (element_2)).

```
EXCLUDE_OR:
   - element_1
   - element_2
```

Quellcode 5-58: EXCLUDE_OR-Semantik

Um zu demonstrieren, wie EXCLUDE_OR angewendet wird, wird ein Beispiel betrachtet. In diesem ist das Modul so spezifiziert, dass es als "false" bewertet wird, wenn die Windgeschwindigkeit unter 40 km/h oder die Niederschlagsrate unter 20 mm/h liegt.

```
MODULES:
  beispiel_modul_3:
    TYPE: dsm
    EXCLUDE_OR:
      windgeschwindigkeit: < 40 km/h
      regenrate: < 20 mm/h
```

Quellcode 5-59: Beispielmodul EXCLUDE_OR-Semantik

Zur Verdeutlichung der Auswertung werden beispielhafte COD-Daten im entsprechenden Kontext betrachtet (vgl. Tabelle 5-6). Die Interpretation dieser Daten führt zu folgenden Wahrheitswerten (vgl. Tabelle 5-9):

- COD #01: Windgeschwindigkeit = 10.1 km/h, Regenrate = 2.31 mm/h. Da beide Werte unter den Schwellenwerten liegen, ist die Bedingung nach EXCLUDE_OR "false".
- COD #02: Windgeschwindigkeit = 50.7 km/h, Regenrate = 3.65 mm/h. Trotz höherer Windgeschwindigkeit ist die Niederschlagsrate unter dem Limit. Dies führt ebenfalls zu einem „false“.
- COD #03: Windgeschwindigkeit = 21.3 km/h, Regenrate = 23.29 mm/h. Da die Regenrate den Grenzwert überschreitet, ist das Ergebnis nach EXCLUDE_OR "false".

Tabelle 5-9: Wahrheitstabelle für die EXCLUDE_OR-Semantik

Situation	P: Windgeschwindigkeit < 40 km/h	Q: Regenrate < 20 mm/h	Gesamtaussage ¬ (P ∨ Q)
COD #01	True	True	¬ (True) = False
COD #02	False	True	¬ (True) = False
COD #03	True	False	¬ (True) = False

EXCLUDE_AND-Semantik: Das Konzept „EXCLUDE_AND“ verbindet verschiedene Elemente in der Form, dass deren gleichzeitiges Auftreten konjunktiv ausgeschlossen wird (vgl. Quellcode 5-60). Diese Verbindung wird durch die Boolesche Formel ¬((element_1) ∧ (element_2)) formalisiert. Der Ansatz dient dazu, zu gewährleisten, dass eine spezifische Kombination von Elementen nicht gleichzeitig vorhanden ist.

```
EXCLUDE_AND :
   - element_1
   - element_2
```

Quellcode 5-60: EXCLUDE_AND-Semantik

Um die Verwendung von EXCLUDE_AND zu verdeutlichen, wird ein Beispiel betrachtet. Dieses spezifiziert in einem Modul, das die gleichzeitige Erfüllung der Bedingungen „windgeschwindigkeit“ unter 40 km/h und „regenrate“ unter 20 mm/h ausgeschlossen werden soll (vgl. Quellcode 5-61).

```
MODULES:
  beispiel_modul_4:
    TYPE: DSM
    EXCLUDE_AND:
      windgeschwindigkeit: < 40 km/h
      regenrate: < 20 mm/h
```

Quellcode 5-61: Beispielmodul EXCLUDE_AND-Semantik

Die semantische Auswertung wird unter Zuhilfenahme der beispielhaften CODs verdeutlicht (vgl. Tabelle 5-6). Die Interpretation führt zu folgenden Wahrheitswerten (vgl. Tabelle 5-10):

- COD #01 liegt außerhalb der ODD, da sowohl die Windgeschwindigkeit als auch die Regenrate unter den festgelegten Grenzwerten liegen.
- COD #02 liegt innerhalb der ODD, da die Windgeschwindigkeit den Grenzwert von 40 km/h überschreitet.
- COD #03 liegt ebenfalls innerhalb der ODD, da die Regenrate den Grenzwert von 20 mm/h überschreitet.

Tabelle 5-10: Wahrheitstabelle für die EXCLUDE_AND-Semantik

Situation	P: Windgeschwindigkeit < 40 km/h	Q: Regenrate < 20 mm/h	Gesamtaussage ¬ (P ∧ Q)
COD #01	True	True	¬ (True) = False
COD #02	False	True	¬ (False) = True
COD #03	True	False	¬ (False) = True

INCLUDE_AND mit EXCLUDE_OR-Semantik: Die Verwendung zweier Schlüsselwörter ermöglicht die genauere Definition von Bedingungen. Dazu können beispielsweise die Schlüsselwörter INCLUDE_AND und EXCLUDE_OR kombiniert werden. Das nachfolgende Beispiel verdeutlicht dies anhand der Definition von Umweltbedingungen (vgl. Quellcode 5-62).

```
MODULES:
  beispiel_modul_5:
    TYPE: dsm
    INCLUDE_AND:
      windgeschwindigkeit: < 40 km/h
      regenrate: < 20 mm/h
    EXCLUDE_OR:
      sichtweite_nebel: < 50 m
      bandbreite: < 1 Mbps
```

Quellcode 5-62: Beispielmodul INCLUDE_AND mit EXCLUDE_OR-Semantik

Das Modul wird als "wahr" bewertet, wenn folgende Bedingungen gleichzeitig erfüllt sind:

- Die Windgeschwindigkeit liegt unter 40 km/h und die Regenrate ist weniger als 20 mm/h.
- und die Sichtweite im Nebel ist nicht unter 50 m oder die Verbindungsbandbreite ist nicht unter 1 Mbps.

Dazu werden fünf CODs betrachtet, um das Modul anhand beispielhafter Daten auszuwerten (vgl. Tabelle 5-11).

Tabelle 5-11: Beispielhafte CODs für kombinierte Semantik

#	Temporal Extent	Spatial Extent	Windgeschwindigkeit [km/h]	Regenrate [mm/h]	Sichtweite [m]	Bandbreite [Mbps]
1	2024-06-01 08:12:53:784	*48.0232 11.7153*	10.1	2.31	100	7.1
2	2024-06-01 08:13:53:784	*48.0232 11.7153*	50.7	3.65	90	4.2
3	2024-06-01 08:14:53:784	*48.0232 11.7153*	21.3	23.29	80	2.8
	2024-06-01 08:15:53:784	*48.0232 11.7153*	21.3	23.29	40	1.9
	2024-06-01 08:16:53:784	*48.0232 11.7153*	21.3	23.29	60	0.6

Die Interpretation führt damit zu folgenden Wahrheitswerten (vgl. Tabelle 5-12):

- COD #01 liegt innerhalb der festgelegten ODD, da alle Werte innerhalb der festgelegten Grenzen liegen.
- COD #02 liegt außerhalb der ODD, weil die Windgeschwindigkeit 40 km/h übersteigt.
- COD #03 liegt außerhalb der ODD, da die Regenrate 20 mm/h übersteigt.
- COD #04 liegt außerhalb der ODD, weil die Nebelsichtweite unter 50 m fällt.
- COD #05 liegt außerhalb der ODD, da die Verbindungsbandbreite unter 1 Mbps liegt.

Tabelle 5-12: Wahrheitstabelle kombinierte Semantik 1

Situation	P: Wind	Q: Regen	R: Nebel	S: Bandbreite	(P∧Q)∧¬(R VS)
COD #01	True	True	False	False	True
COD #02	False	True	False	False	False
COD #03	True	False	False	False	False
COD #04	True	False	True	False	False
COD #05	True	False	False	True	False

INCLUDE_OR mit EXCLUDE_AND-Semantik: Das Modul verfügt sowohl über einen INCLUDE_OR- als auch einen EXCLUDE_AND-Abschnitt (vgl. Quellcode 5-63). Dieses wird als "wahr" bewertet, wenn die folgenden Bedingungen erfüllt sind:

- Die Windgeschwindigkeit liegt unter 40 km/h oder die Niederschlagsrate ist weniger als 20 mm/h.
- und die Sichtweite bei Nebel beträgt mindestens 40 m oder die Verbindungsbandbreite ist gleich oder größer als 1 Mbps.

```
MODULES:
  beispiel_modul_6:
    TYPE: dsm
    INCLUDE_OR:
      windgeschwindigkeit: < 40 km/h
      regenrate: < 20 mm/h
    EXCLUDE_AND:
      sichtweite_nebel: < 50 m
      bandbreite: < 1 Mbps
```

Quellcode 5-63: Beispielmodul INCLUDE_OR mit EXCLUDE_AND-Semantik

Für den weiteren Vergleich werden dieselben fünf CODs des vorherigen Beispiels verwendet (vgl. Tabelle 5-11). Die Interpretation führt dann zu folgenden Wahrheitswerten (vgl. Tabelle 5-13):

- COD #01 liegt innerhalb der ODD, da alle Werte mindestens den definierten Schwellenwerten entsprechen.
- COD #02 liegt innerhalb der ODD, da trotz einer Windgeschwindigkeit über 40 km/h die Niederschlagsrate unter 20 mm/h liegt. Zudem liegen auch die Nebelsichtweite und die Verbindungsbandbreite über den erforderlichen Mindestwerten.
- COD #03 liegt innerhalb der ODD, da die Windgeschwindigkeit unter 50 km/h liegt und sowohl die Nebelsichtweite als auch die Verbindungsbandbreite über den Mindestwerten sind.
- COD #04 liegt außerhalb der ODD, da die Sichtweite zu gering ist.
- COD #05 liegt außerhalb der ODD, da die Verbindungsbrandbreite zu gering ist.

Tabelle 5-13: Wahrheitstabelle kombinierte Semantik 2

Situation	P: Wind	Q: Regen	R: Nebel	S: Bandbreite	(PVQ) ∧¬(R ∧ S)
COD #01	True	True	False	False	True
COD #02	False	True	False	False	True
COD #03	True	False	False	False	True
COD #04	True	False	True	False	False
COD #05	True	False	False	True	False

INCLUDE_AND mit OR-Unterbedingungen: Die Struktur verknüpft mehrere Elemente konjunktiv und integriert eine disjunktive Unterbedingung (vgl. Quellcode 5-64). Formal kann dies als Boolesche Gleichung ausgedrückt werden: (element_1) ∧ (element_2) ∧ (subcondition_3.1 ∨ subcondition_3.2). Dieser Ausdruck stellt sicher, dass alle Hauptelemente präsent sind und gleichzeitig mindestens eine der spezifizierten Unterbedingungen erfüllt ist.

```
INCLUDE_AND:
   - element_1
   - element_2
   - OR:
        - subcondition_3.1
        - subcondition_3.2
```

Quellcode 5-64: INCLUDE_AND mit OR-Unterbedingungen Semantik

Zur Verdeutlichung wird ein Beispiel betrachtet. In diesem Beispiel wird das Modul in einem Kontext verwendet, in dem die Downlink-Latenz weniger als 10 ms und der Downlink-Durchsatz mehr als 1 Mbps betragen muss. Zusätzlich muss für die Positionierung entweder GPS oder ein lokaler Beacon verfügbar sein. Ist all dies erfüllt, wird das Modul gesamtheitlich als "true" bewertet.

```
MODULES:
  beispiel_modul_7:
    TYPE: dsm
    INCLUDE_AND:
      latenz: < 10 ms
      download_durchsatz: > 1 Mbps
      OR:
        global_positioning_GPS: true
        lokal_beacon: true
```

Quellcode 5-65: Beispielmodul INCLUDE_AND mit OR-Unterbedingungen

Um die Auswertung des Moduls zu veranschaulichen, werden beispielhafte CODs betrachtet (vgl. Tabelle 5-14).

Tabelle 5-14: Beispielhafte CODs für verschachtelte Aussagen

ID	Temporal Extent	Geographic Extent	Latenz [ms]	Durchsatz_ [mbps]	GPS	Beacon
#01	2024-06-01 08:12:53.784	48.0232 11.7153	11	2	True	False
#02	2024-06-01 08:13:53.784	48.0232 11.7153	13	1	False	True
#03	2024-06-01 08:14:53.784	48.0232 11.7153	14	3	False	False
#04	2024-06-01 08:15:53.784	48.0232 11.7153	9	0.5	False	True

Die Interpretation führt zu folgenden Wahrheitswerten (vgl. Tabelle 5-15):

- COD #01 liegt innerhalb der ODD, da der Download-Durchsatz über 1 Mbps liegt und GPS aktiv ist.
- COD #02 liegt außerhalb der ODD, da der Durchsatz nicht mehr als 1 Mbps beträgt.
- COD #03 liegt außerhalb der ODD, da weder GPS noch ein Beacon verfügbar sind.
- COD #04 liegt außerhalb der ODD, weil der Durchsatz weniger als 1 Mbps beträgt.

Tabelle 5-15: Wahrheitstabelle für „INCLUDE_AND“ mit „OR“ Semantik

Situation	P: Latenz	Q: Durchsatz	R: GPS	S: Beacon	P ∧ Q ∧ (R ∨ S)
COD #01	True	True	True	False	True
COD #02	True	False	False	True	False
COD #03	True	True	False	False	False
COD #04	False	False	False	True	False

EXCLUDE_OR mit AND-Unterbedingungen: Das Konstrukt „EXCLUDE_OR mit AND“ kombiniert das Ausschließen mehrerer Hauptelemente disjunktiv mit einer konjunktiven Unterbedingung. Dies bedeutet, dass keines der Hauptelemente zutreffen darf und zusätzlich die beiden Unterbedingungen nicht gleichzeitig erfüllt sein dürfen (vgl. Quellcode 5-66). Formal wird dies durch die Boolesche Gleichung ¬((element_1) ∨ (element_2) ∨ (subcondition_3.1 ∧ subcondition_3.2)) ausgedrückt.

```
MODULES:
  EXCLUDE_OR:
```

```
    - element_1
    - element_2
     AND:
        - subcondition_3.1
        - subcondition_3.2
```

Quellcode 5-66: EXCLUDE_OR mit AND-Unterbedingungen Semantik

Anhand eines Beispiels wird dies verdeutlicht. In diesem dürfen weder die Latenz unter 10 ms fallen noch der Download-Durchsatz über 1 Mbps liegen. Zusätzlich dürfen GPS und Beacon nicht gleichzeitig vorhanden sein.

```
MODULES:
  beispiel_module_8:
    TYPE: dsm
    EXCLUDE_OR:
      latenz: < 10 ms
      download_durchsatz: > 1 Mbps
      AND:
        global_GPS: true
        lokal_beacon: true
```

Quellcode 5-67: Beispielmodul EXCLUDE_OR mit AND-Unterbedingungen

Zur Verdeutlichung der Auswertung werden die beispielhafte COD-Daten im entsprechenden Kontext betrachtet (vgl. Tabelle 5-14). Die Interpretation führt zu folgenden Wahrheitswerten (vgl. Tabelle 16):

- COD #01 liegt außerhalb der ODD, weil sowohl die Latenz als auch der Download-Durchsatz unzulässig sind (beide überschreiten die zulässigen Werte).
- COD #02 liegt außerhalb der ODD, da die Latenz unzulässig ist.
- COD #03 liegt außerhalb der ODD, weil sowohl die Latenz als auch der Download-Durchsatz die zulässigen Werte überschreiten.
- COD #04 liegt innerhalb der ODD, da keine der Hauptbedingungen zutrifft und GPS und Beacon nicht gleichzeitig verfügbar sind.

Tabelle 5-16: Wahrheitstabelle für „EXCLUDE_OR“ mit „AND“ Semantik

Situation	P: Latenz	Q: Durchsatz	R: GPS	S: Beacon	¬((P) ∨ (Q) ∨ (R ∧ S))
COD #01	True	True	True	False	False
COD #02	True	False	False	True	False
COD #03	True	True	False	False	False
COD #04	False	False	True	False	True

INCLUDE_OR mit AND-Unterbedingungen – Verknüpft mehrere Elemente disjunktiv und integriert eine konjunktive Unterbedingung. Dies kann als Boolesche Gleichung folgendermaßen dargestellt werden: (element_1) ∨ (element_2) ∨ (subcondition_3.1 ∧ subcondition_3.2). Diese Konfiguration wird genutzt, um festzulegen, dass das Vorhandensein eines der Hauptelemente oder die gleichzeitige Erfüllung beider Unterbedingungen ausreichend ist.

```
INCLUDE_OR:
      - element_1
      - element_2
       AND:
          - subcondition_3.1
          - subcondition_3.2
```

Quellcode 5-68: INCLUDE_OR mit AND-Unterbedingungen Semantik

In diesem Beispiel wird das Modul als wahr bewertet, wenn entweder die Downlink-Latenz unter 10 ms liegt, der Downlink-Durchsatz mehr als 1 Mbps beträgt oder sowohl GPS als auch Beacon gleichzeitig verfügbar sind (vgl. Codeblock 5-69).

```
MODULES:
  beispiel_modul_9:
     TYPE: dsm
     INCLUDE_OR:
        latenz: < 10 ms
        download_durchsatz: > 1 Mbps
        AND:
           global_GPS: true
           lokal_beacon: true
```

Quellcode 5-69: Beispielmodul INCLUDE_OR mit AND-Unterbedingungen

Zur Verdeutlichung der Auswertung werden die beispielhafte COD-Daten im neuen Kontext betrachtet (vgl. Tabelle 5-14) Die Interpretation davon führt zu folgenden Ergebnissen und folgender Wahrheitstabelle (vgl. Tabelle 5-17):

- COD #01 liegt innerhalb der ODD, da sowohl die Latenz als auch der Download-Durchsatz die Bedingungen erfüllen.
- COD #02 liegt innerhalb der ODD, da die Latenz unter dem Schwellenwert von 10 ms liegt.
- COD #03 liegt innerhalb der ODD, da sowohl die Latenz als auch der Download-Durchsatz die Bedingung erfüllen.
- COD #04 liegt außerhalb der ODD, da GPS und Beacon nicht gleichzeitig verfügbar sind und auch die Latenz und der Durchsatz als falsch bewertet sind.

Tabelle 5-17: Wahrheitstabelle für „INCLUDE_OR“ mit „AND“ Semantik

Situation	P: Latenz	Q: Durchsatz	R: GPS	S: Beacon	P ∨ Q ∨ (R ∧ S)
COD #01	True	True	True	False	True
COD #02	True	False	False	True	True
COD #03	True	True	False	False	True
COD #04	False	False	True	False	False

EXCLUDE_AND mit OR-Unterbedingungen – Schließt verschiedene Elemente konjunktiv aus und integriert eine disjunktive Unterbedingung. Als boolesche Gleichung hat dies die Form: ¬((element_1) ∧ (element_2) ∧ (subcondition_3.1 ∨ subcondition_3.2)). Diese Struktur wird genutzt, um auszuschließen, dass alle Hauptelemente gleichzeitig erfüllt sind und mindestens eine der Unterbedingungen präsent ist.

```
EXCLUDE_AND:
      - element_1
      - element_2
       OR:
          - subcondition_3.1
          - subcondition_3.2
```

Quellcode 5-70: EXCLUDE_AND mit OR-Unterbedingungen Semantik

Anhand eines Beispiels kann dies verdeutlicht werden: In diesem dürfen die Downlink-Latenz nicht unter 10 ms und der Downlink-Durchsatz nicht über 1 Mbps gleichzeitig liegen. Zudem darf nicht mindestens eine der Unterbedingungen – entweder GPS oder Beacon – erfüllt sein. Dies stellt sicher, dass weder beide Hauptelemente noch eine der Unterbedingungen gemeinsam zutreffen dürfen.

```
MODULES:
  beispiel_modul_10:
     TYPE: dsm
     EXCLUDE_AND:
       latenz: < 10 ms
       download_durchsatz: > 1 Mbps
       OR:
         global _GPS: true
         lokal_beacon: true
```

Quellcode 5-71: Beispielmodul EXCLUDE_AND mit OR-Unterbedingungen

Zur Veranschaulichung der Auswertung werden die beispielhaften COD-Daten im neu beschriebenen Kontext herangezogen (vgl. Tabelle 5-14). Die Interpretation dieser Daten führt zu den folgenden Ergebnissen und der dazugehörigen Wahrheitstabelle (vgl. Tabelle 5-18):

- COD #01 liegt außerhalb der ODD, da sowohl die Latenz als auch der Download-Durchsatz gleichzeitig die Bedingungen erfüllen. Zusätzlich ist GPS verfügbar (eine der Unterbedingungen ist erfüllt).
- COD #02 liegt innerhalb der ODD, da nicht alle Hauptelemente gleichzeitig erfüllt sind und die Bedingung für den Download-Durchsatz nicht verletzt wird.
- COD #03 liegt innerhalb der ODD, weil nicht alle Hauptelemente gleichzeitig zutreffen, da die Unterbedingungen nicht gleichzeitig erfüllt sind.
- COD #04 liegt innerhalb der ODD, da keine der Hauptelemente zutrifft und eine Unterbedingung verletzt wird.

Tabelle 5-18: Wahrheitstabelle für „EXCLUDE_AND“ mit „OR“ Semantik

Situation	P: Latenz	Q: Durchsatz	R: GPS	S: Beacon	¬(P ∧ Q ∧ (R ∨ S))
COD #01	True	True	True	False	False
COD #02	True	False	False	True	True
COD #03	True	True	False	False	True
COD #04	False	False	True	False	True

5.5.4.5. Kombination und Auswertung von Modulen

Die Kombination und Wiederverwendung von Modulen spielen eine zentrale Rolle in der Modellierungssprache. Nachfolgend wird ein Überblick gegeben, wie Module kombiniert und

gemeinsam ausgewertet werden können (vgl. Quellcode 5-72). Das Beispiel definiert dafür mehrere Module, die unter folgenden Bedingungen als wahr bewertet werden:

- „modul_2“ wird als wahr bewertet, wenn die Geschwindigkeit über 50 km/h liegt.
- „modul_3“ wird als wahr bewertet, wenn ein spezifisches Verkehrszeichen sichtbar ist.
- „modul_4“ wird als wahr bewertet, wenn der Straßentyp eine Einfahrt ist.
- „modul_1“ hängt von den Wahrheitswerten anderer Module ab:
    - Es wird als wahr bewertet, wenn der Straßentyp „*stadtstraße_lokal*“ oder „*stadtstraße_zubringer*“ ist und die Module „modul_2“ (Geschwindigkeit über 50 km/h) und „modul_3“ (sichtbares Verkehrszeichen) als wahr bewertet werden.
    - Zudem dürfen die Geschwindigkeit nicht größer als 80 sein und die Module „modul_4“ (Straßentyp = Einfahrt) und „modul_5“ müssen wahr sein.
- „modul_5“ hat eigene spezifische Bedingungen:
    - Es wird als wahr bewertet, wenn der Straßentyp „indoor“ ist und
    - keine Überschwemmungen vorhanden sind.

```
…
MODULES:
  modul_1:
    TYPE: dsm
    INCLUDE_AND:
      straßentyp:
        - stadtstraße_lokal
        - stadtstraße_zubringer
      modul_2
      modul_3
    EXCLUDE_OR:
      max_geschwindigkeit: > 80 kmh
      modul_4
      modul_5

  modul_2:
    TYPE: dsm
    INCLUDE_AND:
      max_geschwindigkeit: >50 km/h

  modul_3:
    TYPE: dsm
    INCLUDE_AND:
      schild_feature: sichtbar

  modul_4:
    TYPE: dsm
    INCLUDE_AND:
      straßentyp: auffahrt

  modul_5:
    TYPE: dsm
    INCLUDE_AND:
```

```
      straßentyp: indoor
    EXCLUDE_OR:
      dynamische_zone:
        - Überschwemmung
```

Quellcode 5-72: Beispielmodule für die Kombination und Auswertung

Diese Modularität ermöglicht die Kombination verschiedener Module unter variierenden Bedingungen, um komplexere Regeln oder Betriebsbedingungen zu modellieren. Die Möglichkeit zur Wiederverwendung reduziert zudem den Arbeitsaufwand, da bestehende Regeln nicht neu formuliert, sondern einfach referenziert werden können.

5.5.4.6. Disjunktives Referenzieren von Modulen mit Labeln

Labels ermöglichen das disjunktive Referenzieren mehrerer Module durch ein einzelnes Label. Dies ist besonders effektiv, wenn eine große Anzahl von Modulen existiert und alle Module berücksichtigt werden sollen, die einen spezifischen Bereich ansprechen (z. B. Wetter). Label können entweder mit Hilfe von INCLUDE_AND oder EXCLUDE_OR eingebunden werden.

- **Inklusion mit Label (INCLUDE_AND)**: Hierbei werden mehrere Label kombiniert, wobei jedes Label verschiedene Module repräsentieren kann. Die Bedingung wird erfüllt, wenn alle spezifizierten Label als wahr ausgewertet werden.
- **Exklusion mit Label (EXCLUDE_OR):** Label können auch verwendet werden, um Bedingungen auszuschließen. Wenn eines der Label wahr ist, wird die übergeordnete Bedingung falsch.

Nachfolgend wird aufgezeigt, wie Label das disjunktive Referenzieren mehrerer Module durch ein einzelnes Label ermöglichen. Zur Verdeutlichung werden die Spezifikationen von Labeln und Modulen gegenübergestellt (vgl. Tabelle 5-19).

Tabelle 5-19. Gegenüberstellung von Labeln und Modulen

	Geschrieben als Label	Geschrieben mit Modulen
INCLUDE_AND	INCLUDE_AND: - label_A - label_B	INCLUDE_AND: - (modul_1A OR modul_2A) - (modul_1B OR modul_2B)
Boolesche Formalisierung	label_A ∧ label_B	(modul_1A ∨ modul_2A) ∧ (modul_1B ∨ modul_2B)
EXCLUDE_OR	EXCLUDE_OR: - label_A - label_B	EXCLUDE_OR: - OR (modul_1A OR modul_2A) - OR (modul_1B OR modul_2B)
Boolesche Formalisierung	¬(label_A ∨ label_B)	¬(modul_1A ∨ modul_2A ∨ modul_1B ∨ modul_2B)

Anhand eines Beispiels wird dies weiter erläutert (vgl. Quellcode 5-73). „module_1“ und „modul_2“ werden im „hauptmodul“ über das Label „wetter_label“ referenziert. In diesem Beispiel

wäre das ADS innerhalb der ODD, wenn die Regenmenge unter 7.7 mm/h liegt oder wenn der Regen nicht gefriert.

```
…
  hauptmodul:
    TYPE: ucm
    INCLUDE_AND:
      - wetter_label

  modul_1:
    TYPE: dsm
    LABELS:
      - wetter_label
    INCLUDE_AND:
      regenmenge: < 7.7

  modul_2:
    TYPE: dsm
    LABELS:
      - wetter_label
    INCLUDE_AND:
     eisregen: false
```

Quellcode 5-73: Beispielmodul für die Label Semantik

Die Verwendung von Labels ermöglicht es, die Komplexität zu reduzieren und die Modularität sowie die Wiederverwendung von Codes zu fördern. Dies gelingt, indem Bedingungen effektiv unter einem gemeinsamen Label gruppiert werden.

5.5.4.7. Zusammenstellung der ODD

Die ODD ist das Kernstück der Definition von Betriebsbedingungen für ADS und besteht aus einer Reihe von Modulen und Labeln. Die Struktur und Logik der ODD basieren auf den gleichen Prinzipien, die auch für die Auswertung einzelner Module oder Label verwendet werden. Jedes Modul oder Label innerhalb einer ODD repräsentiert bestimmte Bedingungen, unter der das ADS operieren darf oder eben nicht. Diese Elemente werden einzeln ausgewertet, indem jeder Bedingung ein Wahrheitswert zugewiesen wird. Die einzelnen Wahrheitswerte der Module und Labels werden anschließend gemäß der in der ODD definierten Logik miteinander verknüpft und es wird eine Gesamtaussage gebildet. Diese bestimmt, ob die Gesamtkonfiguration der ODD als „true“ oder „false“ ausgewertet wird. Die Gesamtbewertung der ODD entscheidet darüber, ob sich das ADS innerhalb der definierten Betriebsbedingungen befindet:

- False: Wird die ODD als falsch ausgewertet, bedeutet dies, dass mindestens eine der notwendigen Bedingungen für den sicheren Betrieb des ADS nicht erfüllt ist. In diesem Fall befindet sich das Fahrzeug außerhalb der zulässigen ODD.
- True: Wird die ODD als wahr bewertet, sind alle erforderlichen Bedingungen für den Betrieb erfüllt.

Auch wenn die genutzte Semantik der ODD sich nicht von der der Module unterscheidet, wird zur weiteren Verdeutlichung ein Beispiel gewählt (vgl. Quellcode 5-74). Dieses inkludiert das „module1“ sowie das „label1“. Zudem wird das „module2“ exkludiert. Die zugehörige Boolesche

Formalisierung entspricht dem Bekannten und lässt sich wie folgt ableiten: (module1 ∧ label1) ∧ ¬module2.

```
…
ODD:
   odd_handle:
     INCLUDE_AND:
       modul1
       label1
     EXCLUDE_OR:
       modul2
```

Quellcode 5-74: ODD-Semantik

6. Modellierungspatterns

In Analogie zu Entwurfsmustern in der Softwareentwicklung, spielen Modellierungsmuster eine entscheidende Rolle, um eine geordnete Entwicklung und vorhersehbare Ergebnisse zu gewährleisten. Dieses Kapitel präsentiert nützliche Modellierungsmuster, die die Entwicklung einer modularen Spezifikation der ODD vereinfachen. Die Verwendung dieser Muster trägt dazu bei, die Komplexität zu reduzieren und eine kohärente Struktur innerhalb der ODD zu schaffen. Indem spezifische, wiederholbare Muster in der Modellierung angewendet werden, kann die Entwicklung sowohl standardisiert als auch optimiert werden. Nachfolgend wird auf eine Auswahl von Modellierungsmustern eingegangen.

6.1. Dekomposition anhand von Use Cases

Das Muster der "Use Case Dekomposition" ist ein fundamentales Prinzip zur Strukturierung der Spezifikationen innerhalb einer ODD in der Entwicklung ADS. Im Gegensatz zur Integration sämtlicher Anwendungsfälle in einem einzigen Modul, ermöglicht dieses Muster die Trennung und spezifische Aufgliederung in einzelne, klar definierte Use Case-Module. Jedes Modul adressiert einen einzelnen und unabhängigen Anwendungsfall, was eine bessere Übersicht und Handhabbarkeit gewährleistet.

Die Funktionsweise dieses Musters kann anhand einer ODD-Spezifikation verdeutlicht werden (vgl. Quellcode 6-1). Die Strukturierung erfolgt über eine INCLUDE_OR-Sektion innerhalb der ODD, welche eine Liste der unterstützten Anwendungsfälle definiert und disjunktiv verlinkt.

```
…
ODD:
  beispiel_ODD_v1.01.23:
    INCLUDE_OR:
      ucm_passagier_pickup
      ucm_stadtstraße
      ucm_autobahn
      ucm_passagier_dropoff
      ucm_parken

…
MODULES:
  ucm_passagier_pickup:
    ...
  ucm_stadtstraße:
    ...
```

Quellcode 6-1: Dekompensation von UCM-Modulen

In dieser Struktur wird jeder Anwendungsfall, wie beispielsweise der „ucm_passagier_pickup“, in einem eigenen UCM definiert. Die semantische Interpretation dieser Module innerhalb der ODD gestaltet sich dann folgendermaßen:

- Eine COD wird als zulässig in Bezug auf die ODD angesehen, wenn diese einem der in der INCLUDE_OR-Sektion aufgeführten Use Case-Module entspricht.
- Eine COD wird ausgeschlossen, wenn diese von keinem der aufgelisteten Module als zulässig betrachtet wird.

6.2. Dekomposition anhand der Taxonomie

Ein weiteres Dekompositionsmuster besteht darin, die innerhalb der Bedingungen verwendeten Konzepte auf einen Teilbereich der Taxonomie zu beschränken. Solche Einschränkungen vermeiden eine unübersichtliche Kombinatorik, die mit der Spezifizierung und Bewertung von Bedingungen verbunden sein kann. Als Beispiel kann eine Taxonomie betrachtet werden, die Unterbäume für Wetter, Szenerie und dynamische Umgebungen spezifiziert. Eine solche Hierarchie kann dann genutzt werden, um Bedingungen zu dekomponieren. Dazu werden alle wetterbezogenen Bedingungen in einem Wettermodul, alle szeneriebezogenen Bedingungen in einem Szeneriemodul und alle Bedingungen für dynamische Umgebungen in einem dynamischen Modul gruppiert. Diese Art der Dekomposition kann rekursiv angewendet werden. Beispielsweise können die Szeneriemodule gemäß akzeptablen Straßenbedingungen und Kreuzungsbedingungen weiter unterteilt werden. Dieses Muster nutzt für die Erstellung INCLUDE_AND“-Abschnitte. Zur weiteren Verdeutlichung wird ein Beispiel genutzt, das folgende Module definiert (vgl. Quellcode 6-2):

- Passenger Pickup: Stellt ein UCM dar und enthält zwei DSM für „zulässiges_wetter“ und „zulässige_szenerie“. Die beiden DSM orientieren sich hierbei an dem Aufbau einer Taxonomie, der diese Konzepte getrennt beschreibt.
- DSM „zulässiges_wetter“: Dieses Modul beinhaltet Bedingungen wie Licht, Niederschlag und Wind, die jeweils in eigenen Untermodulen spezifiziert sind.
- DSM „zulässige_szenerie“: Dieses Modul wird weiter in die taxonomiebasierten Unterbaummodule für akzeptable Straßenbedingungen und akzeptable Kreuzungsbedingungen zerlegt.

Die Semantik dieser Dekomposition basiert auf einer UND-Verknüpfung. Komplexere Dekompositionen können eine AND-OR-Baumstruktur zur Definition von Bedingungen verwenden.

```
1  TAXONOMIE:
2    wetter:
3      helligkeit:
4        ...
5      regen:
6        ...
7      wind:
8        ...
9  MODULES:
10   passagier_pickup:
11       TYPE: ucm
12       INCLUDE_AND:
13         zulässiges_wetter
14         zulässige_szenerie
15     ...
16
17   zulässiges_wetter:
18       TYPE: dsm
19      INCLUDE_AND:
20         zulässige_helligkeit
21         zulässiger_regen
22         zulässiger_wind
23     ...
```

```
24
25   zulässige_szenerie:
26       TYPE: dsm
27       INCLUDE_AND:
28         zulässige_straßenbedingungen
29         zulässige_kreuzungsbedingungen
30       ...
```

Quellcode 6-2: Veranschaulichung taxonomiebasierte Dekomposition

Das Beispiel zeigt auf, das nur CODs als zulässig gelten, die zwei grundlegende Kriterien erfüllen. Zum einen müssen akzeptable Wetterbedingungen vorliegen. Zum anderen muss eine geeignete Verkehrsinfrastruktur gegeben sein. Die UND-Verknüpfung gewährleistet hierbei, dass alle akzeptierten Situationen den festgelegten Anforderungen in Bezug auf Wetterbedingungen und Infrastrukturmerkmale genügen. Dieses Beispiel zeigt, wie durch die Nutzung einer taxonomiebasierten Baumdekomposition die Komplexität bei der Verwaltung von Bedingungen in verschiedenen Umweltdomänen effektiv reduziert und die Modularität sowie Wiederverwendbarkeit innerhalb der Systemarchitektur verbessert werden kann.

6.3. Bedingtes Inkludieren und Exkludieren von Bedingungen

Bei der Konzeption von Modulen ist die Definition von bedingten Betriebsbedingungen ebenfalls wichtig. Ein Beispiel beschreibt die Einbeziehung ländlicher Straßen unter spezifischen Bedingungen, welche wiederum von der Art der Fahrbahnmarkierung und der zugehörigen Geschwindigkeitsbegrenzung abhängen. Auf dieser Basis wird definiert, dass ländliche Straßen nur dann Teil der ODD sind, wenn bei durchgezogener Fahrbahnmarkierung eine Geschwindigkeit von über 70 km/h vorliegt. Sofern die Geschwindigkeit unter 70 km/h liegt, sind andere Kriterien ausschlaggebend. Diese differenzierte Betrachtung erfordert die Modellierung mittels einer disjunktiven INCLUDE_OR-Bedingung. Diese ermöglicht es, verschiedene Szenarien für Fahrbahnmarkierungen zu berücksichtigen und eine flexible Anpassung der ODD an die gegebenen Verkehrssituationen vorzunehmen.

Anhand der entwickelten Sprache kann dieses Beispiel wie folgt umgesetzt werden (vgl. Quellcode 6-3):

- Das Modul „fähigkeiten_landstraße“ schließt ländliche Straßen nur dann ein, wenn spezifische Bedingungen bezüglich der Fahrbahnmarkierung erfüllt sind und mindestens eine der Unterbedingungen erfüllt wird. Diese sind in anderen Modulen definiert und werden über ein OR eingebunden. Für die Unterbedingungen reicht es aus, wenn eine der beiden spezifischen Bedingungen für Fahrbahnmarkierungen erfüllt wird, damit die COD als zulässig gilt.
- Die erste Unterbedingung stellt das Modul *„fahrspur_typ_durchgezogen_geschwindigkeitslimit*“ dar, welches für Hochgeschwindigkeitssituationen eine durchgezogene Fahrbahnmarkierung fordert.
- Die zweite Unterbedingung *„fahrspur_typ_alle_geschwindigkeitslimit*“ gilt für Niedriggeschwindigkeitssituationen und ist unabhängig von einer Fahrbahnmarkierung.

```
1  …
2  MODULES:
3    fähigkeiten_landstraße:
4      TYPE: dsm
5      INCLUDE_AND:
6        straßentyp: landstraße
```

```
      OR:
        fahrspur_typ_durchgezogen_geschwindigkeitslimit:
        fahrspur_typ_alle_geschwindigkeitslimit:

  fahrspur_typ_durchgezogen_geschwindigkeitslimit:
    TYPE: dsm
    INCLUDE_AND:
      spur_markierung: durchgezogen
      erlaubte_geschwindigkeit: > 70 km/h

  fahrspur_typ_alle_geschwindigkeitslimit:
    TYPE: dsm
    INCLUDE_AND:
      erlaubte_geschwindigkeit: < 70 km/h
```

Quellcode 6-3: Bedingtes Inkludieren

6.4. Nutzung von Label

In der Entwicklung von ADS ist es essenziell, Moduldefinitionen so zu gestalten, dass sie fortlaufend erweitert oder verändert werden können. Dies gewährleistet, dass jederzeit zusätzliche Situationen einbezogen oder ausgeschlossen werden können, ohne dass explizite Änderungen an bereits festgelegten Bedingungen notwendig werden. Das Muster der erweiterbaren Label Disjunktion unterstützt dieses Vorgehen und bietet eine flexible Strukturierung der ODD. Im Folgenden soll dies anhand eines Beispiels verdeutlicht werden, indem eine Liste von gefährlichen Umweltbedingungen betrachtet wird. Anfänglich könnte diese Liste auf große Regentropfen beschränkt sein, die die Effektivität von Kamerasystemen signifikant beeinträchtigen. Im Laufe der Zeit können weitere Bedingungen, wie beispielsweise vereiste Straßenoberflächen, hinzugefügt werden. Nachfolgend wird aufgezeigt, wie Bedingungen hinzugefügt werden können, ohne bereits vorhandene Definitionen zu ändern (vgl. Quellcode 6-4).

```
MODULES:
  passagier_pickup:
    TYPE: ucm
    INCLUDE_AND:
      zulässige_pickup_locations
      ...
    EXCLUDE_OR:
      gefährliche_bedingungen

  pickup_locations_gruppe1:
    TYPE: dsm
    LABELS:
      - zulässige_pickup_locations
    INCLUDE_AND:
      straßenbereich:
        - main_st_sec1
        - main_st_sec2

  pickup_locations_gruppe2:
    TYPE: dsm
```

```
    LABELS:
      - zulässige_pickup_locations
    INCLUDE_AND:
      bahnhof:
        - pole5
.       - pole11
    ...

  zu_viel_regen:
    TYPE: dsm
    LABELS:
      - gefährliche_bedingungen
    INCLUDE_AND:
      regenrate: heavy_rain
    ...

  vereiste_straße:
    TYPE: dsm
    LABELS:
      - gefährliche_bedingungen
    INCLUDE_AND:
      straßenoberfläche: glatteis
    ...
```

Quellcode 6-4: Nutzung von Labeln

Das UCM definiert, das eine COD dann zulässig ist, wenn die Bedingungen für „zulässige_pickup_locations“ erfüllt sind und keine „gefährliche_bedingungen“ vorliegen. Das Label „zulässige_pickup_locations“ wiederum wird als erfüllt angesehen, wenn mindestens eines der zugeordneten Module als wahr bewertet wird. Diese Struktur ermöglicht es, dass weitere Module unter diesem Label gruppiert werden können, ohne dass Änderungen an bereits bestehenden Modulen erforderlich sind. Gleichermaßen gilt dies auch für das Label „gefährliche_bedingungen“. Dieses Label wird als wahr betrachtet, wenn Bedingungen wie starker Regen oder Glatteis, die in Modulen wie „zu_viel_regen“ oder „vereiste_straße“ definiert sind, zutreffen. Auch hier ermöglicht die modulare Struktur das einfache Hinzufügen weiterer Bedingungen, ohne die bestehenden Definitionen zu modifizieren.

Diese Modellierungsstrategie bietet signifikante Flexibilität und erlaubt es, die ODD fortlaufend an veränderte Umstände anzupassen. Die Nutzung von erweiterbaren Labels trägt wesentlich dazu bei, das System aktuell und anpassungsfähig zu halten, ohne die Grundstruktur jedes Mal neu gestalten zu müssen.

6.5. Implizites Inkludieren

In der Modellierung komplexer Systeme kann die explizite Auflistung aller akzeptablen Bedingungen unpraktikabel sein. Stattdessen kann es mehr Sinn machen, einen Ansatz zu verfolgen, bei dem nur die nicht akzeptable Bedingungen explizit ausgeschlossen werden. Dies führt dazu, dass implizit alle anderen Bedingungen als akzeptabel angesehen werden. Für dieses Vorgehen wird ein konkretes Beispiel betrachtet. In diesem ist es erforderlich zu spezifizieren, dass alle Regenarten, mit der Ausnahme von „torroidal_regen“ akzeptabel sind (vgl. Quellcode 6-5).

```
…
MODULES:
  zulässiges_wetter:
    TYPE: dsm
    EXCLUDE_OR:
      regentyp:
        - torroidal_regen
```

Quellcode 6-5: Implizites Inkludieren

Durch den Einsatz der EXCLUDE_OR-Anweisung zur Angabe des ausgeschlossenen Elements, werden alle anderen Regenarten, die in der Taxonomie als weitere Kategorien hinterlegt sind, implizit als akzeptabel eingestuft. Dadurch entfällt die Notwendigkeit, jede einzelne akzeptable Bedingung explizit zu definieren.

6.6. Implizites Exkludieren

Andersherum kann es bei der Definition von Bedingungen auch von Vorteil sein, nicht jede einzelne ausgeschlossene Bedingung explizit zu spezifizieren. Vielmehr ist es oft zielführender, eine selektive Anzahl von akzeptierten Bedingungen zu definieren. Anhand dessen kann dann davon ausgegangen werden, dass alle anderen Zustände implizit ausgeschlossen sind. Ein gutes Beispiel hierfür ist die Spezifikation von Wetterbedingungen. Für diese soll gelten, dass ausschließlich die Abwesenheit von Regen akzeptabel ist, ohne jedoch jede mögliche Form von Niederschlag explizit benennen zu müssen. Diese Vorgehensweise lässt sich durch folgendermaßen abbilden (vgl. Quellcode 6-6).

```
MODULES:
  zulässiges_wetter:
    TYPE: dsm
    INCLUDE_OR:
      regenintensitätslevel:
        - kein_regen
```

Quellcode 6-6: Implizites Exkludieren

Durch die Verwendung der INCLUDE_OR-Klausel zur Kennzeichnung des akzeptablen Zustands „kein_regen“, werden alle anderen Formen des Regens implizit als unakzeptabel betrachtet. Dies reduziert die Notwendigkeit, alle denkbaren unerwünschten Regenstufen explizit aufzuführen, was wiederum die Komplexität verringert und die Übersicht verbessert.

6.7. Verwaltung unzulässiger Betriebsbedingungen

Eine weitere Herausforderung stellt die effiziente Verwaltung von unzulässigen Betriebsbedingungen dar. Ein effektiver Ansatz hierfür ist die Einbindung von Modulen, die wiederverwendbare Ausschlüsse repräsentieren. Dafür wird das Modul "schlechtes_wetter " betrachtet, das spezifisch dafür entwickelt wurde, verschiedene Formen von schlechtem Wetter zu erfassen (vgl. Quellcode 6-7).

```
MODULES:
  mein_use_case_modul:
    TYPE: ucm
    INCLUDE_OR:
      ...
    EXCLUDE_OR:
      schlechtes_wetter:

```

```
  schlechtes_wetter:
    TYPE: dsm
    EXCLUDE_OR:
      regenintensitätslevel: starker_regen
      windgeschwindigkeit: >= 5- km/h
      ...
```

Quellcode 6-7: Unzulässige Betriebsbedingungen

Diese beinhalten beispielsweise starken Regen, hohe Windgeschwindigkeiten und dichten Nebel. Jede dieser Bedingungen reicht aus, um eine Situation von der ODD auszuschließen. Im übergeordneten Modul "mein_use_case_modul" wird das DSM "schlechtes_wetter" exkludiert. Die Logik dazu ist folgendermaßen definiert:

- Das Modul "schlechtes_wetter" wird als „wahr“ bewertet, wenn mindestens eine der spezifizierten Bedingungen zutrifft.
- Das Use Case Modul "mein_use_case_modul" wird nur dann als „wahr“ bewertet, wenn das "schlechtes_wetter"-Modul als falsch bewertet wird. Dies bedeutet, dass keine der im "schlechtes_wetter"-Modul definierten Bedingungen erfüllt sein darf.

6.8. Dekomposition von konfliktären Bedingungen

Die konsistente Weiterentwicklung von Modulen zur Berücksichtigung neuer Bedingungen ist von entscheidender Bedeutung. Ein prägnantes Beispiel hierfür ist die Evolution eines Moduls zur Spezifikation zulässiger Betriebsbedingungen eines Autobahn-Piloten. Bei diesem wurden über mehrere Versionen hinweg, zusätzliche Spezifikationen integriert. In der initialen Version „taxi_v1“ wurden grundlegende Bedingungen für den Betrieb auf Autobahnen definiert. Diese umfassen (vgl. Quellcode 6-8):

```
ODD:
  taxi_v1:
    INCLUDE_OR:
      ucm_autobahnpilot
MODULES:
  ucm_autobahnpilot:
    TYPE: ucm
    INCLUDE_AND:
      dsm_autobahnpilot_szenerie

  dsm_autobahnpilot_szenerie:
    TYPE: dsm
    INCLUDE_AND:
      straßentyp: autobahn
      fahrbahnmarkierung_farbe: weiß
```

Quellcode 6-8: Initiale ODD taxi_v1

Die ODD „taxi_2“ erweitert das ursprüngliche Modul um Baustellen, was die Komplexität der Modellspezifikationen erhöht (vgl. Quellcode 6-9).

```
ODD:
  taxi_v2:
    INCLUDE_AND:
      ucm_autobahnpilot
MODULES:
  ucm_autobahnpilot:
    TYPE: ucm
    INCLUDE_AND:
      dsm_autobahnpilot_szenerie
      dsm_autobahnpilot_baustelle

  dsm_autobahnpilot_szenerie:
    TYPE: dsm
    INCLUDE_AND:
      straßentyp: autobahn
      fahrbahnmarkierung_farbe: weiß

  dsm_autobahn_baustelle:
    TYPE: dsm
    INCLUDE_AND:
      straßentyp: autobahn
      fahrbahnmarkierung_farbe: gelb

```

Quellcode 6-9: Weiterentwickelte ODD taxi_v2

Jedoch führt die Einführung von Baustellenbedingungen zu einem Konflikt. Dies gilt insbesondere dann, wenn die Straßenmarkierungsfarbe Gelb als Indikator für Baustellen verwendet wird. Dies könnte fälschlicherweise Situationen als zulässig deklarieren, obwohl die Bedingungen für eine Baustelle vorliegen. Dies liegt darin begründet, dass die Straßenmarkierungsfarbe ein gemeinsames Attribut (weiß für Autobahnszenerie, gelb für Baustellen) der Szenerie und der Baustelle ist, und sich lediglich anhand der Enumeration unterscheidet. Für die Vermeidung solcher Konflikte gibt es zwei mögliche Lösungen. Die erste Lösung zielt darauf ab, die konfliktreiche Bedingung zu ändern, indem beispielsweise die „fahrbahnmarkierung_farbe“ durch „schild“ oder andere Indikatoren für Baustellen ersetzt wird (vgl. Quellcode 6-10).

```
  dsm_autobahn_baustelle:
    TYPE: dsm
    INCLUDE_AND:
      straßentyp: autobahn
      schild: baustelle
```

Quellcode 6-10:Modifiziertes Baustellenmodul

Die zweite Lösung beschreibt das Refactoring. Dadurch wird das Modul so geändert, dass lediglich auf spezifische Autobahn-Baustelle verweisen wird und nicht jede Art von Baustelle „angesprochen“ wird. Zur Umsetzung dieses Vorgehens wird ein INCLUDE_OR im Modul „ucm_autobahnpilot“ verwendet, um entweder das DSM „dsm_autobahn“ oder das DSM „dsm_autobahn_baustelle“ zu inkludieren (vgl. Quellcode 6-11).

```
…
ODD:
```

```
  taxi_v1.02.04:
    INCLUDE_AND:
      ucm_autobahnpilot

MODULES:
  ucm_autobahnpilot:
    TYPE: ucm
    INCLUDE_OR:
      dsm_autobahn
      dsm_autobahn_baustelle

  dsm_autobahn:
    TYPE: dsm
    INCLUDE_AND:
      straßentyp: autobahn
      fahrbahnmarkierung_farbe: weiß

  dsm_autobahn_baustelle:
    TYPE: dsm
    INCLUDE_AND:
      straßentyp: autobahn
      dsm_generische_baustelle

  dsm_generische_baustelle:
    TYPE: dsm
    INCLUDE_AND:
      fahrbahnmarkierung_farbe: gelb
```

Quellcode 6-11:Refactoring der ODD

Im Detail sieht der Ablauf des Refactoring wie folgt aus (vgl. Abbildung 6-1):

- Identifizieren der generischen Elemente, die nicht spezifisch für den Anwendungsfall sind (im obigen Beispiel die gelbe Fahrbahnmarkierung).
- Aufbau des Use Case-Moduls, indem die relevanten DSM mit einer logischen OR-Verknüpfung kombiniert und Elemente, die nicht Teil des Anwendungsfalls sind, ausgeschlossen werden.

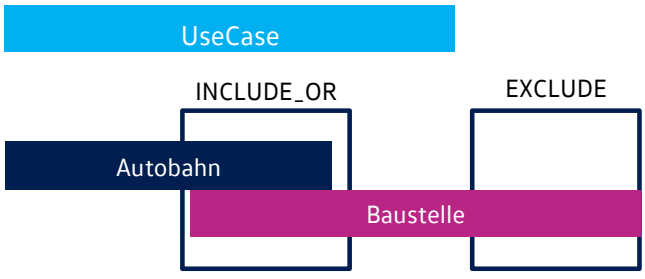

Abbildung 6.1: Visualisierung des Refactorings

6.9. Quantifizierung von Unsicherheiten

Die Quantifizierung von Unsicherheiten ist oft notwendig, um sichere Fahrbedingungen festzulegen. Beispielsweise muss spezifiziert werden, dass der Autobahnpilot nur sicher ist, wenn das Vorhandensein von Radfahrern äußerst unwahrscheinlich ist. Im Gegensatz dazu sind Straßen, auf denen Radfahrer häufig vorkommen, zu riskant. In dem spezifischen Beispiel des Moduls „ads_v1“, wird das DSM „zulässiges_risiko“ verwendet, um sicherheitskritische Bedingungen zu definieren (vgl. Abbildung 6-12). Dazu werden verschiedene Unsicherheiten mit der Hilfe von Messgrößen quantifiziert und so bewertbar gemacht (vgl. Kapitel 5.3.1.6).

```
…
MODULES:
  ads_v1:
    TYPE: ucm
    INCLUDE_AND:
      zulässiges_risiko

  zulässiges_risiko:
    TYPE: dsm
    INCLUDE_OR:
      schneedecke.höhe: <2cm
      starker_regen.dauer: < 1 hr
    EXCLUDE_OR:
      radfahrer.auftretenshäufigkeit: > 1 pro_tag
      fußgänger.auftretenshäufigkeit: > 2 pro_stunde
```

Quellcode 6-12: Modellierung von Unsicherheiten

Das Modul „zulässiges_risiko“ wird als erfüllt betrachtet, wenn die Höhe der Schneedecke gering ist und die Dauer des starken Regens weniger als eine Stunde beträgt. Zusätzlich wird das Modul als unzulässig bewertet, wenn die Auftrittsrate von Radfahrern höher als einmal pro Tag ist oder die Auftrittsrate von Fußgängern zweimal pro Stunde übersteigt. Diese Bedingungen sind darauf ausgerichtet, die Sicherheit durch die Identifikation und Vermeidung von Risikobereichen zu maximieren.

7. Werkzeugunterstützung

Dieses Kapitel widmet sich der Herausforderung, eine ODD sowohl qualitativ als auch quantitativ basierend auf einer Vielfalt von Inputdaten, zu bewerten. Die ODD definiert die spezifischen Bedingungen, unter denen ein ADS betrieben werden darf. Diese Bewertungen sind entscheidend, um sicherzustellen, dass das ADS unter allen vorgesehenen Umgebungsbedingungen zuverlässig funktioniert. Dabei umfasst diese Bewertung sowohl einfache Ja/Nein-Entscheidungen — beispielsweise, ob ein bestimmtes Szenario innerhalb der Grenzen der ODD liegt —, als auch komplexere Beurteilungen. Zu Letzterer gehört unter anderem die Analyse der Abdeckung durch verschiedene Szenarien und Testfälle oder die Bestimmung zulässiger Straßennetzwerke (vgl. Kapitel 3). Diese Anforderungen machen es notwendig, große Datenmengen aus unterschiedlichen Quellen (Kartendaten, Wetterdaten, Realfahrtdaten, …) zu integrieren und zu verarbeiten. Das entwickelte Lösungskonzept basiert daher auf dem Abgleich der ODD mit den CODs, die die spezifischen Situationen beschreiben, die eine ODD abdecken sollte. Durch den systematischen Vergleich von ODD und CODs wird es möglich, die Übereinstimmung und Vollständigkeit der ODD zu bewerten und so zu einer fundierten Einschätzung der Funktionalität und Sicherheit des ADS zu gelangen. Darüber hinaus wird nicht nur der Abgleich des vorgesehenen Betriebsbereiches des ADS mit der ODD unterstützt, sondern es werden auch relevante Daten für die weitere Entwicklung des Systems geliefert. Dies schafft eine gemeinsame Wissensbasis für alle relevanten Stakeholder in Form des geforderten Single Point of Konwledge (vgl. Kapitel 3.8). Maßgeblich für dieses Kapitel ist die Fragestellung, wie dieses definierte Lösungskonzept toolgestützt umgesetzt werden kann.

Dazu wird auf zwei verschiedene Realisierungskonzepte eingegangen. Zuerst wird in Kapitel 7.1 die initiale Lösung, mit dem der Proof of Concept entwickelt wurde beschrieben. Im Anschluss daran wird in Kapitel 7.2 eine Umsetzung mit einem Standard-Toolset aufgezeigt und so Übertragbarkeit und Skalierbarkeit des Lösungskonzeptes demonstriert.

7.1. Nutzung von Elastic Search

Die erste toolseitige Realisierung basiert auf standardisierten Komponenten von Elastic Search [GT15] und einer Datenstruktur aus inversen Indizes (vgl. Abbildung 7-1). Ein inverser Index ist eine Datenstruktur, die jedem Begriff aus dem Dokumentenkorpus eine Liste von Dokumenten zuordnet, in denen dieser Begriff vorkommt. Dieser Mechanismus ermöglicht eine schnelle Suche nach Dokumenten, die bestimmte Wörter oder Phrasen enthalten. Dies bildet die Grundlage der Funktionsweise von Elastic Search und wird beispielsweise genutzt, um JSON-Dokumente basierend auf ihrem Übereinstimmungsgrad mit einer Suchanfrage zu bewerten. Wie in Abbildung 7-1 dargestellt, resultiert jede Suchanfrage in einer nach Relevanz sortierten Liste von Dokumenten. Diese Listen sind das Ergebnis des Abgleichs von Suchbegriffen (aus der Anfrage) mit den im inversen Index verzeichneten Begriffen.

Eine Erweiterung von Elastic Search ist der Percolator. Dieser kehrt die übliche Suchrichtung um, indem, statt mit einer Anfrage nach Dokumenten zu suchen, auf Basis eines Dokument zutreffende Suchanfragen gesucht werden. Diese Funktionalität ist normalerweise besonders dann nützlich, wenn eine Echtzeit-Verarbeitung erforderlich ist (z.B., das Überwachen von eingehenden Dokumenten gegen voreingestellte Kriterien.) Zusammengefasst verwendet der Percolator den inversen Index, um alle Anfragen zu identifizieren, die mit den im Dokument vorhandenen Begriffen übereinstimmen. [GT15]

Übertragen auf die vorliegende Arbeit unterstützt Elastic Search zwei Schlüssel-Use-Cases:

- Finden von CODs, die mit Modulen übereinstimmen: In diesem Use-Case werden die Indizes mit verschiedenen realen oder hypothetischen CODs gefüllt. Module, anhand dessen die CODs

bewertet werden sollen, werden als Suchanfragen genutzt. Elastic Search sucht dann mit Hilfe der inversen Indizes nach allen CODs, die diesen Modulen entsprechen. Als Ergebnis wird eine sortierte Liste von übereinstimmenden CODs geliefert.

- Finden von Modulen, die durch CODs unterstützt oder verletzt werden: Hier werden die Indizes stattdessen mit Modulen gefüllt und CODs als Eingaben für den Percolator verwendet. Dies ermöglicht die Überprüfung, welche Module durch die gegebenen CODs unterstützt oder verletzt werden. Die Suchmaschine generiert als Ergebnis eine Liste von Modulen, die sortiert ist, nach der Stärke der Unterstützung oder Verletzung durch die jeweilige COD. [SRR22]

Eine einzelne Instanz von Elastic Search, die auf einem lokalen Rechner betrieben wird, kann solche Ranking-Operationen typischerweise innerhalb von 10-20 Millisekunden abschließen. [SRR22] Diese schnelle Verarbeitungszeit macht Elastic Search zu einem effizienten Werkzeug für zahlreiche Anwendungen, die schnelle und präzise Such- und Abgleichfunktionalitäten benötigen.

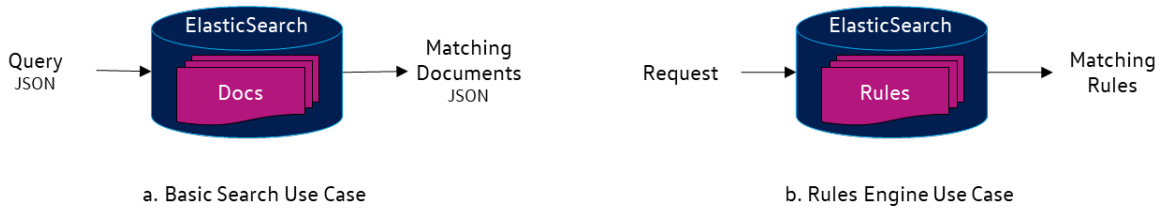

Abbildung 7.1: Adaption von Elastic Search, nach [SRR22]

Zusammengefasst bietet Elastic Search eine leistungsstarke Plattform für die schnelle und genaue Verarbeitung großer Datenmengen. Die Einbindung des Percolators erweitert die Funktionalität, indem sie die Durchführung von Echtzeit-Datenanalysen und die dynamische Überprüfung von Modulen ermöglicht. Diese Technologie ist daher nicht nur ein Werkzeug für einfache Suchaufgaben, sondern auch eine robuste Lösung für komplexe Datenmengen. Die Anwendung von Elastic Search ist dabei maßgeblich hilfreich für Use-Cases, die sich auf verschiedene Aspekte der Datenanalyse und Compliance-Überprüfung konzentrieren. Dazu gehören:

- Evaluierung der Compliance beobachteter Verhaltensweisen: Der Prozess beginnt mit der Extraktion von Schlüssel-Wert-Paaren aus Fahrprotokollen, die dann zu interpretierten Verhaltenszusammenfassungen aggregiert werden. Die Evaluierung der Compliance erfolgt auf der Ebene einzelner Module, indem geprüft wird, inwieweit die beobachteten Verhaltensweisen mit den definierten Bedingungen der Module übereinstimmen. Dieser Prozess kann genutzt werden, um eine Gesamteinschätzung der Compliance zu erhalten (vgl. auch Kapitel 8.2).
- Bestimmung der Abdeckung von Modulen: Dieser Anwendungsfall bewertet, in welchem Umfang alle beobachteten CODs durch eine vorgegebene Anzahl von Modulen abgedeckt werden. Diese methodische Herangehensweise ähnelt der Abdeckungsbewertung in traditionellen Testszenarien und zielt darauf ab, die Vollständigkeit der Module in Bezug auf die erfassten CODs zu verifizieren.
- Bewertung des akzeptablen Risikos: Hierbei wird analysiert, ob die beobachteten Verhaltensweisen ein akzeptables Risiko darstellen. Diese Analyse ist ein integraler Bestandteil der HARA und wird häufig bspw. für den Sicherheitsnachweis der SOTIF benötigt. Für diesen Fall unterstützt Elastic Search die Aggregation und Analyse großer Datenmengen, um eine fundierte Bewertung des Risikos zu ermöglichen (vgl. Kapitel 8.2).

7.2. Übertragbarkeit auf SMT Solver

Neben der Realisierung mit Elastic Search, lassen sich auch etablierte Werkzeugunterstützungen nutzen. Um dies zu demonstrieren, wird nachfolgend die Übertragbarkeit des Lösungskonzeptes auf SMT-LIB aufgezeigt.

Um eine effiziente Überprüfung von ODD und COD zu gewährleisten, ist es unerlässlich, dass die Spezifikationen von automatisierten Verifizierungswerkzeugen verarbeitet werden können. Diese Werkzeuge ermöglichen dann die Durchführung von Syntaxprüfungen und Konsistenzkontrollen der ODD-Spezifikationen sowie die Überprüfung der CODs gegen alle ODD-Beschränkungen. Die Methode, zur Umsetzung dieser Anforderungen ist in zwei Ebenen unterteilt und gliedert sich in mehrere Schritte. Die erste Schicht beschreibt das Spezifikationslayer und die zweite das Verifikationslayer (vgl. Abbildung 7-2). [ASK+23] Beide werden im Folgenden detaillierter beschrieben.

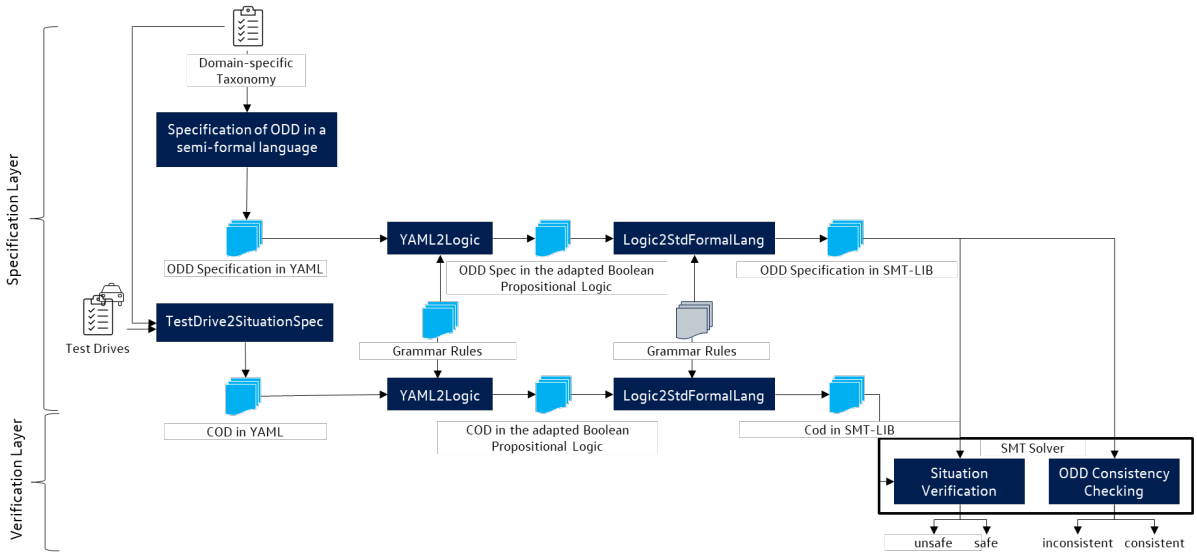

Abbildung 7.2: Methode zur Übertragbarkeit auf den SMT-Solver, nach [ASK+23]

Spezifikationslayer - Das Spezifikationslayer beginnt mit der manuellen Beschreibung der ODD in einer semi-formalen Sprache. Auf Basis einer Taxonomie definieren hier Experten spezifische Module und die Beziehungen zwischen diesen, die für das zu entwickelnde ADS relevant sind. Das Ergebnis dieses Prozesses ist eine ODD-Spezifikation im YAML-Format (vgl. Kapitel 5.5). [ASK+23]

```
supported_parking_lot_conditions:
  MODULES:
    module_id:
      TYPE: dsm
      INCLUDE_AND:
        parking_lot_length: > 12 m
        is_curve: true
```

Quellcode 7-1: Extrahierte ODD-Spezifikation i

Im zweiten Schritt werden die Spezifikationen der CODs extrahiert. Die Extraktion erfolgt durch ein automatisiertes Werkzeug, wobei das Konzept und die detaillierte Verarbeitung nicht Bestandteil dieser Arbeit sind. (vgl. Quellcode 7-2) Das Ergebnis sind COD-Spezifikationen im YAML-Format (vgl. Kapitel 5.4). [ASK+23]

```
parking_lot_length: = 13
is_curve: true
```

Quellcode 7-2: A Extrahierte COD-Spezifikation im YAML-Format.

Im dritten Schritt erfolgt die Übersetzung der Spezifikation aus einer YAML-basierten semi-formalen Sprache in eine angepasste Form der Booleschen Aussagenlogik. Um diese Übersetzung zu ermöglichen, wurde eine Grammatik für die angepasste Boolesche Aussagenlogik definiert. Diese definiert eine Sprache, die den Aufbau und die Referenzierung von Modulen in einer eingeschränkten Aussagenlogik ermöglicht (vgl. Quellcode 7-3). Die eingeschränkte Aussagenlogik gestattet die Formulierung von logischen Aussagen mit den binären logischen Operatoren &(UND) und |(ODER) sowie dem unären Operator !(NICHT). Diese Logik wird weiter angepasst, um die Formulierung von ODD-Aussagen mit arithmetischen Operationen wie < und ≤ sowie > und ≥ zu ermöglichen. [ASK+23]

```
Module ::= ModuleIdentifier ":" Expression
Expression ::= TruthValue | Variable |
LogicalExpr | "("XRefExpr")" |
ArithExpression
LogicalExpr ::= XRefExpr BinOp XRefExpr |
UnOp XRefExpr
ArithExpression ::= Variable ArithOp Numerical
XRefExpr ::= ModuleIdentifier | Expression
```

Quellcode 7-3: Grammatik für die angepasste Boolesche Aussagenlogik

Die Grammatik für die angepasste Boolesche Aussagenlogik umfasst spezifische Produktionsregeln zur Strukturierung des Aufbaus logischer und mathematischer Ausdrücke:

- Module: Werden durch einen ModulIdentifier definiert, gefolgt von einem Doppelpunkt und einer Expression (vgl. Quellcode 7-3, Zeile 1).
- Ausdrücke: Können als TruthValue, Variable, LogicalExpression oder ArithExpression auftreten (vgl. Quellcode 7-3, Zeilen 2, 3, und 4). Logische Ausdrücke nutzen binäre (z. B. UND, ODER) und unäre (NICHT) Operatoren auf XRefExpression. Arithmetische Ausdrücke nutzen dagegen Variablen mit numerischen Werten.
- XRefExpression: Grundlegende Bausteine der Grammatik, der als ModuleIdentifier oder einer anderen Expression definiert ist (vgl. Quellcode 7-3, Zeile 8). [ASK+23].

Daraus folgend, wird unter Verwendung der eingeführten Grammatikregeln ein Auszug der ODD-Spezifikation, der zunächst in einer semi-formalen Sprache formuliert ist, in die angepasste Boolesche Aussagenlogik übersetzt (vgl. Quellcode 7-4). Für die COD-Spezifikation ergibt sich kein Unterschied.

```
supported_parking_lot_conditions :
(parking_lot.length > 12 & is_curved)
```

Quellcode 7-4: ODD-Beschreibung für die angepasste Boolesche Aussagenlogik

Der vierte Schritt überführt die Spezifikation aus der angepassten Booleschen Aussagenlogik in die standardisierte formale Sprache von SMT-LIB. [ASK+23] Unter Nutzung der Grammatikregeln von SMT-LIB wird der Auszug der angepassten ODD-Spezifikation in SMT-LIB Code überführt (vgl. Quellcode 7-5).

```
(set-logic QF_LIA)
(set-option :produce-models true)

(declare-const parking_lot_length Int)
```

```
(declare-const is_curve Bool)

(assert
(and
(> parking_lot_length 12)
is_curve)
)
```

Quellcode 7-5: ODD-Spezifikation in SMT-LIB

Die Übersetzung der COD (vgl. Quellcode 7-2) von der angepassten Booleschen Aussagenlogik nach SMT-LIB beinhaltet die Transformation der Variablenzuweisungen in Behauptungen (vgl. Quellcode 7-6). Eine Behauptung ist eine spezifische Aussage in SMT-LIB, mit der überprüft wird, ob eine bestimmte Variable, den in der COD vorab aufgezeichneten Wert aufweist. [ASK+23]

```
(assert (= parking_lot_length 13))
(assert (= is_curve true))

(check-sat)
(get-model)
(exit)
```

Quellcode 7-6: COD in SMT-LIB

Verifikationslayer - Das Verifikationslayer nutzt die ODD-Spezifikation und die Spezifikation der CODs in der standardisierten formalen Sprache SMT-LIB. Anhand dieser Inputs können die Überprüfung der Konsistenz und der Situationen/CODs vorgenommen werden. Die Konsistenzprüfung der ODD überprüft, ob die ODD-Spezifikation in sich schlüssig und damit logisch erfüllbar ist. Eine ODD ist erfüllbar, wenn es möglich ist, die Spezifikation so mit Werten zu belegen, dass sie wahr wird. Diese Prüfung erfolgt mittels eines SMT-Solvers (Satisfiability Modulo Theories Solver), der die logische Formel der ODD im Kontext einer spezifischen Theorie (z. B. lineare ganzzahlige Arithmetik) analysiert. Der Solver gibt "sat" zurück, wenn eine solche Belegung gefunden wird. Dies ist gleichbedeutend mit der Erfüllung der Konsitenz. Gibt der Solver "unsat" zurück, ist die Spezifikation inkonsistent und kann in der definierten Form nicht erfüllt werden. [ASK+23] Die Situationsüberprüfung evaluiert, ob die COD-Spezifikationen innerhalb der Grenzen der ODD liegen. Hierbei wird für jede COD geprüft, ob ihre Variablenzuweisung die ODD-Spezifikation erfüllt. Diese Überprüfung dient dazu festzustellen, welche CODs zulässig oder unzulässig in Bezug auf die Spezifikation sind. [ASK+23] Diese Schritte gewährleisten, dass sowohl die interne Konsistenz der ODD überprüft wird als auch jede COD einzeln gegen die ODD validiert wird. Das Ergebnis dieser Überprüfungen bestimmt, ob die operationalen Spezifikationen für das ADS praktisch umsetzbar sind.

Um die Leistungsfähigkeit und Skalierbarkeit des SMT-Solvers zu demonstrieren, wurde ein Experiment mit dem fortschrittlichen SMT-Solver CVC5 durchgeführt. In diesem Experiment werden CODs anhand künstlicher ODD-Bedingungen überprüft. Dieses Experiment zielt darauf ab, die Anwendbarkeit des SMT-Solvers bei der Handhabung großer Datenmengen zu testen. [ASK+23] Für die Studie wurden zwei Dimensionen der Skalierbarkeit untersucht. Zum einen die Anzahl der CODs und zum anderen die Anzahl der ODD-Bedingungen. Um die Arbeitszeit des SMT-Solvers zu messen, wurden mehrere Durchläufe mit zunehmenden Mengen von CODs durchgeführt.

Für die ODD wurden zwei verschiedene Größen definiert. Die Erste umfasste sechs Variablen (gleichzusetzen mit Bedingungen), während die Zweite etwa 1000 Variablen enthielt. Für die Durchführung wurden hypothetische ODDs automatisiert erstellt. Die CODs wurden zufällig mit

Wahrheitswerten belegt, um bilden eine Vielfalt an Wertzuweisungen ab. Die Anzahl der CODs variiert zwischen 10 und 5000. In zwei Versuchsreihen wurden die CODs gegen die ODD überprüft, um die Leistung des CVC5 bei kleinen und großen Beschränkungen zu vergleichen. [ASK+23]

Die Ergebnisse zeigen die benötigte Zeit in Sekunden (y-Achse) und die Anzahl der CODs (x-Achse) für beide Versuchsreihen (vgl. Abbildung 7.3). Die violetten Punkte zeigen die Zeit für die kleinere ODD (6Variablen) und werden auf der rechten Skala abgebildet. Die türkisfarbenen Dreiecke hingegen zeigen die benötigte Rechenzeit für die größere ODD (1000 Variablen) und werden über die linke Zeitachse abgebildet. Die Ergebnisse zeigen, dass die benötigte Zeit des SMT-Solvers, um die CODs zu evaluieren, linear mit der Anzahl der CODs steigt. [ASK+23]

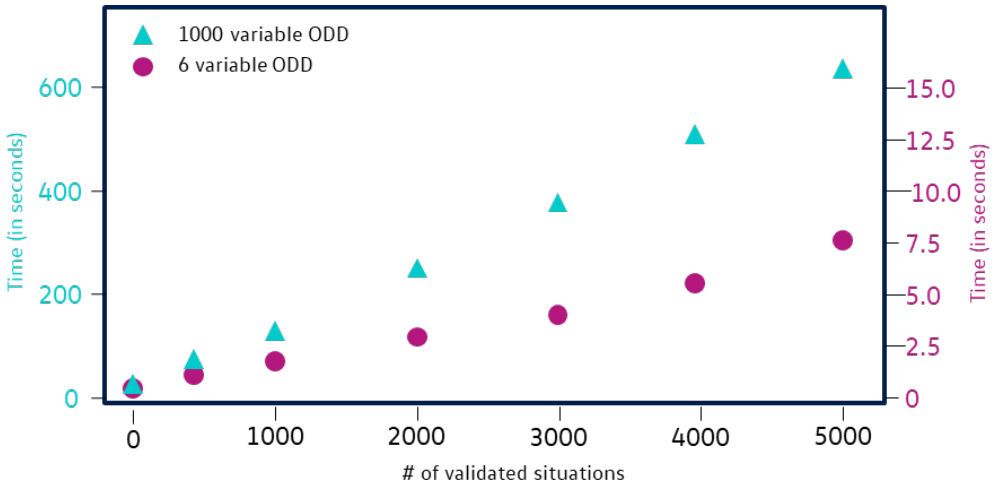

Abbildung 7.3: Laufzeitmessung für verschiedene CODS und ODD-Größen, nach [ASK+23]

Diese Erkenntnis wird bestätigt, wenn man die Ergebnisse weiter aufschlüsselt (vgl. Tabelle 7-1). Die Darstellung als Zeit/COD zeigt, dass eine lineare Verarbeitungszeit für die Überprüfung der gegebenen CODs in Relation zu den ODD-Variablen angenommen werden kann. [ASK+23] Abschließend kann die Aussage getroffen werden, dass CVC5 eine gute Skalierbarkeit aufweist und standardisierte Werkzeugunterstützungen für das entwickelte Lösungskonzept angewendet werden können. Anhand der Ergebnisse wurde dies demonstriert.

Tabelle 7-1: Verifikationszeit (in Sekunden) der Experimente, nach [ASK+23]

# CODs	Large Constraint		Small Constraint	
	Total Time	Time/#CODs	Total Time	Time/#CODs
10	1.3	0.13	0.01	0.001
20	2.81	0.1405	0.03	0.0015
50	6.84	0.1368	0.08	0.0016
100	14.17	0.1417	0.15	0.0015
500	70.59	0.1412	0.74	0.0015
1000	140.85	0.1409	1.46	0.0015
2000	266.04	0.1330	3.13	0.0016
3000	390.26	0.1301	4.21	0.0014
4000	549.77	0.1374	5.51	0.0014
5000	690.2	0.1380	7.3	0.0015

8. Ausgewählte Anwendungsfälle anhand der entwickelten formalen Sprache

Abschließend beschäftigt sich dieses Kapitel mit der dritten Forschungsfrage, der Demonstration der praktischen Nutzbarkeit der entwickelten DSL und der zugehörigen Werkzeuge anhand verschiedener Anwendungsfälle. Mit Hilfe von drei Beispielen wird gezeigt, wie die Sprache und Tools genutzt werden können, um Entwicklungs- und Testprozesse für automatisierte Fahrsysteme zu unterstützen. Dies umfasst in Kapitel 8.1 die Erstellung zulässiger Straßennetzwerke, in Kapitel 8.2 die Validierung des Fahrverhaltens sowie in Kapitel 8.3 die Relevanzbewertung von SOTIF-Szenarien. Damit wird die in der Problemanalyse als erforderlich identifizierte Prozessintegration über verschiedene Entwicklungsstufen veranschaulicht.

8.1. Erstellung von zulässigen GeoNets

Dieses Kapitel fokussiert die Evaluierung des entwickelten Konzepts zur formalen Beschreibung und Anwendung der ODD im Entwicklungskontext, indem es die Anwendbarkeit für die Bestimmung von fahrbaren Straßennetzwerken demonstriert.

Die Problemanalyse hat gezeigt, dass ein wesentlicher Bedarf besteht (vgl. Kapitel 3):

- die Straßen zu bestimmen, auf denen das ADS technisch in der Lage ist zu fahren (BCM1).
- eine nachvollziehbare Verbindung zwischen der definierten ODD und dem tatsächlichen Betriebsbereich des ADS herzustellen (REG3).
- Taxonomien für die einheitliche Kommunikation zu nutzen (REQ7).

Um die Möglichkeiten des entwickelten Konzepts in Bezug auf den identifizierten Bedarf zu evaluieren, wird ein potenzieller Betriebsbereich, repräsentiert durch CODs, mit einer formalen Beschreibung der ODD verglichen. Als Datenquelle für die COD wird OpenStreetMap (OSM) verwendet, da frei verfügbar und umfassend dokumentiert ist. Das Konzept ist jedoch nicht auf die Nutzung von OSM-Daten limitiert. Die Inferenz klassifiziert dann zulässige und unzulässige CODs, wodurch die Auswirkungen auf ein Servicenetz dargestellt und ein befahrbares Straßennetz entwickelt werden können. Eine grafische Visualisierung macht diese Ergebnisse entsprechend sichtbar. Im Folgenden wird detailliert auf die Umsetzung eingegangen.

8.1.1. Konzept zur Erstellung der GeoNets

Den Ausgangspunkt stellt die Taxonomie dar, die oftmals als Weltmodell zur Beschreibung der Systemfähigkeiten eines ADS dient und Elemente der Verkehrsinfrastruktur, dynamische Elemente und Umweltbedingungen integriert [Bri20, Sae21]. Diese Taxonomie bildet jedoch nicht nur die Grundlage für die Beschreibung der Systemfähigkeiten im Sinne der ODD, sondern auch für die COD, die in diesem Fall aus OSM [Ope24] erstellt werden. Eine Inferenzmaschine vergleicht die ODD mit den CODs, um deren Einhaltung zu überprüfen. Der Output der Inferenzmaschine wird anschließend genutzt, um ein befahrbares Straßennetz zu erzeugen.

Das Lösungskonzept wird somit durch zwei Aspekte erweitert (vgl. Abbildung 8-1):

- Erzeugung der CODs aus Inputdaten
- Erzeugung eines Straßennetzes sowie potenzieller Feature Erweiterungen aus dem Output der Inferenz.

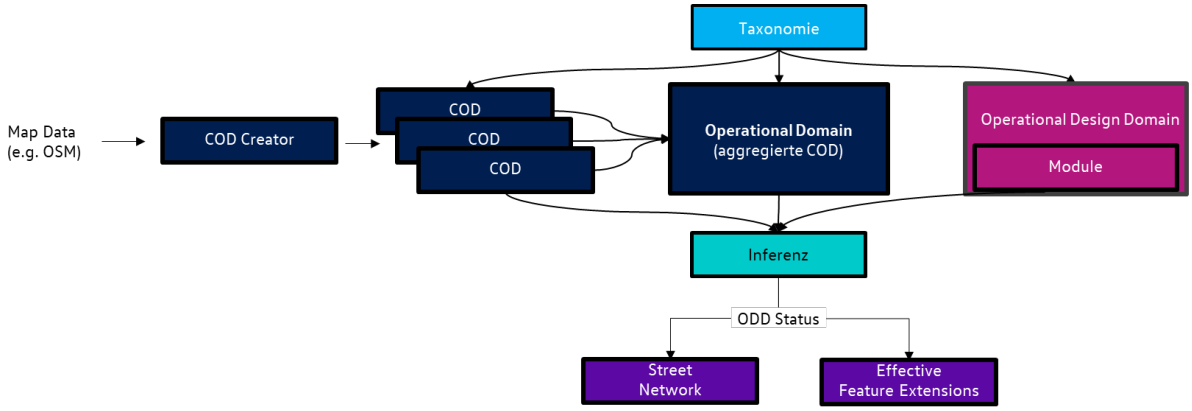

Abbildung 8.1: Erweiterung des Lösungskonzepts zur Ableitung von GeoNets

Die weitere Kapitelstruktur folgt diesem beschriebenen Prozess, beginnend mit der Taxonomie.

8.1.2. Taxonomie

Als erster Schritt ist eine gemeinsame Taxonomie erforderlich. Da es keine standardisierte und universelle Taxonomie zur Beschreibung der Umwelt gibt, ist eine Zuordnung oder Übersetzung von einer Taxonomie zur anderen erforderlich. OSM verfügt über eine eigene Taxonomie, die zur Kennzeichnung von Kartenelementen verwendet wird. Mit mehreren Tags, bestehend aus einem „key“ und einem „value“, können spezifische Umweltmerkmale dargestellt und beschrieben werden. Beispielsweise kann eine Straße auf der OSM-Karte Tags wie highway=residential, surface=asphalt oder maxspeed=50 enthalten. [Ope24]

Haben die Elemente der ODD-Taxonomie und der OSM-Taxonomie unterschiedliche Begrifflichkeiten, muss eine spezifische Zuordnung zwischen beiden Taxonomie erfolgen. Zu diesem Zweck wird eine Zuordnungstabelle erstellt, um Elemente der ODD-Taxonomie mit denen von OSM zu verbinden (vgl. Abbildung 8-2). Da Taxonomien unterschiedliche Detailstufen oder verschiedene Anwendungsgebiete haben können, ist eine direkte Zuordnung nicht immer möglich. Folgende Möglichkeiten werden unterschieden:

1. **Idealfall**: Ein exaktes Matching zwischen den Begriffen beider Taxonomien existiert.
2. **Mehrfache Zuordnung**: Mehrere Attribute der OSM-Taxonomie können einem ODD-Attribut zugeordnet werden. Eine einfache ODER-Logik kann diesen Fall lösen. Beispielsweise kann eine Baustelle in der ODD-Taxonomie mit den OSM-Attributen „building-construction“, „highway-construction“ oder „landuse-construction“ verknüpft werden.
3. **Lücken in der OSM-Taxonomie**: Wenn die ODD-Taxonomie auf eine Oberflächenbeschaffenheit aus Ziegeln verweist, die in der OSM-Taxonomie nicht existiert, müssen diese Informationen aus anderen Datenquellen bezogen werden oder es kann keine Auswertung gegen dieses Element vorgenommen werden.
4. **Implizite Informationsableitung**: Die OSM-Taxonomie hat keine Informationen über verschiedene Kreuzungstypen enthalten. Diese Informationen können jedoch implizit durch Geodatenanalysen an den Kreuzungen abgeleitet werden.
5. **Informationsdeaggregation**: Ein OSM-Attribut zahlt in mehrere ODD-Elemente ein. Zum Beispiel gibt es in der ODD-Taxonomie gerade und kurvige Lichtsignalanlagen, während in OSM nur ein generischer Lichtsignalanalagentyp beschrieben wird. Hier geht die Information verloren, ob die Lichtsignalanlage gerade oder kurvig ist.

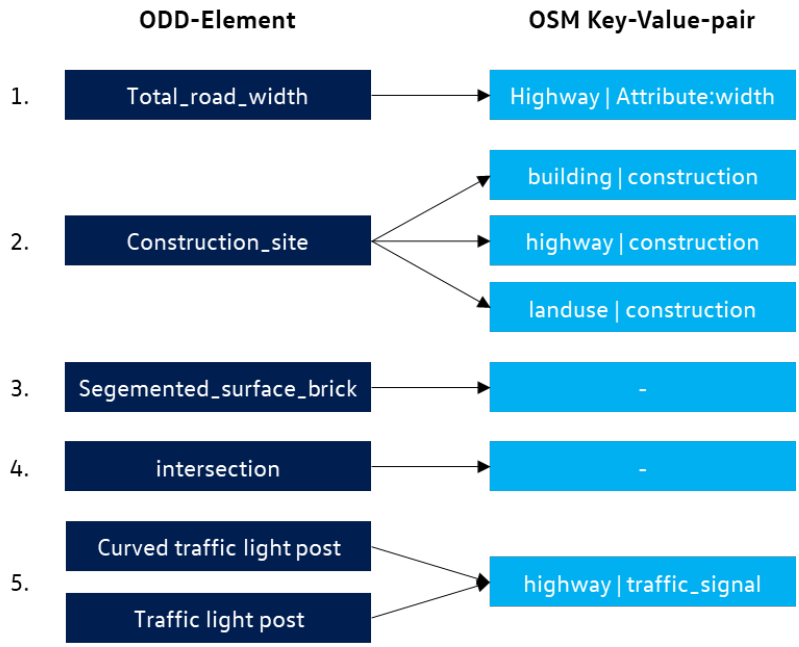

Abbildung 8.2: Zuordnung ODD-Element zu OSM-Key-Value-Pair

Diese strukturierte Herangehensweise ermöglicht eine präzise und umfassende Beschreibung der Betriebsumgebungen für ADS und stellt sicher, dass alle relevanten Umweltbedingungen berücksichtigt werden.

8.1.3. Erstellung der COD aus OSM

Auf Basis der gemeinsamen bzw. gemappten Taxonomie sollen CODs aus OSM automatisiert erzeugt werden. Dafür wird ein vierstufiger Prozess angewendet:

Schritt 1 - Definition des geografisch begrenzten Gebietes: Zur Erstellung von CODs wird ein lediglich ein Teilbereich von OSM verwendet, der für die Analyse eines möglichen Services relevant ist. Für diese Abfrage kann bspw. Overpass genutzt werden [Ove24]. Die Overpass-Abfragesprache ermöglicht es, Abfragen in einer durch die maximalen und minimalen Werte für Breiten- und Längengrad begrenzten Box zu beschränken. Dieser Schritt bietet eine erste Möglichkeit, Daten für eine spezifische Region aus OSM abzurufen. Da jedoch die meisten Städte und Stadtteile, die potenzielle Servicegebiete darstellen, keine rechteckigen Formen aufweisen, wird stattdessen ein Polygon verwendet, um das Servicegebiet zu definieren.

Schritt 2 - Extraktion relevanter Daten aus OSM: Als nächster Schritt muss für die Entwicklung von CODs ein funktionales Straßennetz erzeugt werden. Dies hat mehrere Gründe. Zum einen ist die Filterung relevanter Daten ein entscheidender Aspekt. Durch die Identifizierung der tatsächlich befahrbaren Straßen können irrelevante Wege wie Rad- oder Fußwege ausgeschlossen werden. Zum anderen ermöglicht ein vorgefiltertes Straßennetz eine strukturierte Datenanalyse. Das Straßennetz bildet ein Skelett, auf dem relevante Daten ausgewertet werden können.

Die dafür abgerufenen Daten aus OSM umfassen zwei Hauptelemente: Nodes und Ways. Ein Node repräsentiert einen spezifischen Punkt im Raum, wie beispielsweise den Beginn einer Straße oder eine Bushaltestelle. Ein Way hingegen bildet eine Verbindung mehrerer Nodes und beschreibt somit eine Strecke, (z.B. eine Straße). Beide Elemente sind mit Tags versehen, die wichtige Attribute wie den Straßennamen, den Straßentyp oder die zulässige Höchstgeschwindigkeit beinhalten. Durch die

Integration von Nodes und Ways in ein zusammenhängendes Netzwerk, lässt sich die Struktur als gerichteter Graph darstellen. Dieser bildet das funktionale Straßennetz.

Auf dieser Grundlage können dann CODs erstellt werden, indem entweder die Ways oder die Nodes analysiert werden. Dabei stellt sich die Frage, welcher Informationsradius bzw. welche Entfernung relevant für die Erstellung einer COD ist und ab wann die nächste beginnt. Diese Relevanzfrage ist komplex und nicht Teil dieser Arbeit. Das Ergebnis dieses Prozessschritts ist eine Zuordnung von relevanten Ways und Nodes zu einer COD.

Schritt 3 -Anreicherung der CODs mit Details: Bisher bestehen die CODs aus einer Menge von Ways und Nodes. Um diese mit Details anzureichern und Key-Value Pairs zu bilden, werden die grundlegenden Attribute, die den Nodes und Ways zugeordnet sind, aus den Tags abgeleitet (vgl. Quellcode 8-1).

```
"autobahn": tertiär",
"spur": "1",
"einspurig": "ja"
"straßenoberfläche": "asphalt"
```

Quellcode 8-1: Darstellung Key-Value-Pair in OSM

Um die beschriebenen Unterschiede der Taxonomiebegriffe zu überbrücken, kann die Zuordnungstabelle regelbasiert umgesetzt werden. In diesem Kontext besteht eine Regel aus zwei Teilen. Zum einen einer Bedingung, die in der OSM-Nomenklatur formuliert ist und die Regel auslöst. Zum anderen einem entsprechenden Key-Value-Paar aus der ODD-Taxonomie, dass bei Aktivierung der Regel eine COD erstellt. Das Ergebnis ist eine aus OSM abgeleitete COD auf Basis der ODD-Taxonomie (vgl. Quellcode 8-2).

```
straßentyp:
  - schnellstraße
straßenoberfläche:
  - asphalt
anzahl_fahrstreifen_in_ego_richtung: 1
anzahl_fahrstreifen_in_nicht_ego_richtung: 0
```

Quellcode 8-2: Abgeleitete COD aus OSM

Schritt 4 - Ergebnis: Am Ende dieses Prozesses wird eine Liste von CODs erstellt, die im YAML-Format repräsentiert werden kann und für weitere Inferenzprozesse geeignet ist.

8.1.4. ODD-Module

Neben der COD müssen für die Bestimmung eines fahrbaren Straßennetzes auch eine ODD und zugehörige Module erstellt werden. Um die nachfolgende Inferenz zu verdeutlichen, wird ein Beispiel genutzt. Das Beispiel beschreibt ein Modul, in dem Spielstraßen oder Schnellstraßen und eine Straßenoberfläche aus Asphalt als zulässig definiert sind (vgl. Quellcode 8-3).

```
...
MODULES:
  modul_v1:
    TYPE: dsm
    INCLUDE_AND:
      straßenoberfläche:
        - asphalt
```

```
straßentyp:
  - spielstraße
  - schnellstraße
```

Quellcode 8-3: Beispiel OSM ODD-Modul

8.1.5. Inferenz

Die Inferenz wird dann genutzt, um automatisiert zu bestimmen, ob eine COD den spezifizierten Bedingungen innerhalb der Module entspricht. Die Implementierung hierbei erfolgt auf Basis von Elastic Search. Eine Umsetzung mit einem SAT-Solver wurde nicht vorgenommen, könnte aber durchaus Vorteile hinsichtlich Skalierbarkeit und Konsistenz bieten (vgl. Kapitel 0**Error! Reference source not found.**).

In Elastic Search werden typischerweise Dokumente innerhalb der Engine gespeichert, die das Ausführen von Abfragen auf diesen Daten ermöglichen. In diesem spezifischen Anwendungskontext repräsentieren diese Abfragen die Module der ODD, während die Dokumente den CODs entsprechen. Um die Funktionalität zu operationalisieren, erfolgt eine Konvertierung der CODs von einer YAML-basierten Syntax in eine JSON-Syntax, die von Elastic Search verarbeitet werden kann (vgl. Codeblock 8-4).

```
{
  "query": {
    "bool": {
      "filter": [
        {
          "term": {
            "Straßenoberfläche ": "Asphalt"
          }
        },
        {
          "terms": {
            "Straßentyp ": ["Spielstraße", "Schnellstraße"]
          }
        }
      ]
    }
  }
}
```

Quellcode 8-4: Abbildung des Moduls als Elastic Search Query

Die Abfrage für Elastic Search ist speziell dafür gestaltet, Dokumente zu filtern, die bestimmte Kriterien erfüllen, ohne die Relevanzbewertung (Score) der Ergebnisse zu beeinflussen. Dafür wird eine Boole-Abfrage mit einem Filter-Segment verwendet, um die notwendigen Bedingungen für die Auswahl der Dokumente festzulegen. Innerhalb des Filter-Abschnitts gibt es zwei spezifische Bedingungen:

- Der erste terms-Filter prüft, ob die Straßenoberfläche den Wert "Asphalt" hat.
- Der zweite terms-Filter prüft, ob der Straßentyp entweder "Spielstraße" oder "Schnellstraße" ist.

Diese Abfrage würde sicherstellen, dass nur Dokumente zurückgegeben werden, die beide Kriterien erfüllen. Dies bedeutet, dass die entsprechenden Straßen sowohl aus Asphalt bestehen als auch entweder als Spielstraßen oder Schnellstraßen klassifiziert sind.

Elastic Search wird in einem Perkolator-Szenario genutzt, in dem die Abfrage-Dokument-Beziehung umgekehrt wird. Hier werden Abfragen gespeichert und Dokumente diesen zugeordnet, was die Verwaltung einer relativ kleinen Menge von Modulen ermöglicht und diesen gleichzeitig eine größere Liste von CODs zuordnet. Dieser Prozess generiert eine Liste von CODs, die jedem Modul in einer einzigen Sammelabfrage entsprechen. Die Inferenz zeigt am Ende, ob jede COD innerhalb der ODD liegt. Die analysierten CODs umfassen Daten über ihre räumliche Ausdehnung und ihren Verlauf. Diese können genutzt werden, um die Ergebnisse als Netzwerk zulässiger Straßen innerhalb des festgelegten Servicegebietes darzustellen (vgl. Abbildung 8.3). Unzulässige CODs werden als violett und zulässige CODs in blau dargestellt.

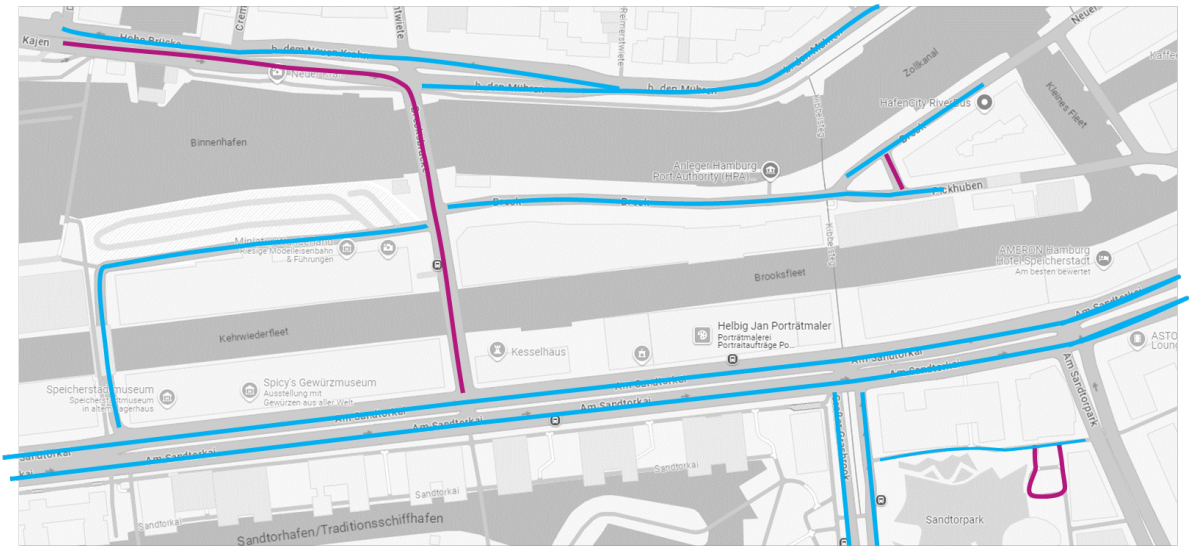

Abbildung 8.3:Ergebnisdarstellung des generierten Straßennetzes

Eine weitere Analysemöglichkeit ergibt sich aus der Untersuchung der potenziellen Gründe, warum eine Situation außerhalb der ODD fällt. Diese Informationen unterstützen die Priorisierung von Maßnahmen zur Erweiterung des Servicegebietes. Beispielsweise könnten damit wirtschaftlich vielversprechende Funktionserweiterungen bestimmt werden, indem potenzielle Erweiterung in Bezug zur Länge des GeoNets gesetzt werden. Beispielsweise könnte die Einbeziehung des Straßentyps A, der in einer einzelnen 2 km langen Situation auftritt, im Vergleich zu Typ B, der in zehn Situationen à 20 m auftritt, eine größere Ausdehnung des befahrbaren Gebiets ermöglichen. Aus wirtschaftlicher Sicht sollte dann die Realisierung dieses Straßenabschnitts bevorzugt werden. Darauf aufbauend kann dann eine technische Bewertung vorgenommen werden und je nach Ergebnis, die Funktionserweiterung entwickelt werden.

8.2. Validierung von Fahrverhalten

Dieses Kapitel konzentriert sich auf die die Validierung von Verhaltensweisen von ADS auf Basis des entwickelten Lösungskonzeptes. Hierbei liegt der Fokus auf der Herausforderung, festzustellen, ob beobachtetes Verhalten mit formalen Anforderungen (Modulen) übereinstimmt.

Die Problemanalyse hat dafür die wesentlichen Bedarfe aufgezeigt:

- **Anforderung TES 1:** Die ODD muss die Überprüfung des Fahrzeugverhaltens auf Basis verschiedener Dateninputs (Simulation, Realfahrt etc.) ermöglichen.

Ziel dieser Evaluierung ist es, die Effektivität und Nutzbarkeit des entwickelten Konzepts zu verdeutlichen und dessen potenzielle Anwendungen in der Praxis zu demonstrieren. Die Ergebnisse wurden in [SRR22] veröffentlicht. Nachfolgend wird auf die Umsetzung eingegangen, um die einzelnen Schritte und Ergebnisse umfassend darzustellen.

8.2.1. Konzept zur Validierung von autonomen Fahrverhalten

Das entwickelte Lösungskonzept stellt grundlegend die Möglichkeit bereit, das Fahrverhalten eines ADS automatisiert zu validieren (vgl. Abbildung 8.4). Dafür bedarf es:

- einer Abbildung von Verhaltensweisen mittels COD.
- der Nutzung von für diesen Zweck angepassten Modulen für die Spezifikation von zulässigen und unzulässigen Verhaltensanforderungen.
- einer Bewertung der Verhaltensweisen gemäß ihrer Übereinstimmung mit den definierten Verhaltensanforderungen auf Basis einer gemeinsamen Taxonomie.

Nachfolgend wird auf die notwendigen Prozessschritte sowie Anpassungen im Detail eingegangen.

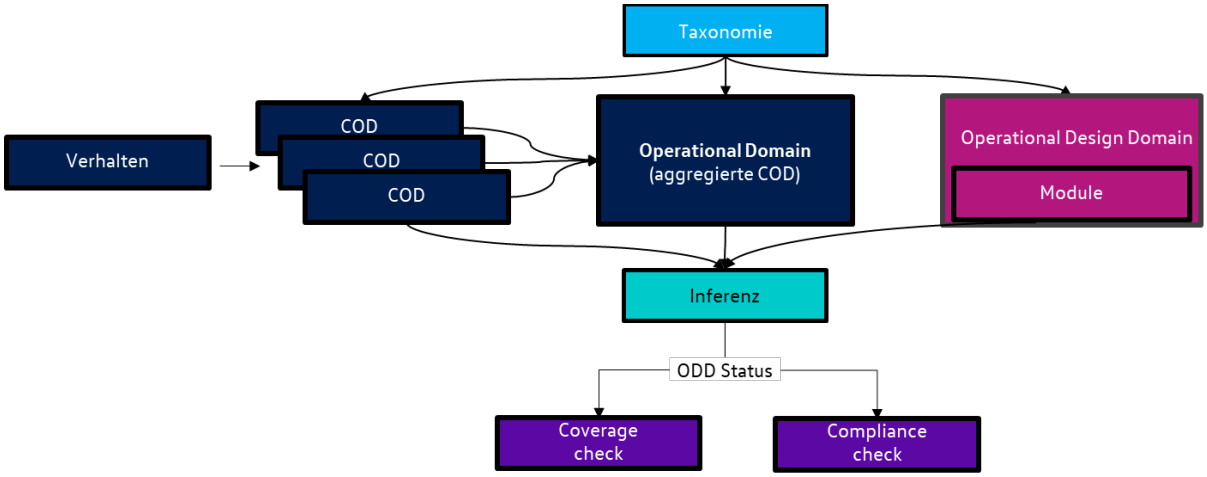

Abbildung 8.4: Konzept zur Validierung von autonomen Fahrverhalten

8.2.2. COD-Erstellung zur Abbildung von Verhalten

Normalerweise werden CODs als eine Sammlung von Schlüssel-Wert-Paaren modelliert, die einen Ausschnitt der Welt während einer kurzen Zeitdauer repräsentieren. In diesem Kontext beschreiben CODs jedoch einen spezifischen Moment oder Zustand innerhalb eines Szenarios. Ein Szenario besteht aus einer Abfolge solcher CODs und bildet einen kontinuierlichen oder logisch zusammenhängenden Vorgang ab, wie etwa das Durchfahren einer Kreuzung. Das Szenario Überqueren einer Kreuzung könnte dann beispielsweise anhand von drei CODs modelliert werden (vgl. Tabelle 8-1). [SRR22]

Tabelle 8-1: Beispielhafte CODs für Kreuzungen

#COD	Ankommen	In Kreuzung	Verlassen	Zeit
S1	True	False	False	t1
S2	False	True	False	t2
S3	False	False	True	t3

Für die Repräsentation von Verhalten bedarf es dafür jedoch Anpassungen. Verhalten repräsentiert das „Was passiert ist“ in Form einer Sequenz von CODs und diese treten typischerweise über mehrere Szenarien hinweg auf. Um die Notwendigkeit der Spezifikation von Verhaltensanforderungen für jede Sequenz zu vermeiden, wird ein Verhalten durch Reduktion einer Sequenz in eine einzige zusammengefasste Situation dargestellt. Die Verhaltensanforderung, eine Kreuzung nicht bei roter Ampel zu überqueren, könnte somit durch eine aggregierte Darstellung des Verhaltens bewertet werden. Diese besteht aus „Überquert“, „Farbe vorher“ und „Farbe“ (vgl. Tabelle 8-2). [SRR22]

Tabelle 8-2: Darstellung des aggregierten Verhaltens

#COD	Überquert	Farbe vorher	Farbe nachher
B1	True	grün	grün
B2	False	rot	grün
B3	True	rot	rot
B4	True	grün	rot

8.2.3. Definition von Verhaltensanforderungen mit Modulen

Verhaltensanforderungen sind spezifische Regeln, die definieren, wie sich das Fahrzeug in bestimmten Situationen verhalten soll. Diese Regeln werden als Bedingungen formuliert und können anhand von Modulen spezifiziert werden. Jedes Verhalten (COD) wird daraufhin überprüft, ob es den definierten Verhaltensanforderungen (Modulen) entspricht. Daraufhin kann die Übereinstimmung zwischen Verhalten und Verhaltensanforderungen bestimmt werden. Die Verhaltensanforderung „Kreuzung nicht bei rotem Licht zu überqueren“ kann anhand eines Moduls dargestellt werden (vgl. Quellcode 8-5).

```
…
MODULES:
   kreuzungsverhalten
……..TITLE: kreuzungsverhalten
……..TYPE: dsm
……..INCLUDE_AND:
…….ist_kreuzung: true
……..EXCLUDE_OR:
…….überqeuren_bei_rot

….überqueren_bei_rot:
……..TITLE: bei rot nicht überqueren
……..TYPE: dsm
……..EXCLUDE_OR:
…….in_kreuzung: false
…….Ampelfarbe:
……..- grün
……..- gelb
```

Quellcode 8-5:Verhaltensanforderung als Modul

Dieses Beispiel zeigt jedoch auch, dass Mehrdeutigkeiten bei der Auswertung eine große Rolle spielen. Beispielsweise sind die Definition einer Kreuzung, die Definition von “in der Kreuzung“, die relativen Positionen innerhalb der Kreuzung sowie die Farbe der relevanten Ampel unklar und bedürfen einer Präzisierung. Erste Ideen dazu werden im Paper genannt, für diese Arbeit sind diese aber nicht von

Relevanz. [SRR22] Neben der Mehrdeutigkeit sind auch die benötigten Daten sehr heterogen und erfordern unterschiedliche Datenquellen, um eine konsistente Aussage zuzulassen. Für das beschriebene Kreuzungsbeispiel könnten dies zur Auswertung des Verhaltens folgende Quellen umfassen:

- die Fahrprotokolle des Fahrzeugs, die das Zeitintervall für das Überqueren der Kreuzung sowie die genaue Position und Fahraufgabe während dieses Zeitraums bestimmen,
- den Plan der Kreuzung, der festlegt, welche Ampel für die entsprechende Position und Fahraufgabe gilt,
- Kamera- und Verkehrsdaten, die die Farbe jeder Ampel für das zugehörige Zeitintervall bestimmen [SRR22]

8.2.4. Auswertung des Verhaltens mittels Inferenz

Die Auswertung basiert auf einer Implementierung von Elastic Search, wie sie ähnlich auch für die Erstellung von GeoNet genutzt wurde. In diesem speziellen Kontext repräsentiert jedes Dokument in Elastic Search ein bestimmtes Verhalten (COD), während jede Anfrage eine spezifische Verhaltensanforderung (Modul) darstellt. Auf Basis des beschriebenen Ansatzes lassen sich drei Anwendungsfälle zur Bewertung von Verhaltensanforderungen identifizieren:

Einhaltung beobachteter Verhaltensweisen: Dieser Prozess beginnt mit der Auswertung von Fahrprotokollen, dem Extrahieren individueller Schlüssel-Wert-Paare und deren Zusammenführung zu interpretierten Verhaltenszusammenfassungen als COD. Anschließend wird die Einhaltung einzelner Regeln und letztlich die Gesamteinhaltung festgestellt. [SRR22]

Abdeckung von Modulen: Dieser Prozess zielt darauf ab, den Abdeckungsgrad aller beobachteten Cods durch eine Bibliothek von Modulen zu bestimmen. Dies ähnelt einer Abdeckungsbeurteilung bei herkömmlichen Tests und umfasst folgende Schritte:

- **Schritt 1**: Ermittlung der Anzahl der Module, die durch jede Situation COD ausgelöst werden.
- **Schritt 2**: Feststellung, wie häufig jede extrahierte COD auftritt.
- **Schritt 3**: Aufteilung der CODs nach deren Auftrittshäufigkeit (täglich, wöchentlich, monatlich, vierteljährlich, jährlich).
- **Schritt 4**: Berechnung des Anteils der CODs, die mindestens ein Modul auslösen. [SRR22]

Das Resultat zeigt den Anteil der CODs, die keine Module ausgelöst haben und deckt damit Abdeckungslücken auf. Wenn beispielsweise 10 % der CODs innerhalb eines Moduls nicht abgedeckt sind, bedeutet dies, dass die Module lediglich 90 % der beobachteten CODs abdecken.

8.3. Relevanzbewertung von SOTIF Szenarien

Dieses Kapitel widmet sich der Identifikation relevanter SOTIF-Szenarien, basierend auf dem entwickelten Lösungskonzept (vgl. Kapitel 5, 6 und 7). Angesichts der steigenden Komplexität der Systemfunktionalität ist eine umfassende Definition des Entwicklungsumfangs, insbesondere in Bezug auf die Sicherheitsargumentation, von zunehmender Bedeutung. Obwohl SOTIF bereits methodische Ansätze bereitstellt, fehlt eine nachvollziehbare Definition des Geltungsbereichs für Sicherheitsanalysen. Um diese Lücke zu schließen, wurden mithilfe einer formalen Spezifikation der ODD CODs innerhalb der SOTIF-Szenarien identifiziert, die für die ODD relevant sind. Die Ergebnisse dieser Analyse bieten aufschlussreiche Informationen über die Relevanz der identifizierten CODs.

Die Problemanalyse hat mehrere Anforderungen aufgezeigt, die den Bedarf dafür definieren (vgl. Kapitel 3):

- Anforderung REQ1: Die ODD muss klar die zulässigen und unzulässigen Betriebsbedingungen (Szenerie, dynamische Elemente und Wetter) innerhalb des Operational Domain definieren.
- Anforderung REQ15: Es muss möglich sein, die OD/COD mit der ODD abzugleichen. Hierfür ist es erforderlich, dass für jede Bedingung der ODD festgestellt werden kann, ob sie sich innerhalb oder außerhalb der OD/COD befindet. Diese Überprüfung soll mittels einer binären Auswertung (True/False) erfolgen.
- Anforderung SAF6: Die ODD sollte die Ableitung relevanter Szenarien unterstützen und fördern.

Das Ziel ist die Anwendbarkeit des entwickelten Konzepts zu demonstrieren und dessen Einsatzmöglichkeiten in der Praxis aufzuzeigen. Die Ergebnisse dieser Untersuchung wurden im Detail veröffentlicht [RSR23]. Der weitere Verlauf geht detailliert auf die einzelnen Schritte ein.

8.3.1. Konzept für die Relevanzbewertung

Das Konzept nutzt eine formale Spezifikation der ODD, um den Geltungsbereich von SOTIF-Szenarien durch Überprüfung der Relevanz zu definieren. Dafür werden Szenarien als eine Abfolge von Situationen/COD, verbunden durch Events, aufgefasst (vgl. Abbildung 3.12). Im Kontext der ODD, kann dann jede COD entweder als relevant oder nicht relevant klassifiziert werden (vgl. Abbildung 8.5, violette Linie). Zudem kann unter Anwendung der HARA jede COD als gefährlich oder nicht gefährlich bewertet werden (vgl. Abbildung 8.5, blaue bzw. hellblaue Linie). Das Ziel dieser Methodik ist es, HARAs für nicht relevante CODs zu vermeiden und damit unnötigen Aufwand bei der Klassifizierung gemäß der violetten Linie zu reduzieren. [RSR23]

Für CODs, die als nicht relevant für die ODD eingestuft werden, kann die Durchführung einer HARA möglicherweise vollständig entfallen oder nur in einer vereinfachten Form erforderlich sein. Dabei ist jedoch ist jedoch zu beachten, dass nicht relevante CODs nicht gänzlich ignoriert werden dürfen., da auch CODs, die außerhalb der ODD liegen, auftreten können. Nach dem aktuellen Stand der Technik werden solche CODs durch den Übergang des Systems in eine Minimal Risk Condition (MRC) behandelt und können mit Hilfe vereinfachter Sicherheitsanalysen betrachtet werden (vgl. Kapitel 2.1.1). Andere CODs, die außerhalb der ODD liegen, können unter Umständen vollständig ignoriert werden, vorausgesetzt, es kann gewährleistet werden, dass diese Elemente im späteren Betriebsbereich nicht auftreten. Dies ist beispielsweise für ortsfeste Elemente wie Straßenbahnen der Fall. Aus Gründen der Einfachheit, wird nachfolgend keine weitere Differenzierung zwischen diesen Klassifikationen vorgenommen. [RSR23]

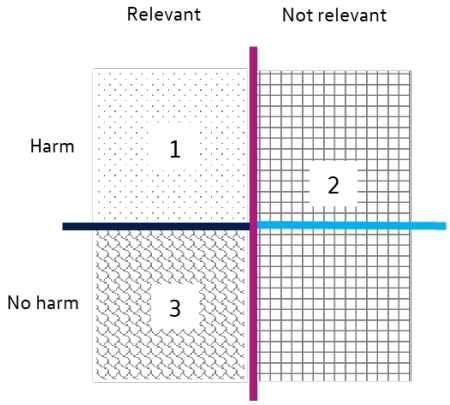

Abbildung 8.5: Klassifikation der CODs, nach [RSR23]

Für die Implementierung des Konzeptes werden die notwendigen Aktivitäten in zwei Hauptphasen gegliedert. In der ersten Phase erfolgt die Definition des Geltungsbereichs durch die Bestimmung der

Relevanz der aus Szenarien abgeleiteten COD in Bezug auf die ODD. Anschließend wird in der zweiten Phase die Durchführung einer HARA für die als relevant eingestuften Situationen betrachtet. Die Inputs für diesen Prozess umfassen verschiedene Schlüsselkomponenten (vgl. Abbildung 8.6):

- **COD**: Jedes Szenario enthält mehrere CODs, die durch Ereignisse miteinander verbunden sind. Für die Relevanzbewertung gegenüber der ODD, müssen die CODs vereinfacht werden, da SOTIF-Szenarien auch Informationen enthalten, die nicht für die Zuordnung zur ODD geeignet sind. Zu diesen Informationen gehören beispielsweise spezifische Manöver. Diese Vereinfachung stellt eine Herausforderung dar, da die Inferenzmechanismen andere Informationen benötigen als jene, die in der HARA verwendet werden.
- **ODD-Spezifikation**: Eine präzise Definition der geeigneten und ungeeigneten Betriebsbedingungen innerhalb der ODD ist erforderlich, um eine effektive Zuordnung zu ermöglichen.
- **Inferenz**: Die Inferenz prüft die Zuordnung der CODs gegenüber der ODD, um ihre Relevanz zu identifizieren. Basierend auf den Ergebnissen dieser Analysen wird die Identifikation gefährlicher Events durchgeführt, wie es in der funktionalen Sicherheit und den Systems Theoretic Process Analysis (STPA) üblich ist [LT18]. Diese Identifikation erfolgt für jede potenziell gefährliche und relevante COD. Die HARA folgt auf die Identifikation gefährlicher Events. In dieser Analyse werden die Schwere des Schadens, die Exposition gegenüber dem Risiko und die Kontrollierbarkeit des Ereignisses bewertet, sowie ein akzeptables Risikoniveau bestimmt. [RSR23]

Das Ergebnis dieses Prozesses ist eine Auflistung von SOTIF-Szenarien, die gemäß den in Abbildung 8.5 illustrierten Kriterien klassifiziert wurden sind. Diese systematische Klassifikation und die daraus resultierenden Analysen tragen wesentlich zur Verfeinerung der Sicherheitsbewertungen und zur Minimierung unnötiger Analysen bei, indem nur die relevanten und potenziell gefährlichen CODS bzw. Szenarien berücksichtigt werden.

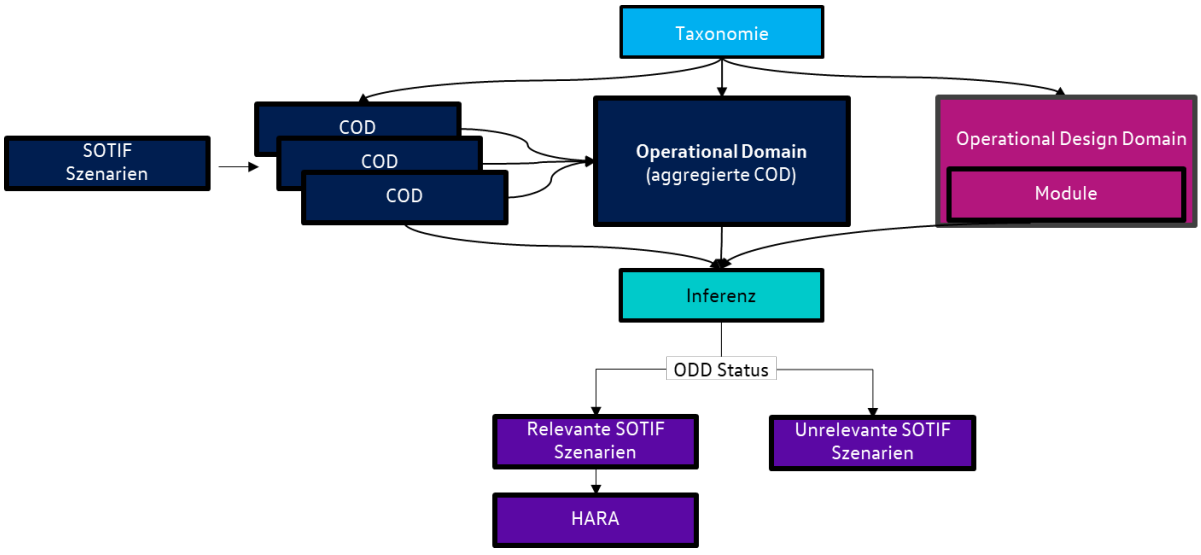

Abbildung 8.6: Erweiterung des Lösungskonzeptes für die Bewertung von SOTIF Szenarien, nach [RSR23]

8.3.2. Demonstration der Anwendbarkeit anhand eines Beispiels

Um das Konzept an einem Beispiel zu illustrieren, wird eine ODD definiert (vgl. Quellcode 8-6). Diese beschreibt, dass signalisierte Kreuzungen innerhalb der ODD liegen, spezifische Kreuzungen, die eine Straßenbahn beinhalten, jedoch nicht.

```
...
MODULES:
  modulev1:
  TYPE: dsm
  INCLUDE_AND:
    Intersection:
      - signalized_intersection
  EXCLUDE_OR:
    dynamic_objects:
      - lightrail
```

Quellcode 8-6:Beispiel ODD – Bestimmung von SOTIF Szenarien

Als nächster Schritt müssen die CODs aus den SOTIF-Szenarien bestimmt werden. Das erste Szenario beschreibt eine reguläre signalisierte Kreuzung und das zweite eine signalisierte Kreuzung mit Straßenbahn (vgl. Abbildung 8.7). Das erste Beispiel besteht aus einem Szenario, in dem das Ego-Fahrzeug eine Kreuzung betritt, während die Ampel von Grün auf Rot wechselt (vgl. Abbildung 8.7, linke Seite). Das Szenario kann durch die hier dargestellte Abfolge von CODs und Events repräsentiert werden:

- Die initiale COD beschreibt das Ego-Fahrzeug, das sich einer signalisierten Kreuzung nähert, während die Ampel grün ist.
- Das Event „die Farbe der Ampel“ wechselt tritt ein.
- Ein mögliches Ergebnis ist eine COD ohne Schaden, bei der das Ego-Fahrzeug vor dem Eintreten in die Kreuzung zum Stillstand kommt.
- Das andere mögliche Ergebnis führt zu einer COD mit Schaden, bei der das Ego-Fahrzeug in die Kreuzung einfährt und mit einem anderen Fahrzeug kollidiert. [RSR23]

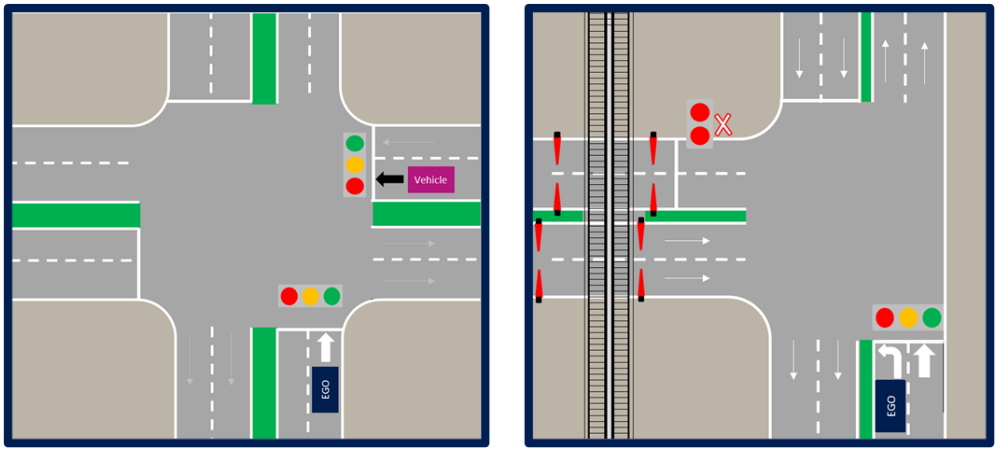

Abbildung 8.7: Darstellung beispielhafter SOTIF Szenarien, nach [RSR23]

Entsprechend des Konzeptes, muss nun anhand der extrahieren COD bestimmt werden, ob das Szenario relevant für die ODD ist. In diesem Fall wird die COD als relevant klassifiziert, da alle Elemente der COD innerhalb der ODD liegen. Der nächste Schritt besteht in der Identifikation gefährlicher Ereignisse in der relevanten COD. In dem Beispiel kann dies zu einem schadhaften Ergebnis führen und daher wird die COD als gefährlich eingestuft. Abschließend wird eine HARA durchgeführt, um die Schwere des Schadens, die Exposition und die Kontrollierbarkeit des Farbwechselereignisses zu bestimmen. Darauf erfolgt die resultierende Situationsklassifikation (vgl. Abbildung 8.8) gemäß der COD-Klassifikation in Quadrat 1 und 3 (vgl. Abbildung 8.5). [RSR23]

Das zweite Szenario (Abbildung 8.7, rechtes Szenario) besteht aus der nachstehenden Abfolge von COD und Events:

- Die initiale COD beschreibt das Ego-Fahrzeug, das sich einer Kreuzung mit Bahnübergang nähert.
- Das Event „Straßenbahnsignal wird rot“ tritt auf.
- Ein mögliches Ergebnis ist eine COD ohne Schaden, bei der das Ego-Fahrzeug die Bahngleise nicht überquert oder sie überquert, aber eine Kollision mit der Straßenbahn vermeidet.
- Das andere mögliche Ergebnis führt zu einer COD mit Schaden, bei der das Ego-Fahrzeug die Bahngleise überquert und mit der Straßenbahn kollidiert. [RSR23]

Die Relevanzprüfung führt zu dem Ergebnis, dass dieses Szenario nicht relevant für die ODD ist. Die weiteren Schritte des Prozesses können entfallen und es muss keine oder nur eine vereinfachte HARA durchgeführt werden. Dies resultiert in einer Einordnung der COD in Quadrant 2 (vgl. Abbildung 8.8) gemäß COD-Klassifikation (vgl. Abbildung 8.5).

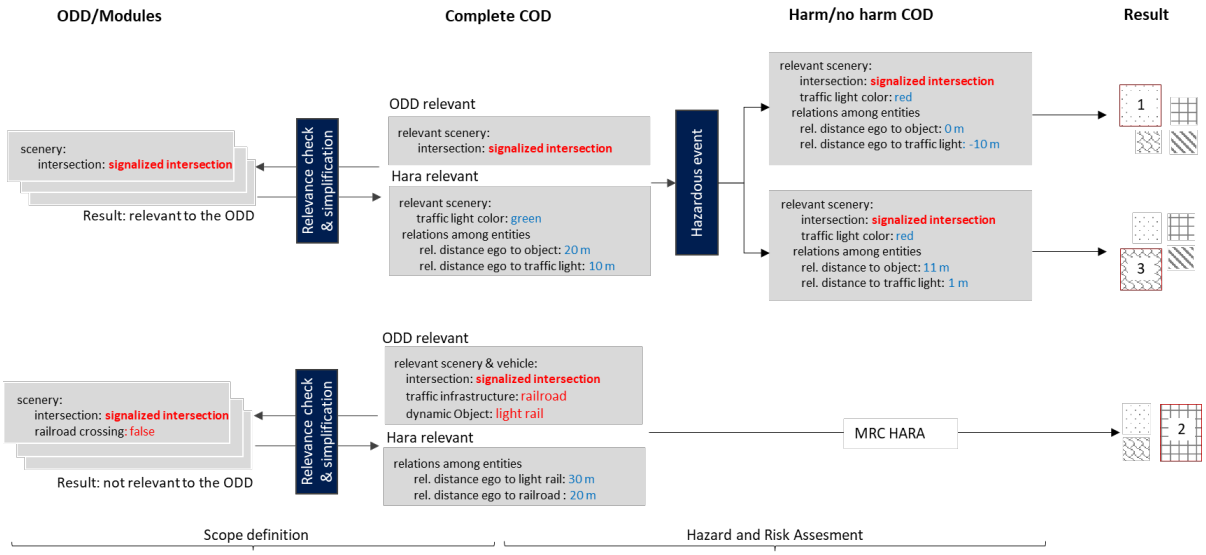

Abbildung 8.8: Visualisierung der Relevanzprüfung, nach [RSR23]

9. Zusammenfassung

In dieser Doktorarbeit wurde die Modellierung von Operational Design Domains für hochautomatisierte Fahrzeuge untersucht. Zu diesem Zweck wurden eine formale Beschreibungssprache und ein Nutzungskonzept entwickelt. Die Vorstellung der dafür benötigten Grundlagen erfolgt in Kapitel 2. Dabei wurden zu Beginn die unterschiedlichen Stufen der Fahrzeugautomatisierung beschrieben sowie notwendige Begrifflichkeiten für das grundlegende Verständnis definiert (vgl. Kapitel 2.1.1-2.1.3). Darauf aufbauend wurden die Grundlagen der Logik erklärt, indem auf drei verschiedene Typen explizit eingegangen wird. Zu diesen Logikarten zählen die Aussagenlogik (vgl. Kapitel 2.2.1), die Beschreibungslogik (vgl. Kapitel 2.2.2) und die typisierte Prädikatenlogik erster Ordnung (vgl. Kapitel 2.2.3). Im Anschluss daran erfolgt die Erläuterung der Grundlagen der formalen Sprache, indem explizit Sprache, Grammatik und Domänenspezifische Sprache beschrieben werden (vgl. Kapitel 2.3.1-2.3.4).

Im weiteren Verlauf werden in Kapitel 3 die ungelösten Probleme im Zusammenhang mit der Beschreibung und Nutzung der ODD im Kontext der Entwicklung herausgearbeitet. Dabei geht dieses Kapitel auf verschiedene Phasen relevanter Entwicklungsprozesse ein. Angefangen bei der Betrachtung der regulatorischen Rahmenbedingungen für ADS (vgl. Kapitel 3.1) und dem Business Case (vgl. Kapitel 3.2), wird in Kapitel 3.3 der Entwicklungsprozess für Anforderungen analysiert. Neben dem Systemdesign wird speziell auf die funktionale Sicherheit und die Sicherheit der intendierten Funktionalität in Zusammenhang mit der ODD eingegangen (vgl. Kapitel 3.5). Des Weiteren werden auf Grundlage des Entwicklungsprozesses notwendige Testprozesse und der sich anschließende Fahrzeugbetrieb beschrieben. Zusammenfassend werden daraus in Kapitel 3.8 Anforderungen an eine Lösung abgeleitet.

Die in Kapitel 3 beschriebenen Prozesse und abgeleiteten Anforderungen bilden für die nachfolgende Betrachtungen in Kapitel 4 die Basis. Dieses Kapitel widmet sich der Erörterung der Forschungsfragen, indem die definierten Anforderungen zu Clustern zusammengefasst werden (vgl. Kapitel 4.1). Diese Cluster stellen bisher ungelöste Problemfelder der Entwicklung dar und sollen durch eine Lösung aufgegriffen werden. Die Cluster sind im Einzelnen:

- Leichte Verständlichkeit und technische Präzision: Die Sprache der ODD muss gleichzeitig verständlich für Nicht-Experten und präzise genug für Verifikations- und Validierungsprozesse sein.
- Repräsentation unterschiedlicher Informationsbedarfe: Die ODD sollte die vielfältigen Informationsanforderungen verschiedener Stakeholder abbilden, ohne dabei Verständlichkeit und Nutzbarkeit zu verlieren.
- Modularität und Wiederverwendbarkeit ohne zusätzliche Komplexität: Die Sprache muss modular und flexibel sein, sodass Komponenten wiederverwendbar sind, ohne dabei unnötige Komplexität zu erzeugen
- Quantitative und qualitative Bewertung der ODD: Die ODD muss in der Lage sein, sowohl qualitative als auch quantitative Bewertungen effizient zu integrieren und darzustellen.
- Prozessintegration: Die ODD-Sprache muss für verschiedene Prozesse und Teams anwendbar sein und den Informationsaustausch nahtlos unterstützen.

Um anhand der Cluster zu verdeutlichen, das eine Forschungslücke besteht, wird im darauffolgenden Kapitel 4.2 der Stand der Wissenschaft beleuchtet und gegenüber der Cluster eingeordnet. Kapitel 4.3 leitet daraus dann die Forschungsfragen ab, ehe in Kapitel 4.4 das Lösungskonzept skizziert wird.

Den Kern der Lösung stellt dabei die Sprache dar. Im Zusammenhang mit einer entwickelten Methode bietet diese die Möglichkeit, Situationsdaten (z.B. Kartendaten oder Fahrprotokolle) mit einer

modularisierten ODD auf Basis einer gemeinsamen Taxonomie zu vergleichen. Damit wird die Möglichkeit geschaffen, zulässige und unzulässige Betriebsbedingungen zu klassifizieren.

In Kapitel 5 wird die dafür benötigte Sprache und das Modularisierungskonzept definiert. Dafür erläutert Kapitel 5.1 zunächst die grundlegende Syntax von YAML, um eine Basis für die sprachlichen Definitionen zu schaffen. Darauf aufbauend werden in Kapitel 5.2 die relevanten Basiselemente der Sprache definiert. Kapitel 5.3 beschreibt die Syntax und Semantik der Taxonomie, wohingegen Kapitel 5.4 die Struktur der Situationsdaten definiert. Abschließend widmet sich Kapitel 5.5 der Syntax und Semantik der ODD und ihrer Module.

Die entwickelte Sprache wird in Kapitel 6 aufgegriffen und es werden Modellierungspattern abgeleitet, um die Nutzbarkeit der Sprache zu verbessern und die Anwendbarkeit zu demonstrieren. Kapitel 7 geht dann auf zwei verschiedene Realisierungskonzepte ein. Zuerst wird dazu Kapitel 7.1 eine Lösung auf Basis von ElasticSearch vorgestellt, ehe in Kapitel 7.2 eine Umsetzung mit einem Standard-Toolset aufgezeigt wird. Letzteres zeigt die Übertragbarkeit und Skalierbarkeit der Lösung auf.

Das letzte Kapitel demonstriert die praktische Nutzbarkeit der entwickelten DSL und der zugehörigen Werkzeuge anhand verschiedener Anwendungsfälle. Diese umfassen in Kapitel 8.1 die Erstellung zulässiger Straßennetzwerke, in Kapitel 8.2 die Validierung des Fahrverhaltens sowie in Kapitel 8.3 die Relevanzbewertung von SOTIF-Szenarien.

Insgesamt wurden in dieser Arbeit drei Forschungsfragen aufgestellt (vgl. Kapitel 4.3) und in unterschiedlichen Kapiteln beantwortet:

- Forschungsfrage 1 *„Wie kann eine Sprache entwickelt werden, die eine präzise Beschreibung und Repräsentation von unterschiedlichen Betriebsbedingungen eines ADS ermöglicht, indem sie Betriebsbedingungen formal spezifiziert und diese modular organisiert? Wie können zusätzlich Modellierungspatterns abgeleitet werden, die die Anwendbarkeit der Sprache vereinfachen?“* wird in Kapitel 5 und 6 beantwortet. In diesem Kontext definiert Kapitel 5 eine umfassende Syntax und Semantik für die Sprache. Kapitel 5.5.2 geht dabei speziell auf das Konzept der Modularisierung ein. Kapitel 6 widmet sich vollumfänglich der Beschreibung von Modellierungspattern, die die Anwendbarkeit der Sprache demonstrieren.
- Forschungsfrage2 *„Wie lässt sich eine toolgestützte Umsetzung des Lösungskonzeptes gestalten, dass nicht nur den Abgleich des vorgesehenen Betriebsbereiches des ADS mit der ODD unterstützt, sondern auch relevante Daten für die weitere Entwicklung des Systems liefert sowie einen skalierbaren Ansatz bietet?“* wird bereits über die Entwicklung des Lösungskonzeptes in Kapitel 4.4 adressiert. Dies bietet nicht nur die Möglichkeit, den Betriebsbereich auszuwerten, sondern kann auch für weitere Anwendungsfälle (z.B. der Bestimmung relevanter SOTIF Szenarien) genutzt werden. Kapitel 7 greift das Lösungskonzept dann auf und demonstriert zwei werkzeuggestützte Umsetzungsmöglichkeiten.
- Forschungsfrage 3 *„Wie können die Nutzbarkeit und Effektivität der entwickelten Lösungen zur präzisen Beschreibung und Strukturierung von Betriebsbedingungen eines ADS sowie deren Entwurf und Nutzbarkeit im Entwicklungsprozess durch ausgewählte Anwendungsfälle demonstriert werden?“* wird in Kapitel 8 beantwortet. Die Kapitel 8.1, 8.2 und 8.3 stellen dafür jeweils Anwendungsfälle entlang des Entwicklungsprozesses vor.

Auch wenn die Doktorarbeit einen Beitrag für die definierten Fragestellungen geleistet hat, bieten verschiedene Themen Ansätze für weiterführende Forschung. Auf einige wird nachfolgend eingegangen.

Für die Anwendung der ODD wird vorausgesetzt, dass Situationsdaten (CODs) in einem relationalen Tabellenformat oder im YAML-Format vorliegen. Diese Daten bestehen aus einer Sammlung von Zeilen, wobei jede Zeile eine eindeutige COD mit definierten Werten für bestimmte Felder repräsentiert (vgl. Kapitel 5.4). Die Felder entsprechen den Spaltennamen der Tabelle. Jede Zeile wird dann im Hinblick darauf ausgewertet, ob sie wahr oder falsch ist. Nur die als „wahr“ bewerteten CODs sind am Ende zulässig und können beispielsweise durch ein ADS befahren werden. Jedes Feld kann dabei auch einen Wert von null oder leer, wenn dieser unbekannt ist, annehmen. Die Behandlung dieser fehlenden Werte ist dann entscheidend, da diese die Auswertung der ODD direkt beeinflussen und damit auf die Sicherheit des Gesamtsystem Auswirkung hat.

Zudem könnte zwischen der Wichtigkeit verschiedener Bedingungen differenziert werden. Beispielsweise können kritische Elemente wie Fahrbahnmarkierungen, die für die sichere Durchführung der Fahraufgabe unerlässlich sind, als besonders relevant deklariert werden. Um diese Problematik zu adressieren, könnte die Syntax erweitert werden, um sicherheitskritische Werte explizit zu definieren. So könnte beispielsweise festgelegt werden, dass wenn Daten zur Existenz von Fahrspuren fehlen, diese automatisch als „falsch“ bewertet werden. Der Umgang mit fehlenden Daten könnte so verbessert werden und die Sicherheit des Systems erhöht.

Für die Evaluation der Werkzeugunterstützung in Kapitel 7.2 wurden künstlichen Datensätze für die ODD und COD genutzt. Dadurch ist bei der Interpretation der Ergebnisse Vorsicht geboten, da künstliche ODD-Beschränkungen möglicherweise nicht repräsentativ für die ODD-Beschränkungen sind, die typischerweise in realen automobilen Fahrsystemen auftreten. Die Verwendung künstlicher Beschränkungen in experimentellen Settings ermöglicht zwar eine kontrollierte Analyse und erleichtert die Identifikation von Leistungstrends und potenziellen Kapazitäten von Systemen, doch die Übertragbarkeit dieser Ergebnisse auf reale Szenarien könnte eingeschränkt sein. Zukünftig könnte dies durch reale Daten ersetzt werden und die Evaluation erneut überprüft werden.

Abschließend werden Anwendungsfälle der ODD demonstriert, dabei wird jedoch nicht auf die Nutzung der ODD während des Betriebs eingegangen. Dieser Anwendungsfall bringt spezielle Anforderung mit sich, wie beispielsweise die echtzeitfähige Auswertung der ODD gegenüber der Umgebung. Das bedeutet, es muss jederzeit bestimmt werden können, ob sich das Fahrzeug noch innerhalb der ODD befindet oder nicht. Generell sollte die definierte Sprache für diese Art von Anwendungsfall nutzbar sein. Ein Nachweis steht jedoch aus und bietet Ansatzpunkte für weitere Forschung.

Literaturverzeichnis

[AHD+18]
Adina Aniculaesei, Falk Howar, Peer Denecke und Andreas Rausch. Automated generation of requirements-based test cases for an adaptive cruise control system. In Cyrille Artho and Rudolf Ramler, editors, 2018 IEEE Workshop on Validation, Analysis and Evolution of Software Tests (VST@SANER), pages 11–15. IEEE, 2018.

[Alu15]
Rajeev Alur. Principles of Cyber-Physical Systems. MIT Press, Cambridge, MA, USA, 1 edition, April 2015.

[Asa21]
ASAM, Hrsg. "ASAM OpenODD Standard – Concept Paper." 2021. URL: https://www.asam.net/index.php?eID=dumpFile&t=f&f=4544&token=1260ce1c4f0afdbe18261f7137c689b1d9c27576 (besucht am 06.07.2022).

[ASK+23]
Adina Aniculaesei, Christian Schindler, Christoph Knieke, Andreas Rausch, Daniel Rohne und Andreas Richter. "A Method for ODD Specification and Verification with Application for Industrial Automated Driving Systems," *2023 International Conference on Computational Science and Computational Intelligence (CSCI)*, Las Vegas, NV, USA, 2023, pp. 1519-1526.

[Aut20]
Automated Vehicle Safety Consortium, Hrsg. AVSC Best Practice for Describing an Operational Design Domain: Conceptual Framework and Lexicon. SAE Industry Technologies Consortia, April 2020.

[Bar12]
Bartels, A.: Systembeschreibung automatischer Fahrfunktionen. In: Gasser, T. M. (Hrsg.); Arzt, C. (Hrsg.); Ayoubi, M. (Hrsg.); Bartels, A. (Hrsg.); Bürkle, L. (Hrsg.); Eier, J. (Hrsg.); Flemisch, F. (Hrsg.); Häcker, D. (Hrsg.); Hesse, T. (Hrsg.); Huber, W. (Hrsg.); Lotz, C. (Hrsg.); Maurer, M. (Hrsg.); RuthSchumacher, S. (Hrsg.); Schwarz, J. (Hrsg.); Vogt, W. (Hrsg.): Rechtsfolgen zunehmender Fahrzeugautomatisierung: gemeinsamer Schlussbericht der Projektgruppe. F83. 2012, S. 23–44.

[Bar16]
Bardt, Hubertus (2016). "Deutsche Autoindustrie und autonomes Fahren." Wirtschaftsdienst, Vol. 96, Issue 10, pp. 776–778.

[Bau20]
Lee Bauer. *Entwicklungsansätze für die Fahrzeugelektronik der Zukunft*. Online article, Springer Professional, 11 November 2020. [URL](https://www.springerprofessional.de/automobilelektronik---software/).

[Bei12]
Sven A. Beiker. “Legal Aspects of Autonomous Driving”. In: Santa Clara Law Review 52.4 (2012), pp. 1145–1561.

[Ber19]
Bradley Berman. „The key to autonomous vehicle safety is ODD.“ 2019.

[BFH+18]
René Bormann, Philipp Fink, Helmut Holzapfel, Stephan Rammler, Thomas Sauter-Servaes, Heinrich Tiemann, Thomas Waschke, Boris Weirauch. *Die Zukunft der deutschen Automobilindustrie: Transformation by Disaster oder by Design?* WISO Diskurs, Friedrich-Ebert-Stiftung, 2018.

[BGM21]
Hans Braun, Michael Gerke und Reiner Marchthaler. "Sicherheit beim autonomen Fahren - Bewertung durch maximale Entropie." ATZ Automobiltechnische Zeitschrift, vol. 123, 2021, S. 64–69.

[BHR+17]
Martin Buechel, Gereon Hinz, Frederik Ruehl, Hans Schroth, Csaba Gyoeri, and Alois Knoll. „Ontology-Based Traffic Scene Modeling, Traffic Regulations Dependent Situational Awareness, and Decision-Making for Automated Vehicles.“ 2017.

[BHS07]
Bernhard Beckert, Reiner Hähnle und Peter H. Schmitt. Verification of Object-Oriented Software. The KeY Approach. Springer Berlin Heidelberg, 2007.

[BK21]
Manfred Broy und Marco Kuhrmann. "Anforderungsanalyse und Anforderungsmanagement." In: Einführung in die Softwaretechnik. Xpert.press. Springer Vieweg, Berlin, Heidelberg, 2021.

[BMM18]
Gerrit Bagschik, Till Menzel, and Markus Maurer. „Ontology-Based Scene Creation for the Development of Automated Vehicles.“ 2018.

[Bri20]
British Standards Institution. PAS 1883:2020 Operational Design Domain (ODD) Taxonomy for an Automated Driving System (ADS) – Specification. BSI Standards Limited, 2020.

[Bun22]
Bundesministerium für Digitales und Verkehr, Hrsg. „Verordnung zur Regelung des Betriebs von Kraftfahrzeugen mit automatisierter und autonomer Fahrfunktion und zur Änderung straßenverkehrsrechtlicher Vorschriften“. Bundesrat Drucksache 86/22, 24.02.2022

[BW11]
Amy Brown und Greg Wilson. “The Architecture of Open Source Applications”. Volume 1. Elegant Design for Real-World Applications. Independently published, 2011.

[Cal22a]
California Department of Motor Vehicles, Hrsg. “ADDENDUM: Operational Design Domain – Driverles Deployment in California”. 2022. URL: https://www.cpuc.ca.gov/-/media/cpuc-website/divisions/consumer-protection-and-enforcement-division/documents/tlab/av-programs/cruise-driverless-deployment-odd-2023-08.pdf (besucht am 06.04.23).

[Cal22b]
California Department of Motor Vehicles, Hrsg. “Attachment A – Statement of Map of Operational Design Domain – Driverles Deployment”. 2022. URL: https://www.cpuc.ca.gov/-/media/cpuc-website/divisions/consumer-protection-and-enforcement-division/documents/tlab/av-programs/waymo-driverless-deployment-odd-202308.pdf (besucht am 05.04.23).

[Cal23a]
California Department of Motor Vehicles, Hrsg. California Autonomous Vehicle Regulations. 2023. URL: https://www.dmv.ca.gov/portal/vehicle-industry-services/autonomous-vehicles/california-autonomous-vehicle-regulations/ (besucht am 30.11. 2023).

[Cal 23b] California Public Utilities Commission, Hrsg. CPUC Approves Permits for Cruise and Waymo to Charge Fares for Passenger Service in San Francisco. 10. August 2023. URL:

https://www.cpuc.ca.gov/news-and-updates/all-news/cpuc-approves-permits-for-cruise-and-waymo-to-charge-fares-for-passenger-service-in-sf-2023 (besucht am 15. 09. 2023).

[Car22]
CarSwitch, Hrsg. What Is Mercedes Drive-Pilot? 1. Juni 2022. URL: https://carswitch.com/newsroom/what-is-mercedes-drive-pilot/ (besucht am 08.07.23).

[CBT21]
Anne Collin, Amitai Y. Bin-Nun und Radboud Duintjer Tebbens. „Plane and Sample: Maximizing Information about Autonomous Vehicle Performance using Submodular Optimization.“ 2021.

[Cho02]
Noam Chomsky. Syntactic Structures. Berlin, New York: De Gruyter Mouton; 2002.

[Cho20]
HongSeok Cho. „Operational Design Domain (ODD) Framework for Driver-Automation Integrated Systems.“ 2020.

[CLR+09]
Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest und Clifford Stein. “Introduction to Algorithms”. 3rd Edition. MIT Press, 2009

[Coo89]
William R Cook. “A Denotational Semantics of Inheritance” PhD diss., Brown University, 1989.

[Cza18]
Czarnecki, K. Operational world model ontology for automated driving systems – Part 1: Road structure. University of Waterloo, Canada: Waterloo Intelligent Systems (WISE) Lab, 2018

[D12]
Donges, E.: Fahrerverhaltensmodelle. In: Winner, H. et al.: *Handbuch Fahrerassistenzsysteme*. 2. Auflage, S. 15–23, Vieweg+Teubner Verlag, Wiesbaden, 2012.

[Deu22]
Deutsches Patent- und Markenamt (DPMA), Hrsg.. Autonomes Fahren Teil 3: Zahlen und Fakten, 2022.

[DKL84]
Veronique Donzeau-Gouge, Gilles Kahn, Bernard Lang, Bertrand Mélèse. Document Structure and Modularity in Mentor. In Proc. of Software Engineering Symposium on Practical Software Development Environments 1984. ACM,1984.

[Ecm20]
ECMA International, Hrsg. "The JSON Data Interchange Syntax - 2nd Edition." ECMA-404, 2020. URL: https://www.ecma-international.org/publications-and-standards/standards/ecma-404/ (besucht am: 17.10.2023).

[Eur22]
European Commission. Regulation (EU) 2022/1426 on Type-Approval of Automated Driving Systems (ADS) for Fully Automated Vehicles. Official Journal of the European Union, 2022.

[EWL+21]
Christof Ebert, Michael Weyrich, Benjamin Lindemann und Sarada Preethi Chandrasekar. "Systematisches Testen für autonomes Fahren." ATZelektronik, vol. 16, 2021, pp. 26–31.

[Fil09]
Filatova, Natalia. Einführung in die Beschreibungslogiken. Ludwig-Maximilians-Universität München, Sommersemester 2009.

[Fra18]
Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung e. V., Hrsg. Hochautomatisiertes Fahren auf Autobahnen – Industriepolitische Schlussfolgerungen. 2018.

[GJ90]
R. Golecki und J. Jungmann. Einführung in die Aussagenlogik. Universität Kaiserslautern, August 1990.

[Goo23]
Google, Hrsg. "Protocol Buffers - Developer Documentation." 2023. URL: https://developers.google.com/protocol-buffers (besucht am 18.10.2023).

[GW20]
Dr. Ulrike Glück und Stephen Wu. Autonomous Vehicles Law and Regulation in China. 2024. URL: https://cms.law/en/int/expert-guides/cms-expert-guide-to-autonomous-vehicles-avs/china (besucht am 30. 09. 2024).

[GJW+20]
Magnus Gyllenhammar, Rolf Johansson, Fredrik Warg, DeJiu Chen, Hans-Martin Heyn, Martin Sanfridson, Jan Söderberg, Anders Thorsen und Stig Ursing. Towards an Operational Design Domain That Supports the Safety Argumentation of an Automated Driving System. Conference Paper, January 2020. URL: https://www.researchgate.net/publication/338969117 (besucht am 12.05.22).

[GWK+12]
Tom M. Gasser, Dietmar Westhoff, Gerhard Koniak, Ali Etemad, Christian Eymer, Steffen Faßbender und et al. “Rechtsfolgen zunehmender Fahrzeugautomatisierung“. BASt-Bericht, F 83. Bergisch Gladbach: Bundesanstalt für Straßenwesen (BASt). 2012.

[GT15]
Clinton Gormley und Zachary Tong. “*Elasticsearch: The Definitive Guide”*. O'Reilly Media. 2015

[HD04]
Anne M. Hickey and Alan M. Davis. „A Unified Model of Requirements Elicitation.“ In: Journal of Management Information Systems 20 (März 2004), S. 65–84.

[Hof18]
Hoffmann, Dirk W. Grundlagen der Technischen Informatik. 5., aktualisierte Auflage. Hanser Verlag, 2018.

[Hed12]
Hedtstück, U.: *Einführung in die Theoretische Informatik - Formale Sprachen und Automatentheorie*. überarbeitete Ausgabe. München: Oldenbourg Verlag, 2012. ISBN: 978-3-486-71404-3.

[Hof18]
Hoffmann, Dirk W. Grundlagen der Theoretischen Informatik. 5., aktualisierte Auflage. Hanser Verlag, 2018.

[HR04]
David Harel and Bernhard Rumpe. Meaningful Modeling: What’s the Semantics of ”Semantics”? IEEE Computer, 37(10):64–72, Oct 2004.

[Int11a]
International Organization for Standardization. Iso 26262-10:2011: Road vehicles - functional safety: Part 10: Guideline to iso 26262, 2011.

[Int11b]
International Organization for Standardization. Iso 26262-1:2011: Road vehicles - functional safety: Part 1: Vocabulary, 2011.

[Int11c]
International Organization for Standardization. Iso 26262-3:2011: Road vehicles - functional safety: Part 3: Concept phase, 2011.

[Int11d]
International Organization for Standardization. Iso 26262-4:2011: Road vehicles - functional safety: Part 4: Product development at the system level, 2011.

[Int11e]
International Organization for Standardization. Iso 26262-5:2011: Road vehicles - functional safety: Part 5: Product development at the hardware level, 2011.

[Int11f]
International Organization for Standardization. Iso 26262-6:2011: Road vehicles - functional safety: Part 6: Product development at the software level, 2011.

[Int11g]
International Organization for Standardization. Iso 26262-6:2011: Road vehicles - functional safety: Part 7: Supporting processes, 2011.

[Int11h]
International Organization for Standardization. Iso 26262-6:2011: Road vehicles - functional safety: Part 9: ASIL-oriented and safety-oriented analyses, 2011.

[Int19]
International Organization for Standardization. Iso/pas 21448 - road vehicles - safety of the intended functionality, 2019.

[Int22]
International Organization for Standardization. ISO 34502:2022 Road vehicles — Test scenarios for automated driving systems — Scenario-based safety evaluation framework. Geneva: ISO, 2022.

[Int23]
International Organization for Standardization. *ISO 34503:2023 Road Vehicles — Test scenarios for automated driving systems — Specification for operational design domain.* Geneva: ISO, 2023.

[Iso10]
ISO, IEC, IEEE. Iso/iec/ieee 24765:2010 - systems and software engineering: Vocabulary, 2010.

[Ito21]
Masao Ito. „ODD Description Methods for Automated Driving Vehicles and Verifiability for Safety.“ 2021.

[Itu21]
ITU-T, Hrsg. "X.680: Information Technology – Abstract Syntax Notation One (ASN.1): Specification of

Basic Notation." ITU-T Recommendation, 2021. URL: https://www.itu.int/rec/T-REC-X.680 (besucht am 20.10.2023).

[IZK+21]
Patrick Irvine, Xizhe Zhang, Siddartha Khastgir, Edward Schwalb und Paul Jennings. “A two-level abstraction odd definition language: Part i,” in 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC). IEEE, 2021, pp. 2614–2621.

[JMB+21]
Inga Jatzkowski, Till Menzel, Ansgar Bock und Markus Maurer. „A Knowledge-based Approach for the Automatic Construction of Skill Graphs for Online Monitoring.“ 2021.

[Kai21]
Bernhard Kaiser. „Application Story of ODD as part of Safety Assurance. The significance of a well-structured ODD specification for the AD Safety and SOTIF Process”. ASAM. 06.2021. URL: https://www.asam.net/index.php?eID=dumpFile&t=f&f=4303&token=3135965e578e5bb92a01725cd37823c3979da158 (besucht am: 23.10.23).

[KCG+20]
Markus Kremer, Sébastien Christiaens, Christian Granrath und Max-Arno Meyer. “Szenarien- und modellbasiertes Systems Engineering für das hochautomatisierte Fahren”. Wiesbaden : Vieweg (2020). Fachzeitschriftenartikel In: Automobiltechnische Zeitschrift : ATZ Band: 122 Heft: 12 Seite(n)/Artikel-Nr.: 16-21

[Kle17]
Martin Kleppmann. „Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems.” O'Reilly Media, 2017.

[Kni02]
John C. Knight. Safety critical systems: Challenges and directions. In Proceedings of the 24th International Conference on Software Engineering, ICSE ’02, pages 547–550, New York, NY, USA, 2002. ACM.

[Kra10]
Krahn, Holger. MontiCore: Agile Entwicklung von domänenspezifischen Sprachen im Software-Engineering. Shaker Verlag, Aachener Informatik-Berichte, Software Engineering Band 1, 2010. ISBN 978-3-8322-8948-5.

[KVK20]
Celina Kacperski, Tobias Vogel und Florian Kutzner. "Ambivalence in Stakeholders’ Views on Connected and Autonomous Vehicles." In HCI in Mobility, Transport und Automotive Systems. Automated Driving and In-Vehicle Experience Design, Lecture Notes in Computer Science, vol. 12212, Springer, Cham, 2020, pp. 46–57.

[KW16]
Philip Koopman and Michael Wagner. “Challenges in Autonomous Vehicle Testing and Validation”. In: SAE International Journal of Transportation Safety 4.1 (2016), pp. 2016–01–0128.

[KW24]
Philip Koopman und William Widen. "Redefining Safety for Autonomous Vehicles." arXiv preprint, 2024. URL: https://arxiv.org/pdf/2404.16768 (besucht am 24.09.24).

[LNG+20]
Chung Won Lee, Nasif Nayeer, Danson Evan Garcia, Ankur Agrawal und Bingbing Liu. „Identifying the Operational Design Domain for an Automated Driving System through Assessed Risk.“ 2020.

[LT18]
Nancy G Leveson und John P. Thomas. STPA Handbook. MIT Partnership for a Systems Approach to Safety. Massachusetts Institute of Technology. 2018.

[Mac22]
Angus MacKenzie.”Mercedes-Benz Drive Pilot EQS Autonomous Driverless First Drive Review.” 16. Mai 2023. URL: https://www.motortrend.com/news/mercedes-benz-drive-pilot-eqs-autonomous-driverless-first-drive-review/ (besucht am 24.10.2023).

[Mat24]
Jessica Mathews. Inside GM Cruise’s Self-Driving Car Accident in San Francisco—What Really Happened? Fortune, 16 May 2024. URL: https://fortune.com/2024/05/16/inside-gm-cruise-self-driving-car-accident-san-francisco-what-really-happened/ (besucht am 15.06.2024).

[MDR+24]
Marcel A. Mehlhorn, Hauke Dierend, Dr. Andreas Richter und Yuri A. W. Shardt, "A Stakeholder Analysis of Operational Design Domains of Automated Driving Systems," 2024 Design, Automation & Test in Europe Conference & Exhibition (DATE), Valencia, Spain, 2024.

[MS04]
Dinesh P. Mehta und Sartaj Sahni. „Handbook of Data Structures and Applications”. Chapman & Hall/CRC Computer & Information Science Series, 2004.

[MS20]
Robert Myers und Zeyn Saigol. „Design Considerations for ODD Ontology.“ März 2020

[MTW21]
Zicong Meng, Tao Tang, Guodong Wei und Lei Yuan. “Analysis of ATO System Operation Scenarios based on UPPAAL and the Operational Design Domain.” 2021.

[Nag79]
Manfred Nagl. Graph-Grammatiken: Theorie, Anwendungen, Implementierung. Vieweg, 1979.

[Nat20]
National Highway Traffic Safety Administration. Automated Driving Systems. https://www.nhtsa.gov/vehicle-manufacturers/automated-driving-systems, 2020.

[NHS+14]
Mikael Nybacka, Xiaogang He, Zhe Su, Lars Drugge, und Egbert Bakker. "Links between subjective assessments and objective metrics for steering und evaluation of driver ratings." Vehicle System Dynamics, vol. 52, sup1, 2014, pp. 31–50.

[OK23]
Matteo Oldoni und Siddartha Khastgir. “Introducing ODD-SAF: An Operational Design Domain Safety Assurance Framework for Automated Driving Systems.” In: Meyer, G., Beiker, S. (eds) Road Vehicle Automation 10. ARTSymposium 2022, 2023.

[Ope24]
OpenStreetMap contributors, Hrsg. “*OpenStreetMap”*. URL: https://wiki.openstreetmap.org/wiki/DE:Map_Features (besucht am 12.02.24).

[Ove24]
Overpass Turbo, Hrsg. *Overpass Turbo Query Tool*. 2024. URL: https://overpass-turbo.eu (besucht am 11.02.24).

[Peg19]
PEGASUS, Hrsg. Projekt zur Etablierung von generell akzeptierten Gütekriterien, Werkzeugen und Methoden sowie Szenarien und Situationen zur Freigabe hochautomatisierter Fahrfunktionen: Schlussbericht für das Gesamtprojekt. Laufzeit: 01.01.2016-30.06.2019. Ingolstadt, Audi AG, 2019.

[Pie02]
Benjamin C. Pierce. “Types and Programming Languages”. MIT Press, 2002.

[Plo81]
Gordon D. Plotkin. A Structural Approach to Operational Semantics. Technischer Bericht DAIMI FN-19, University of Aarhus, 1981.

[Pri15]
Prinz, Christoph. Entwicklung eines Compilers zur Programmierung von Sequenz-Generatoren. Bachelorarbeit. Duale Hochschule Baden-Württemberg Mannheim, 14. September 2015.

[Rei78]
Ray Reiter (1978). *On closed world data bases*. In H. Gallaire and J. Minker, editors, Logic and Data Bases, pp. 119–40. Plenum Publ. Co., New York.

[Reg08]
Ralf Regele. „Using Ontology-Based Traffic Models for More Efficient Decision Making of Autonomous Vehicles.“ 2008.

[RRS22]
Daniel Rohne, Dr. Andreas Richter und Dr. Edward Schwalb. "Implementing ODD as single point of knowledge to support the development of automated driving," 2022 IEEE International Conference on Systems, Man und Cybernetics (SMC), Prague, Czech Republic, 2022, pp. 1364-1370.

[RSR23]
Daniel Rohne, Dr. Edward Schwalb und Dr. Andreas Richter. "Determination of relevant SOTIF scenarios with the help of an Operational Design Domain." IFAC-PapersOnLine, Volume 56, Issue 2, 2023, Pages 4889-4895.

[Sae21]
SAE International, Hrsg. “Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles”. SAE Standard J3016, April 2021.

[Sal22]
Aniket Salvi. "Operational Design Domain – Narrow frame provides safety." Fraunhofer Institute for Cognitive Systems IKS, October 13, 2022. URL: https://safe-intelligence.fraunhofer.de/en/articles/odd-narrow-frame-provides-safety (besucht am 10.01.2024)

[San23]
San Francisco City Attorney Files Motion to Pump the Breaks on Driverless Cars*, NBC Bay Area (Aug.

18, 2023), [URL](https://www.nbcbayarea.com/news/local/san-francisco-city-attorney-driverless-car-expansion/3298221/).

[Sch21]
Peter Schlicht. „KI in der Automobilindustrie“. In: I. Knappertsbusch and K. Gondlach (Eds.), *Arbeitswelt und KI 2030*, Springer Fachmedien Wiesbaden, 2021. DOI: 10.1007/978-3-658-35779-5_29.

[SDC+21]
Chen Sun, Zejian Deng, Wenbo Chu, et al. „Acclimatizing the Operational Design Domain for Autonomous Driving Systems.“ 2021.

[Sei22]
Gabriel Seiberth. "Operational Design Domain als Schlüssel zum Erfolg." ATZelektronik, vol. 17, 2022, pp. 50–54.

[Sha24]
Ali Shakeri. "Formalization of Operational Domain and Operational Design Domain for Automated Vehicles." 2024. arXiv preprint: 2408.14481. Available at: https://arxiv.org/abs/2408.14481.

[SHC23]
Bigad Shaban, Michael Horn und Jeremy Caroll. San Francisco City Attorney Files Motion to Pump the Breaks on Driverless Cars. NBC Bay Area, 18. August 2023. URL: https://www.nbcbayarea.com/news/local/san-francisco-city-attorney-driverless-car-expansion/3298221/ (besucht am 23.09.2023).

[SIZ21]
Edward Schwalb, Patrick Irvine, X. Zhang, Siddartha. Khastgir und P. Jennings, “A two-level abstraction odd definition language: Part ii,” in 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC). IEEE, 2021, pp. 1669–1676.

[SMM21]
Markus Steimle, Till Menzel und Markus Maurer. „Toward a Consistent Taxonomy for Scenario-Based Development and Test Approaches for Automated Vehicles.“ 2021.

[Smu12]
Raymond R Smullyan. First-order logic. Vol. 43. Springer Science & Business Media, 2012.

[Som11]
Ian Sommerville. Software Engineering. 9th Edition. Boston: Addison-Wesley, 2011.

[SRR22]
Edward Schwalb, Andreas Richter und Daniel Rohne. "Validating Autonomous Behaviors against Partially Specified Ambiguous Requirements," 2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Prague, Czech Republic, 2022, pp. 1342-1349.

[SS71]
Dana Scott, Christopher Strachey. Toward a Mathematical Semantics for Computer Languages. Programming Research Group Technical Monograph PRG-6, Oxford Univ. Computing Lab., 1971.

[SSJ21]
Andrew Smart, Chess Stetson und Kiran Jesudasan. „Autonomous Vehicle Safety Assessment with Fully Quantified ODDs.“ 2021.

[SWT+22]
Aniket Salvi, Gereon Weiss, Mario Trapp, Fabian Oboril und Cornelius Buerkle. “Fuzzy Interpretation of Operational Design Domains in Autonomous Driving.”2022.

[SZ21]
Mark Schaub und Atticus Zhao. China´s Legislation on Autonomous Cars Rolls Out. April 2021. URL: https://www.chinalawinsight.com/2021/04/articles/corporate-ma/chinas-legislation-on-autonomous-cars-rolls-out/ (besucht am 04.10.2023).

[TBB+21]
Abhimanyu Tonk, Abderraouf Boussif, Julie Beugin und Simon Collart-Dutilleul. „Towards a Specified Operational Design Domain for a Safe Remote Driving of Trains.“ 2021.

[UH21]
Constance Ugé und Stephanie Hochgeschurz. „Learning to swim - how operational design parameters determine the grade of autonomy of ships.“ 2021.

[UMR+15]
Simon Ulbrich, Till Menzel, Andreas Reschka, Fabian Schuldt und Markus Maurer. Defining and substantiating the terms scene, situation und scenario for automated driving. In 2015 IEEE 18th International Conference on Intelligent Transportation Systems, pages 982–988, 2015.

[Uni14a]
Universität Bremen. Beschreibungslogiken Kapitel 6: Grundlagen [PDF](https://www.informatik.uni-bremen.de/tdki/lehre/ss14/bl/Kapitel2.pdf), 2014.

[Uni14b]
Universität Bremen. Beschreibungslogiken Kapitel 6: A-Boxen und Anfragebeantwortung [PDF](https://www.informatik.uni-bremen.de/tdki/lehre/ss14/bl/Kapitel6.pdf), 2014.

[W3c08]
W3C, Hrsg. "Extensible Markup Language (XML) 1.0 (Fifth Edition)." W3C Recommendation, November 2008. URL: https://www.w3.org/TR/xml/ (besucht am 18.10.2023).

[Way20]
Waymo, Hrsg. "Waymo Safety Case Approach." Waymo, 2020. URL: https://storage.googleapis.com/waymo-uploads/files/documents/safety/Waymo%20Safety%20Case%20Approach.pdf (besucht am 24.02.23).

[Wil97]
David S. Wile. Abstract Syntax from Concrete Syntax. In Proc. of International Conference on Software Engineering (ICSE) 1997. ACM, 1997.

[WRO+21]
Bowen Weng, Linda Jenny Capito Ruiz, Umit Ozguner und Keith Redmill. „Towards Guaranteed Safety Assurance of Automated Driving Systems with Scenario Sampling: An Invariant Set Perspective.“ 2021.

[WW03]
Matthias Weber und Joachim Weisbrod. "Requirements engineering in automotive development: experiences and challenges." IEEE Software, vol. 20, no. 1, 2003, pp. 16–24.

[Yam21]
YAML Language Development Team, Hrsg. “YAML Ain’t Markup Language (YAML™) version 1.2” 2021. URL: https://yaml.org/spec/1.2.2 (besucht am 10.10.23).

[ZKH20]
Lena Ziegler, Frank Krämer und Nils Haustein. “AI and Big Data Management for Autonomous Vehicles”. ATZ Electron Worldw 15, 40–45 (2020).

[ZW22]
Markus Zimmermann und Olivier de Weck. “Formulating Engineering Systems Requirements." In Handbook of Engineering Systems Design, A. Maier et al. (eds.), Springer Nature Switzerland AG, 2022, pp. 441-491.

[ZZM19]
Hengyu Zhao, Yubo Zhang, Pingfan Meng, Hui Shi, Li Erran Li, Tiancheng Lou und Jishen Zhao. "Towards Safety-Aware Computing System Design in Autonomous Vehicles." arXiv preprint arXiv:1905.08453, 2019. URL: https://arxiv.org/abs/1905.08453 (besucht am 04.07.23).