Gemeinsamer Schlussbericht für das Projekt

SET Level

Simulationsbasiertes **E**ntwickeln und **T**esten von automatisiertem Fahren

Laufzeit: 01.03.2019 – 31.10.2022

Gefördert durch:

Bundesministerium für Wirtschaft und Klimaschutz

aufgrund eines Beschlusses des Deutschen Bundestages

Informationen zum Dokument

Titel	Gemeinsamer Schlussbericht für das Projekt SET Level
Datum	30.04.2023
Autoren	Stefan Rude BMW Group Projektkoordination und Leitung Teilprojekt 4 stefan.rude@bmw.de Frank Köster DLR e.V. – Institut für KI-Sicherheit Projektkoordination frank.koester@dlr.de Hardi Hungar DLR e.V. – Institut für Verkehrssystemtechnik Leitung Teilprojekt 1 hardi.hungar@dlr.de Heinz Sachsenweger ZF Friedrichshafen AG Leitung Teilprojekt 2 heinz.sachsenweger@zf.com Julien Bou Robert Bosch GmbH Leitung Teilprojekt 3 julien.bou@de.bosch.com Hans-Martin Heinkel Robert Bosch GmbH Leitung Teilprojekt 3 hans-martin.heinkel@de.bosch.com Carsten Franke PROSTEP AG Leitung Arbeitspaket 4.1 Thomas Platzer BMW Group Leitung Arbeitspaket 4.2 Martin Fischer DLR e.V. – Institut für Verkehrssystemtechnik Leitung Arbeitspaket 4.3 Arndt-Michael Meyer ETAS GmbH Leitung Arbeitspaket 4.4 Christian Bühler Leitung Teilprojekt 5 PROSTEP AG christian.buehler@prostep.com Florian Plötzwich DLR e.V. – Institut für Verkehrssystemtechnik Projektbüro florian.ploetzwich@dlr.de

Geschlechtergerechte Sprache

Die in diesem Dokument verwendeten Personenbezeichnungen beziehen sich immer gleichermaßen auf Personen jeglichen Geschlechts. Auf eine Doppelnennung und gegenderte Bezeichnungen wird zugunsten einer besseren Lesbarkeit verzichtet.

Inhaltsverzeichnis

Abkürzungsverzeichnis

ADAS	Advanced Driver Assistance Systems
AEB	Automated Emergency Braking
AP	Arbeitspaket
ASE	Abschlussevent
ASAM	Association for Standardization of Automation and Measuring Systems
ASCS	Automotive Solution Center for Simulation
ASM	Automotive Simulation Models
BMWK	Bundesministerium für Wirtschaft und Klimaschutz
CCB	Change Control Board
CI	Configuration Item
CSV	Comma Separated Values
CMP	Credible Modeling Process
CSP	Credible Simulation Process
DiL	Driver in the Loop
EA	Enterprise Architect
EAM	Enterprise Architecture Modeling
FFT	Fast Fourier Transformation
FMI	Functional Mockup Interface
FMU	Functional Mockup Unit
GPS	Global Positioning System
GPU	Graphics Processing Unit
GT	Ground Truth
HAD	Highly Automated Driving
HF	Hochfrequenz
HiL	Hardware in the Loop
HSV	Hue (Farbwert bzw. -winkel), Saturation (Farbsättigung), Value (Hellwert)
IP	Intellectual Property
KI	Künstliche Intelligenz
KMU	Klein- und mittelständisches Unternehmen
KPI	Key Performance Indicator
LIDAR	Light Detection And Ranging
LoD	Level of Detail
MBSE	Model Based Systems Engineering
MiL	Model in the Loop
MoC	Models of Computation
MS	Meilenstein
ODD	Operational Design Domain
OEM	Original Equipment Manufacturer
OpenX	OpenSCENARIO, OpenDRIVE, OpenCRG etc.
OSI	Open Simulation Interface
OSLC	Open Services for Lifecycle Collaboration
OSMP	OSI Sensor Model Packaging
OSPA	Optimal Sub Pattern Assignment Metrik
PEGASUS	Projekt zur Etablierung von generell akzeptierten Gütekriterien, Werkzeugen und Methoden sowie Szenarien und Situationen zur Freigabe hochautomatisierter Fahrfunktionen
PET	Post Encroachment Time

PEW	PEGASUS Expert Workshop
PSM	Physical Sensor Models
POC	Proof of Concept
RFLP	Requirements, Functions, Logics, Physics
RGB	Rot, Grün, Blau
ROS	Robot Operation System
SAE	Society of Automotive Engineers
SaFAD	Safety First for Automated Driving
SET	Simulation Entwicklung Testen (Simulation Engineering Testing)
SiL	Software in the Loop
SME	Small and Medium-sized Enterprises
SRMD	Simulation Ressource Meta Data
STMD	Simulation Task Meta Data
SuT	System under Test
SSP	System Structure and Parametrization
SysML	Systems Modeling Language
UML	Unified Modelling Language
SUC	Simulation Use Case
TCP	Transmission Control Protocol
TP	Teilprojekt
TTC	Time to Collision
VVM	Verifikations- und Validierungsmethoden
VVMethoden	Verifikations- und Validierungsmethoden
V&V	Verifikation und Validierung

1 Kurzdarstellung

1.1 Projektziele und Ergebnisfelder

Die Automatisierung von Fahrzeugen in Kombination mit der Einführung vernetzter Fahrzeugfunktionen leistet einen Beitrag zur Reduktion von Unfallzahlen und zur Erhöhung der Insassen-Sicherheit (vgl. Strategie der Bundesregierung zum automatisierten und vernetzten Fahren). Zudem wird eine Effizienzsteigerung im Straßenverkehr ermöglicht, u. a. im Sinne der Energieeffizienz und verbesserten Ausnutzung vorhandener Verkehrsinfrastrukturen.

Insbesondere Fahrzeuge höherer Automatisierungslevel, die Strecken auch fahrerlos zurücklegen können, werden zukünftig wichtige Bausteine in neuartigen Mobilitätsangeboten bzw. Transportdiensten sein. Zusätzlich kann der Komfort für den Fahrer bzw. Fahrzeugnutzer und die Passagiere durch Automatisierung und vernetzte Fahrzeugfunktionen deutlich verbessert werden.

Seit einigen Jahren steht die Funktionskonzeption und -Entwicklung für automatisierte Fahrzeuge im Mittelpunkt von Forschung und Entwicklung, sodass bereits heute für verschiedene verkehrliche Situationen automatisierte und vernetzte Fahrzeugfunktionen demonstriert werden können. Die reine Demonstration ist für eine Freigabe und Zulassung dieser Fahrzeuge jedoch nicht hinreichend. Diese Thematik ist auch ausdrücklich im Fachprogramm „Neue Fahrzeug- und Systemtechnologien“ des Bundesministeriums für Wirtschaft und Energie (BMWi, 2015) als einer der Schwerpunkte der Förderung genannt. So stand u. a. im Projekt PEGASUS[1] das Testen automatisierter und vernetzter Fahrzeugfunktionen im Fokus, insbesondere wurde das automatisierte Fahren gemäß SAE Level 3 auf der Autobahn betrachtet.

Zukünftig können insbesondere simulationsbasierte Methoden und Werkzeuge zum Testen automatisierter Fahrfunktionen die effiziente Testdurchführung auch für komplexe Fahrzeugfunktionen unterstützen. Erste Ansätze hierfür wurden bereits im Rahmen des Projekts PEGASUS für Autobahnfunktionen aufgebaut – u. a. existieren hier verschiedene Simulations-Modelle von Sensoren unterschiedlicher Messprinzipien, Fahrzeugautomationen, Ansätze zur Umwelt- und Systembeschreibung sowie zur Koppelung verschiedener Simulationsmodelle. Diese Ergebnisse wurden im Projekt SET Level aufgegriffen. Ziel im Projekt SET Level war, dass generische Simulationsplattformen entstehen, die für einen Einsatz bei der Entwicklung und dem Testen von Fahrfunktionen höherer Automationsstufen und höherer Komplexität urbaner Verkehrssituationen geeignet sind. Die entstandenen Einzelwerkzeuge sollten einfach in Werkzeugketten integriert werden können, welche die Entwicklung (u. a. Anforderungsermittlung und -präzisierung) und das effiziente Testen (u. a. Verifikation und Validierung) von höheren Automatisierungsleveln Fahrzeugsystemen in urbanen Räumen unterstützen bzw. effizient ermöglichen. In diesem Zusammenhang wurden

- Plattformen zur Anforderungsermittlung für automatisierte und vernetzte Straßenfahrzeuge im urbanen Raum und für das (entwicklungsbegleitende) Testen konzipiert. Dabei sind Einzelergebnisse, die z. B. auf Komponentenebene erzielt wurden, im Hinblick auf eine Gesamtfahrzeugbeurteilung verwendbar, um später die Homologation und Freigabe des Gesamtfahrzeugs durch Simulationsanteile mittelbar zu unterstützen.

[1] siehe https://www.pegasusprojekt.de/

- Beiträge zur Standardisierung simulationsbasierter Entwicklungs- und Testwerkzeuge erarbeitet - u. a. zur/zum
  - Spezifikation von Szenarien und Anwendungsfällen
  - Spezifikation von Modellbeschreibungen und Modellen (u. a. sind hier Granularität und Integrationsfähigkeit zu nennen)
  - Repräsentation von Modellen und Parametrisierungen in Bibliotheken
  - Modellintegration und Handhabung von Integrationsproblemen
  - Management von Simulationsdaten
  - Konfigurationsmanagement

- Plattformen instanziiert, die zur Ergebnisdarstellung und -demonstration genutzt und insbesondere auch als Aufsatzpunkt für andere und nachfolgende Projekte wie z. B. VVMethoden[2] oder KI-Absicherung[3] herangezogen werden können.

Die erzielten Projektergebnisse können von den beteiligten Industriepartnern (OEMs und Zulieferer) aufgegriffen werden. Ebenfalls können die Projektergebnisse von KMU und IT-Vendoren zur Bereitstellung notwendiger Werkzeuge (IT-Tools) unter Berücksichtigung der entwickelten Standards zur Spezifikation von Schnittstellen genutzt werden. Gleiches gilt für weitere öffentlich geförderte Projekte, wie z. B. VVMethoden und KI-Absicherung.

Hauptziel des Projekts SET Level war die Unterstützung aller Firmen und Institutionen, die von der Erarbeitung neuer Vorgehensweisen (Methoden und Standards) zur Zulassung automatisierter und vernetzter Fahrzeuge betroffen sind. Dieses Hauptziel greift direkt einen der Schwerpunkte des Fachprogramms „Neue Fahrzeug- und Systemtechnologien“ Bundesministeriums für Wirtschaft und Energie (BMWi, jetzt Bundeministerium für Wirtschaft und Klimaschutz, BMWK) auf. Dieses Fachprogramm detaillierte Inhalte der zugehörigen Förderrichtlinie (BMWi, 2015, geändert 2018 und 2021). Im Schwerpunkt „Angepasste Testverfahren und Validierung“ wurde die zukünftige Bedeutung der Simulation für den Test und die Absicherung als Ergänzung zu der prototypischen Erprobung auf Testgeländen und im freien Verkehr hervorgehoben. Für die Gewährleistung der funktionalen Sicherheit automatisierter Systeme waren insbesondere neuartige Indikatoren, Testverfahren und -methoden zu entwikkeln. Dem trug das Projekt SET Level mit den oben aufgelisteten Resultaten Rechnung. Hervorzuheben ist, dass die nach dem Projektansatz konzipierten Simulationswerkzeuge mitsamt den zugehörigen Komponenten und Verfahren flexibel an verschiedenste Simulationsaufgaben angepasst werden können und aufgrund der Verwendung von offenen Standards breiten Nutzerkreisen zugänglich sind. Damit stellen sie einerseits selber einen Baustein in der vom Zuwendungsgeber angestrebten Unterstützung zur digitalen Transformation dar. Andererseits unterstützen sie in den realisierten Ausprägungen die Entwicklung von Lösungen zum automatisierten Fahren, welche eine der beiden Säulen des Förderprogramms darstellt.

1.2 Voraussetzungen für das Vorhaben

Das Projekt SET Level wurde als Folgeprojekt des Projekts PEGASUS konzipiert. Das Projekt PEGASUS (Projekt zur Etablierung von generell akzeptierten Gütekriterien, Werkzeugen und Methoden sowie Szenarien und Situationen zur Freigabe hochautomatisierter Fahrfunktionen) mit einer Laufzeit von 2016 bis 2019 hatte das Ziel, ein methodisches Rahmenwerk für die Absicherung automatisierter Fahrzeuge (SAE Levels 3 und 4) im Einsatzgebiet

[2] siehe https://www.vvm-projekt.de/projekt

[3] siehe https://www.ki-absicherung-projekt.de/projekt

Autobahnen zu entwickeln (siehe Abbildung 1). Urbane Situationen wurden am Rande ebenfalls behandelt. SET Level wurde dabei in enger Abstimmung mit dem weiteren Folgeprojekt zu PEGASUS definiert – dem Projekt VVMethoden. Auch wenn die Projekte unabhängig voneinander durchführbar sein sollten, wurden von vornherein die enge Verzahnung der Projekte VVMethoden und SET Level sowie die integrierte Nutzung ihrer Ergebnisse mitgedacht.

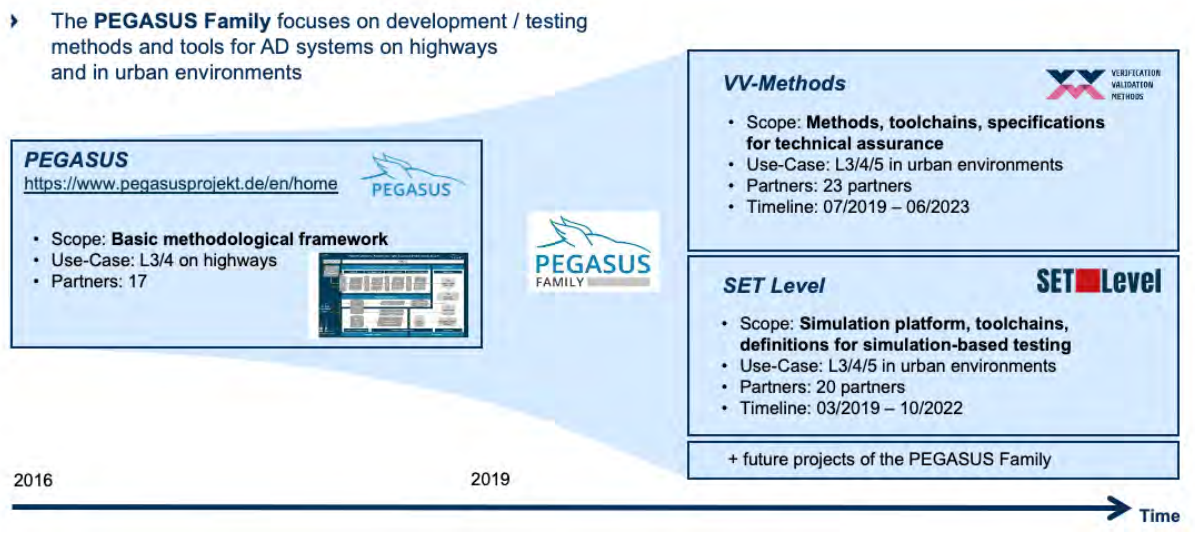


Abbildung 1: Projekte der PEGASUS-Familie

Das Projekt SET Level wurde speziell für den Einsatz von Simulation beim Entwickeln und Testen hoch automatisierter Fahrfunktionen in urbanen Umgebungen konzipiert. Dabei wurden wesentliche Partner aus der Automobil- und ihrer Zuliefererindustrie, IT-Vendoren und Partner aus der relevanten Landschaft der Forschungseinrichtungen zusammengeführt, um Kompetenzen, Methoden und Technologien gemeinsam aufzubauen (siehe Abbildung 2).

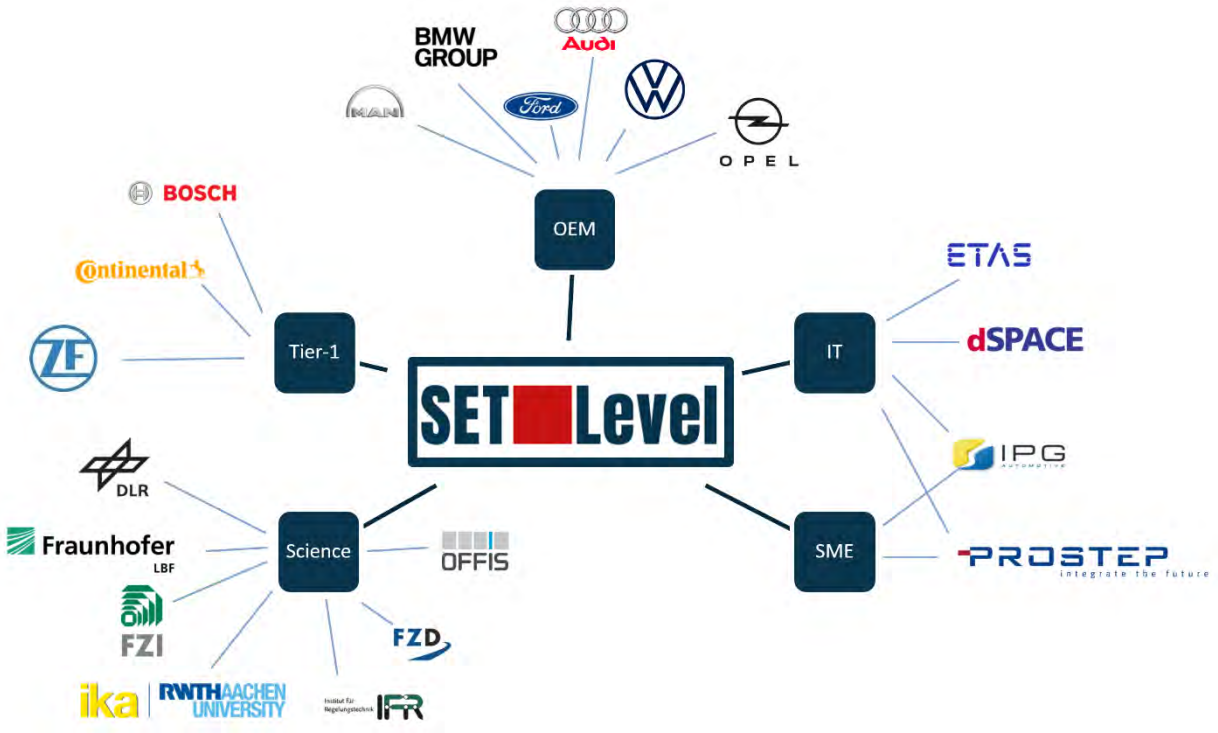


Abbildung 2: Zusammensetzung der Partner des Projekts SET Level

Tabelle 1: Projektpartner SET Level

Bezeichnung	Kurzbezeichnung	Typ
DLR e.V. – Institut für Verkehrssystemtechnik und Institut für KI-Sicherheit	DLR	Forschung (Science)
TU Darmstadt – Fachgebiet Fahrzeugtechnik	FZD	Forschung (Science)
Fraunhofer LBF	LBF	Forschung (Science)
FZI Forschungszentrum Informatik	FZI	Forschung (Science)
OFFIS – Institut für Informatik*	OFFIS	Forschung (Science)
RWTH Aachen – Institut für Kraftfahrzeuge (ika)	ika	Forschung (Science)
TU Braunschweig – Instituts für Regelungstechnik (IfR)	IfR	Forschung (Science)
dSpace GmbH	dSpace	IT
ETAS GmbH	ETAS	IT
IPG Automotive GmbH	IPG	KMU (SME), IT
PROSTEP AG	PROSTEP	KMU (SME), IT
AUDI AG	Audi	OEM
BMW AG	BMW	OEM
Ford-Werke GmbH	Ford	OEM
MAN Truck & Bus AG	MAN	OEM
Opel Automobile GmbH	Opel	OEM
Volkswagen AG	VW	OEM
ADC Automotive Distance Control Systems GmbH (ein Unternehmen des Continental Konzerns)	ADC	Tier 1
Robert Bosch GmbH	Bosch	Tier 1
ZF Friedrichshafen AG	ZF	Tier 1

*Hinweis: Das OFFIS – Institut für Informatik ist am 01.01.2022 in das DLR-Institut Systems Engineering für zukünftige Mobilität übergegangen. Die verbleibenden Arbeiten im Projekt SET Level wurden entsprechend vom DLR weitergeführt.

1.3 Planung und Ablauf

Vorgehen

Grundansatz zur Erschließung und Bearbeitung der komplexen Fragestellung bildete ein iteratives Vorgehen unter schrittweiser Komplexitätssteigerung zur gemeinsamen Durchdringung und Entwicklung des Themas (siehe Abbildung 3).

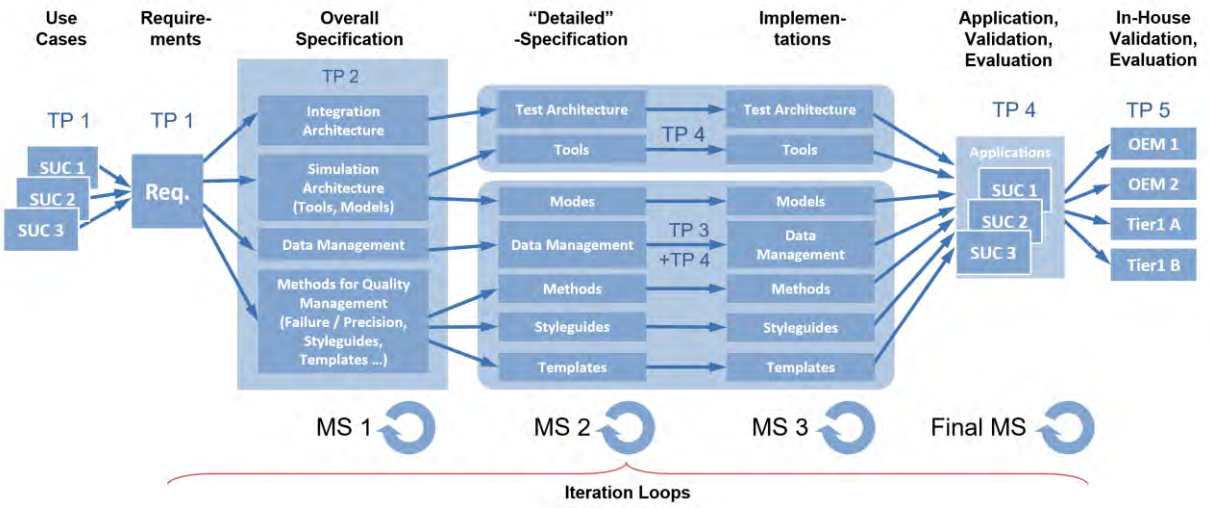


Abbildung 3: Schnelle Iterationen in Projektmeilensteinen

Als Erfolgsmerkmal und -nachweis wurden Simulation Use Cases (SUCs) konzipiert. Um einen Erfolgsnachweis und eine Basis für die Reflexion der gewählten Ansätze zu schaffen, wurden zu definierten Meilensteinen Demonstratoren implementiert und evaluiert. Dieses Vorgehen war entscheidend für ein schnelles Lernen und für den Nachweis, dass die Grundideen des Projekts SET Level – die modularisierten und auf Standards basierenden Architekturen von Simulationstools – auch wirklich funktionieren.

Die Arbeit erfolgte in einer Vielzahl von Teilprojekten (siehe Abbildung 4). Ausgehend von den Use Cases wurden im Teilprojekt 1 (TP 1) Anforderungen (Requirements) an Simulationstools spezifiziert. Inhaltlich fand die Konzeption (Specification) des modularen und auf Standards basierten Aufbaus von Simulationstools im Teilprojekt 2 (TP 2) statt. Die detaillierte Spezifikation notwendiger Prozesse und die Implementierung von Simulationsmodellen wurde im Teilprojekt 3 (TP 3) durchgeführt. Die Ausführung von Simulationen auf unterschiedlichen Simulationstools (von der Implementierung bis zur Evaluation) fand in Teilprojekt 4 (TP 4) statt. Die Sicherstellung der industriellen Anwendbarkeit erfolgte in Teilprojekt 5 (TP 5) durch regelmäßige Bewertungen des aktuellen Entwicklungsstands sowie entsprechendes Feedback. BMW und das DLR koordinierten das Projekt als Tandem aus Wirtschaft und Wissenschaft. Die Arbeit erfolgte in verschiedenen, miteinander interagierenden Teilprojekten (siehe Abbildung 4).

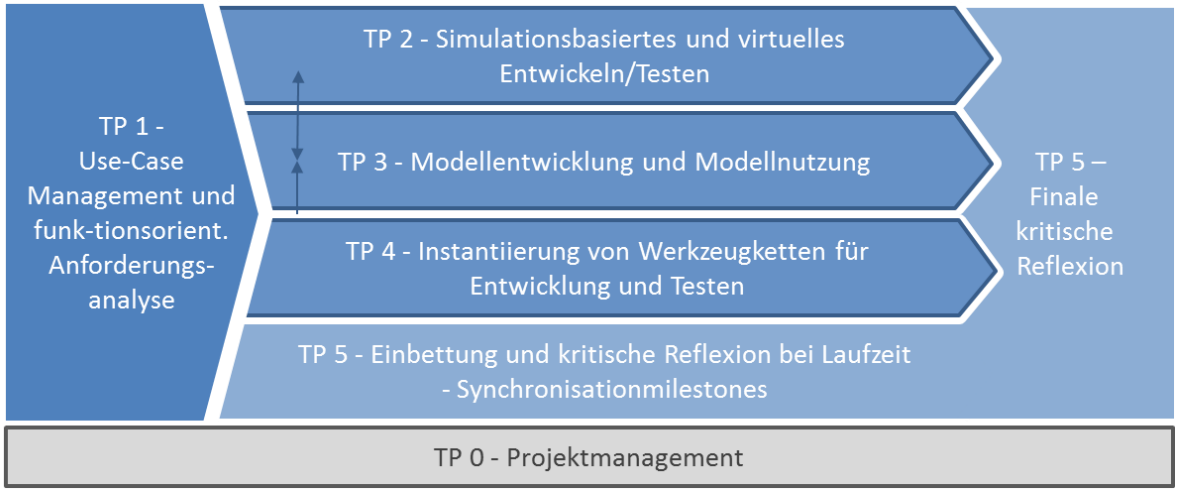


Abbildung 4: Aufbau der Teilprojekte des Projekts SET Level

Zeitplan

In Abbildung 5 ist die Gesamtübersicht der Projektlaufzeit dargestellt. Die Farben sind qualitative Intensitätsangaben, grob eingeteilt in geringe, mittlere und hohe Kapazitätsbedarfe der jeweiligen Teilprojekte. Wie zu erkennen ist, hatte Teilprojekt 1 als Anforderungs- und Synchronisationsprojekt zu VVMethoden einen starken Fokus im ersten Jahr und benötigte im weiteren Projektverlauf immer weniger personelle Ressourcen, um die Kommunikation zum Projekt VVMethoden zu gewährleisten. Die Teilprojekte 2 und 3 hatten ihre Belastungsspitze beginnend mit Q3 über den größten Teil des zweiten Projektjahres. Teilprojekt 4 hatte einen gleichmäßigen Kapazitätsbedarf über die Projektlaufzeit, während der Kapazitätsbedarf von Teilprojekt 5 über die Projektlaufzeit immer höher wurde und gegen Projektende am höchsten war.

In Abbildung 5 sind drei Gesamtmeilensteine jeweils zum Ende eines jeden Projektjahres verzeichnet, deren Hauptergebnisse als Demonstrator, Vorversion und Endversion der Werkzeugketten interpretiert werden können.

	Projektjahr 1				Projektjahr 2				Projektjahr 3				Projektjahr 4		
Arbeitspaket	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15
0															
1.1															
1.2															
2.1															
2.2															
2.3															
3.1															
3.2															
3.3															
4.1															
4.2															
4.3															
4.4															
5.1															
5.2															
5.3															

Abbildung 5: Zeitplanung mit Meilenstein- und Abschlussberichten

Als Erfolgsmerkmal war die Konzeption von **Simulation Use Cases (SUCs)** und deren Implementierung in Demonstratoren entscheidend für schnelles Lernen und den Nachweis, dass

die Grundidee des Projekts SET Level, die modularisierte und auf Standards basierende Architektur von Simulationstools, auch wirklich funktioniert.

Als zielführend hat sich die Organisation sogenannter Quartalstreffen bewährt, bei denen alle Projektbeteiligten nach vorgegebener Agenda ihre jeweiligen Fortschritte berichtet und das jeweils weitere Vorgehen mit den Projektpartnern diskutiert und abgestimmt haben. Im ersten Projektjahr konnten alle Quartalstreffen durch physische Treffen vor Ort organisiert werden.

Ein Erfolgsfaktor war die Einrichtung eines gemeinsamen Repositories für Dokumente, Modelle und weitere Artefakte auf der Basis von GitLab.

Ab dem zweiten Projektjahr waren corona-bedingt zumeist nur noch virtuelle Treffen möglich. Bei diesen Treffen haben die Nutzung von Microsoft Teams als Kommunikationsplattform und die Einführung von Miro als gemeinsame Arbeitsplattform erheblich dazu beigetragen, eine substanzielle Kommunikation führen zu können. Insbesondere in Phasen der gemeinsamen Ideenfindung sind in diesen Umgebungen wichtige Arbeitsschritte durchgeführt worden.

1.4 Gesamthafter Ergebnisüberblick

Das Projekt SET Level hat wesentliche Voraussetzungen dafür geschaffen, modular aufgebaute Simulations-Toolketten auf der Basis von IT-Standards für die Entwicklung und das Testen automatisierter Fahrfunktionen zu schaffen. Dabei wurde das szenarienbasierte Testen als zentrales Konzept genutzt und weiterentwickelt. Von Projektbeginn an galt hierbei die Prämisse, Tests auf mehreren unterschiedlichen Testinstanzen umzusetzen (Multi Pillar Approach) und diejenigen Testinstanzen in den Betrachtungsumfang von SET Level aufzunehmen, die szenario- und simulationsbasiertes Testen erlauben (siehe Abbildung 6).

Unter den im Projekt verwendeten Testinstanzen ist insbesondere die von BMW im Rahmen eines Open-Source Projekts betreute Umgebungssimulation openPASS zu erwähnen. openPASS wurde im Projekt insbesondere für Fragestellungen der Wirksamkeits- und Kritikalitätsanalyse (Effectiveness und Criticality Analysis) eingesetzt.
Neben openPASS wurden weitere Implementierungen aus dem Bereich Software-in-the-loop (SiL) im Projekt betrachtet. Zudem wurden Demonstratoren für Model-in-the-loop (MiL) Simulationen aufgebaut. Beim betrachteten Testobjekt handelte es sich beispielsweise um die automatisierte Fahrfunktion eines Fahrzeugs und um entsprechende Sensormodelle (Kamera, Radar, Lidar).

Basierend auf der Tatsache, dass eine automatisierte Fahrfunktion durch die Steuerung des zugehörigen Fahrzeugs zumeist Einfluss auf die Umgebungssimulation nimmt, wurde diese Umgebungssimulation mit einer bidirektionalen Schnittstelle in MiL-Testinstanzen integriert (Closed Loop). Der Ausgang von Sensormodellen hingegen kann zumeist unabhängig von der Umgebungssimulation ausgewertet werden. Daher wurden entsprechende Modelle in MiL-Testinstanzen für Komponententests zumeist mittels unidirektionaler Schnittstellen (Open Loop) integriert.

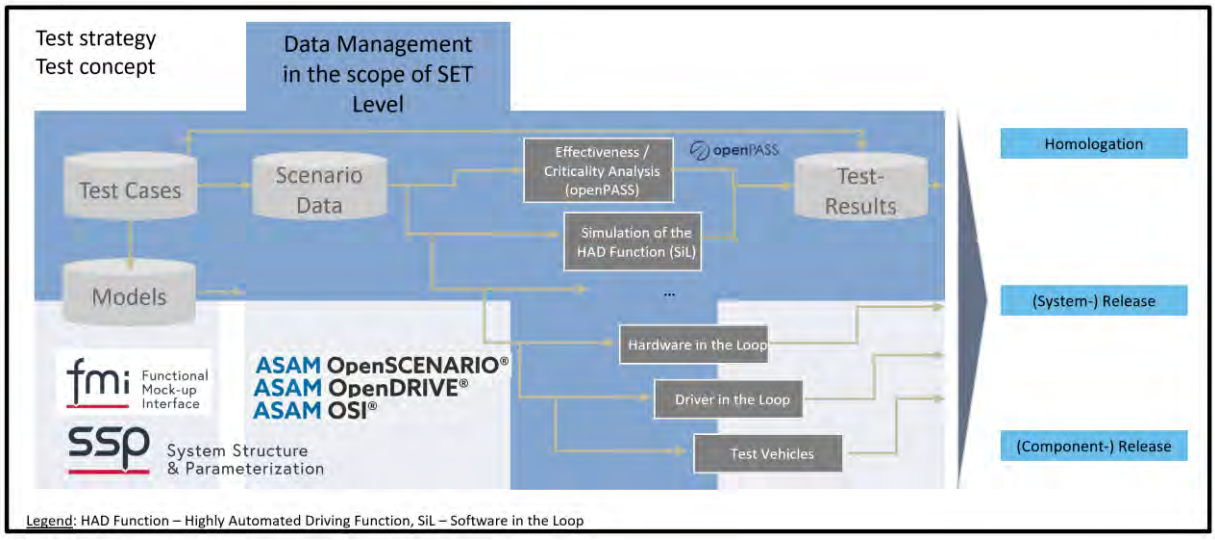


Abbildung 6: Grundansatz des Projekts SET Level

Szenarien müssen gleichermaßen auch für hardwarebasierte Tests, Hardware-in-the-Loop (HiL), Driver-in-the-Loop (DiL) wie auch für Testfahrzeuge auf Erprobungsgeländen oder auf öffentlichen Straßen nutzbar sein. Die Ergebnisse aller Testinstanzen werden zusammengefasst und für Homologation, Systemfreizeichnung und Komponentenfreigaben genutzt. Hier leisten simulationsbasierte Testinstanzen ihren Beitrag.
Grundlage für die Gestaltung und Umsetzung von Tests und deren Zuordnung zu spezifischen Testinstanzen bilden Teststrategie und Testkonzept einer Produktentwicklung. Diese müssen aus den Anforderungen an das künftige Produkt abgeleitet werden. Anforderungen sind hierbei idealerweise direkt mit entsprechenden Testfällen zu verknüpfen.
Das Projekt SET Level hat sich dediziert mit szenarienbasierten Testfällen für die Entwicklung und die Absicherung des automatisierten Fahrens befasst. Für verschiedene Simulationsaufgaben wurden erforderliche Simulationsmodelle beispielhaft zusammengestellt. Diese Modelle wurden bevorzugt durch Standards beschrieben. Ein geeigneter Standard zur Vereinheitlichung der Integration von Modellen ist das Functional Mockup Interface (FMI). Ein Standard zur Beschreibung von komplexen Systemen, bestehend aus mehreren, durch FMI beschriebene Functional Mockup Units (FMUs), bildet System Structure and Parametrization (SSP). SSP erlaubt zudem die Parametrierung von FMUs.
Komplementiert werden die genannten Standards durch Formate zur Beschreibung von Eingangsdaten der szenarienbasierten Simulation. Hierzu zählen ASAM OpenSCENARIO und ASAM OpenDRIVE. ASAM Open Simulation Interface (OSI) ergänzt die zuvor genannten Formate in der Beschreibung von anwendungsspezifischen Datenschnittstellen. Die genannten ASAM Standards wurden im Projekt SET Level eingesetzt und weiterentwickelt. Entstandene Projektergebnisse und gesammelte Erfahrungen bei der Nutzung der Standards wurden in die Standardisierungsprojekte zurückgespiegelt. Ebenfalls wurde die von ASAM vorgeschlagene Grundstruktur simulationsbasierter Entwicklungs- und Testwerkzeuge übernommen und auf die Bedürfnisse des Projekts hin weiterentwickelt (siehe Generic Open Analysis/Testing Architecture in Abbildung 14).
Im Fokus des Projekts SET Level stand insbesondere die Durchführung von szenarienbasierten- und simulationsbasierten Tests auf den in Abbildung 6 blau unterlegten Testinstanzen. Die beschriebenen Tests und die zugehörigen Artefakte (Eingangsdaten und Simulationsmodelle) sollten konsistent über die verschiedenen Testinstanzen hinweg umgesetzt bzw. integriert werden. Dazu ist es erforderlich, ebenso wie für sämtliche involvierte Software- und Hardwarekomponenten, spezifische Ausprägungen derart zu spezifizieren, versionieren und

dokumentieren, dass die Entstehung von Testergebnissen zu jeder Zeit nachvollzogen werden kann (Traceability). Aus diesem Grund wurden im Projekt SET Level auch Themen beleuchtet, die der Bereitstellung einer geeigneten Infrastruktur für das simulationsbasierte Testen dienen, z. B. die Daten- und Versionsverwaltung (Data Management) in Repositories, die Erstellung und Verwendung von Metadaten und die Etablierung eines Prozesses für die konsistente und nachvollziehbare Durchführung von simulationsbasierten Tests (Credible Simulation Prozess, CSP).

Die Arbeitsschwerpunkte im Projekt SET Level wurden in einer kompakten Darstellung angeordnet und eingerahmt von den wesentlichen Eckpunkten der simulationsbasierten Engineering-Aufgabe sowie der Nutzung von Standards (siehe Abbildung 7).

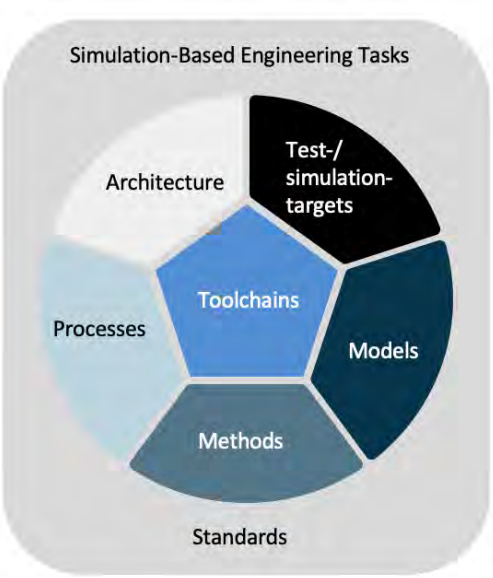


Abbildung 7: Kompakte Darstellung der Hauptarbeitsgebiete des Projekts SET Level

Die Simulationstoolkette (Toolchain) ist zentraler Bestandteil der kompakten Darstellung. Diese Toolkette wird von der simulationsbasierten Engineeringaufgabe, der sie beschreibenden Simulationsziele (Simulation Targets), der dafür notwendigen Architektur und den innerhalb der Architektur (Architecture) erforderlichen Simulationsmodellen (Models) bestimmt, sowohl für das Testobjekt (System under Test) als auch für die Betriebsumgebung (Operational Environment). Prozesse (Processes) unterstützen die Auswahl und den Aufbau von Simulationstoolketten und die Durchführung von Simulationsaufgaben. Methoden und Standards befähigen die Tools zur Ausführung von Simulationen.

Der Grundansatz des Projekts SET Level wurde in einer Matrix-Darstellung visualisiert (siehe Abbildung 8). Für die Verifikation und Validierung enthalten die Spalten der Matrix die wesentlichen Engineeringaufgaben wie Analysieren, Optimieren und Testen. Ergänzt werden diese ausgangsseitig durch Freigabeaktivitäten (Release) wie Homologation, Systemfreizeichnung und Komponentenfreigabe. Eingangsseitig sind erforderliche Testfälle durch synthetische Daten und Szenarien zu bestücken. Vertikal sind verschiedene Ebenen der Testobjekte zu unterscheiden, je nach Simulationsziel Komponenten, Subsysteme, Fahrzeugsysteme, Verkehrsknotenpunkte oder ganze Verkehrssysteme, die in ihren jeweiligen Testumgebungen (Operational Environment) analysiert, optimiert oder getestet werden. Die eingangs erläuterten Hauptarbeitsgebiete erfordern an jeder Stelle der Matrix Festlegungen der jeweils erforderlichen Architektur, der Simulationsziele und der Prozesse.

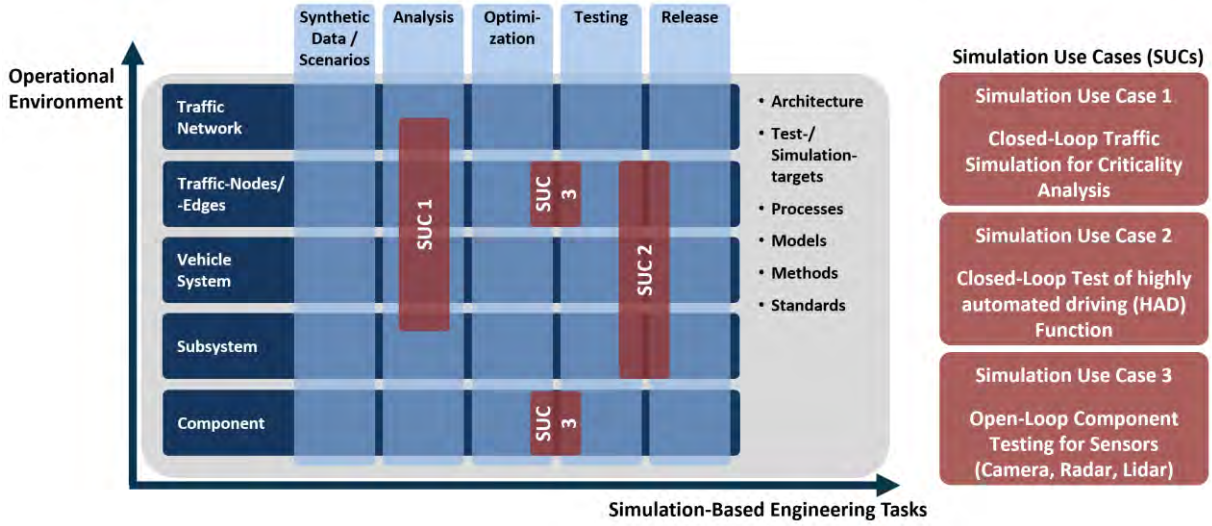


Abbildung 8: Matrix-Darstellung für Grundansatz des Projekts SET Level

Die konzeptuelle Sicht wurde im Projekt an wesentlichen Punkten exemplarisch durch sogenannte Simulation Use Cases (SUCs) auf unterschiedlichen Simulationstools bzw. -Toolketten demonstriert. Eine Closed-loop Verkehrssimulation diente als Demonstrator für eine Kritikalitätsanalyse (SUC 1). Zudem diente eine Closed-loop Systemsimulation für eine hochautomatisierte Fahrfunktion als Demonstrator für den Test eines integrierten Subsystems (SUC 2). Open-loop Simulationen für Kamera-, Lidar- und Radarsensoren dienten als Demonstrator für Komponententests (SUC 3).

Um ein vertieftes Verständnis für den im Projekt verfolgten Grundansatz zu erlangen, ist die Unterscheidung der Simulationsaufgabe (Simulation Task), des Testobjekts (System under Test) und der Testumgebung (Operational Environment) essenziell (siehe Abbildung 9).

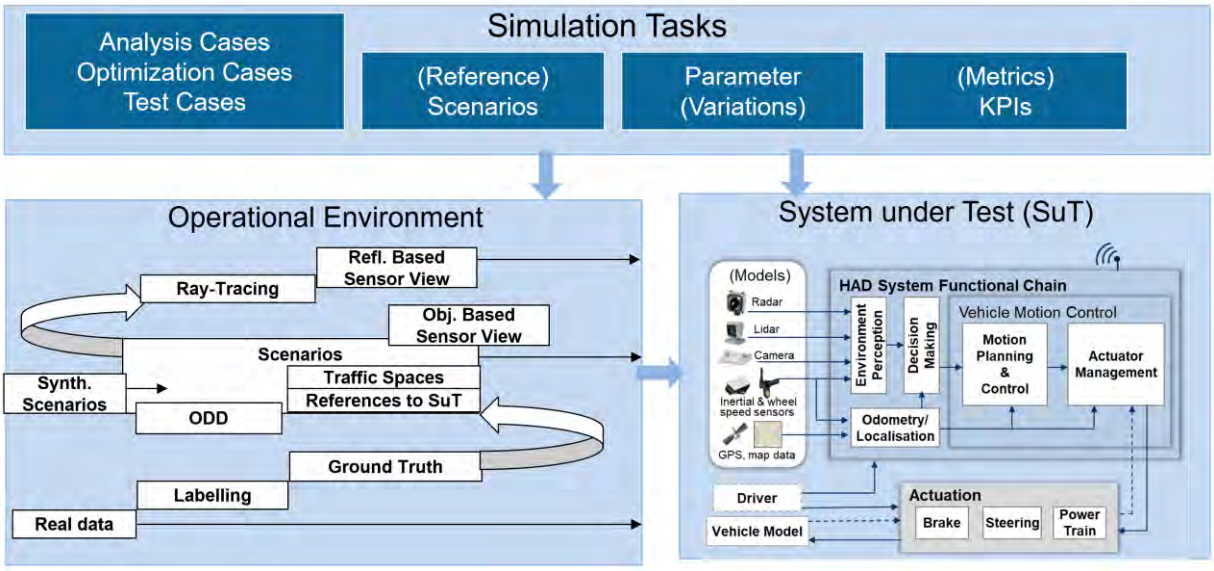


Abbildung 9: Erforderliche Arbeitsgebiete für den Aufbau von Simulationen

Wie eingangs bereits erwähnt, können für die wesentlichen Simulationsaufgaben (Simulation Tasks) folgende Fälle unterschieden werden: Analyse, Optimierung und Testen. Das Projekt SET Level hat sich auf szenarienbasiertes Testen fokussiert und für die Demonstratoren so

genannte Referenzszenarien erstellt. Referenzszenarien beinhalten Parameter auf unterschiedlichen Ebenen (Layern) und ermöglichen die Generierung großer Mengen an Absicherungskilometern. Sie beschreiben Straßengeometrien, feste und bewegliche Straßeninfrastrukturelemente, bewegliche Verkehrsteilnehmer, Umgebungsbedingungen sowie Kommunikationsbeziehungen zu anderen Verkehrsteilnehmern oder Infrastrukturelementen. Schlussendlich definieren Simulationsaufgaben Metriken oder Key Performance Indicators (KPIs) – analog zu klassischen Testaufgaben, die in Vorbedingungen (Pre-conditions), Aktionen (Actions) und erwartete Ergebnisse (Expected Results) untergliedert werden können.
Das Testobjekt (System under Test) ist ein funktionales Modell der automatisierten Fahrfunktion. Eingangsinformationen werden von Sensoren wie Radar-, Lidar- oder Kamerasensoren verarbeitet, ergänzt um von weiteren Sensoren wie GPS oder Raddrehzahlsensoren. Im Kern muss sich die automatisierte Fahrfunktion mit den Aufgaben der Sensordatenverarbeitung und -fusion (Environment Perception), der Trajektorienplanung und der Ansteuerung der Aktuatoren befassen. Aktuatoren wie Bremse (Brake), Lenkung (Steering) und Antrieb (Power Train) bewegen das Fahrzeug (Vehicle Model) und führen damit zu aktualisierten Informationen über das Fahrzeugmodell.

Die Testumgebung (Operational Environment) erzeugt die notwendige Eingangsinformation für das Testobjekt. Daher muss das betrachtete Fahrzeugmodell in der Testumgebung verortet sein. Die Testumgebung kann je nach Simulationsaufgabe und -ziel und abhängig vom Testobjekt völlig unterschiedlich aussehen. Das Testobjekt kann mit Realdaten beaufschlagt werden. Es wird dann von Reprozessierung gesprochen. Realdaten können attributiert werden (Labelling) oder es wird gar ein Modell der Testumgebung erzeugt (Ground Truth). Szenarien beziehen sich immer auf eine Testumgebung, in der Verkehrsräume (Traffic Spaces) zu beschreiben sind. Szenarien referenzieren Testobjekte (SuTs). In Szenarien sind immer auch die Umgebungsbedingungen mit codiert. Dies bedeutet, dass Umgebungsbedingungen, die Operational Design Domain (ODD), wie sie sich aus den Anforderungen an ein künftiges Produkt ergibt, in einem Szenario mit berücksichtigt sein muss. Szenarien können nicht nur auf der Basis von Realdaten, sondern auch synthetisch erzeugt werden. Weiterhin können aus Szenarien Szenen erzeugt werden, die als Eingangsgrößen für Sensoren dienen. Dies ist die Voraussetzung für die sog. reflexionsbasierten Sensormodelle.
Um verschiedene Testobjekte (Systems under Test SuTs) konfigurieren zu können, wurde für SET Level eine Bibliothek frei verfügbarer Modelle aufgebaut (Open Source Model Library), siehe Abbildung 10. Die Modelle sind frei zugänglich und können in den unterschiedlichen Demonstratoren von SET Level und über SET Level hinaus (z. B. in anderen Förderprojekten) verwendet werden.

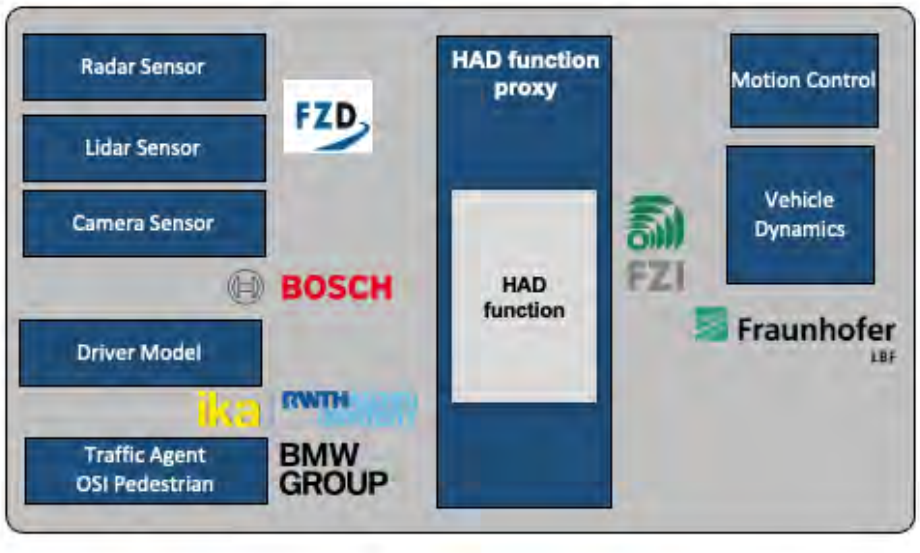


Abbildung 10: Verfügbare Open Source-Modelle im Projekt SET Level

Für die Open Source Model Library wurden unterschiedliche Sensormodelle vom FZD der TU Darmstadt und von Bosch entwickelt. Das im Projekt SET Level verwendete Fahrermodell (Driver Model) stammt vom ika der RWTH Aachen. Ein Fußgängermodell wurde von BMW entwickelt. Ebenso wurde eine Schnittstelle zu einer proprietären HAD-Funktion des FZI Karlsruhe entwickelt. Modelle für Trajektorienplanung und Fahrdynamik hat das Fraunhofer Institut LBF aus Darmstadt entwickelt.

Basierend auf diesen Modellen konnten komplexere Testobjekte für die bereits erläuterten Simulation Use Cases zusammengestellt werden. So existieren nahezu alle relevanten Teilmodelle (unter Berücksichtigung eines objektbasierten Kameramodells) für den SUC 1, ebenso die meisten Teilmodelle (ebenso objektbasierte Sensormodelle) für SUC 2 und vor allem die reflexionsbasierten Sensormodelle für SUC 3.

Die entwickelten Modelle sind für die künftige Anwendung etwa in weiteren Förderprojekten oder zur Anpassung an neuere Versionen der Standards beim ASCS zugänglich gemacht worden, siehe https://www.asc-s.de/.

Die **Testumgebung (Operational Environment)** stellt verschiedene Möglichkeiten bereit, das System under Test mit jeweils relevanten Eingangsinformationen aus der Umgebung zu versorgen, die beispielsweise basierend auf der simulierten Testumgebung erzeugt werden. Wie oben bereits ausgeführt, kann es sich dabei um Realdaten, reale oder synthetische Umgebungsmodelle oder auch synthetisch erzeugte Realdaten handeln. Dafür wurden die beim ASAM angelegten und teilweise bereits aus dem Projekt PEGASUS bekannten IT-Standards genutzt, erprobt und wo notwendig ergänzt (siehe Abbildung 11).

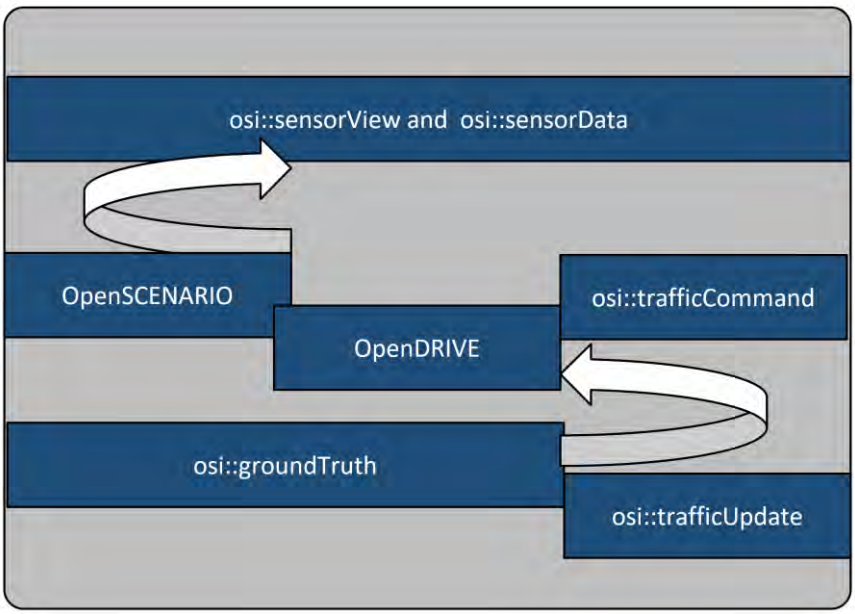


Abbildung 11: ASAM-Standards zur Abbildung der Testumgebung

Basierend auf einem objektbasierten Modell der Testumgebung (Ground Truth) wurde mit der Nachricht OSI::GroundTruth im Rahmen des ASAM Open Simulation Interface (ASAM OSI) Standards die Möglichkeit geschaffen, diese abstrahierte Darstellung der Testumgebung an relevante Komponenten des Simulationstools zu übermitteln. Mit OpenSCENARIO und OpenDRIVE wurden ASAM Standards um Elemente zur Abbildung urbaner Testumgebung erweitert. OpenSCENARIO beschreibt die dynamischen Aspekte eines Szenarios (z. B. die Interaktion zwischen Verkehrsteilnehmern). OpenDRIVE hingegen beschreibt die statischen Anteile eines Szenarios (z. B. das Straßennetzwerk und weitere Infrastruktur). Basierend auf der Ausführung der Szenarien können mittels OSI::TrafficCommand Ausführungsbefehle an das System under Test oder an den Umgebungsverkehr (z. B. andere Fahrzeuge oder Fußgänger) übermittelt werden. Zudem können zu jedem Simulationszeitschritt Aktualisierungen der bewegten Verkehrsteilnehmer durch OSI::TrafficUpdate an die Testumgebung zurück übermittelt werden. Diese verändern bzw. aktualisieren damit die Testumgebung (Ground Truth).

Für die Sensorsimulation können, basierend auf der Ausführung von Szenarien, zu jedem Simulationszeitschritt mittels OSI::SensorView sensorspezifische Sichten generiert und als Eingangsinformation an die Sensormodelle übertragen werden. Als Ausgangsinformation wird von den Sensormodellen die Generierung von OSI::SensorData erwartet. Hierdurch ist eine ISO 23250 konforme und standardisierte Eingangsschnittstelle für das Testen von automatisierten Fahrfunktionen entstanden.

Die verwendeten und weiterentwickelten IT-Standards sind beim ASAM verfügbar (siehe https://asam.net).

Um das Thema Simulationsaufgaben (Simulation Tasks) nochmals zu vertiefen (siehe der obere Teil von Abbildung 9) ist es essenziell, zu verstehen, dass die Simulationsaufgabe Teil des Produktentwicklungsprozesses (Product Development Process) ist (siehe Abbildung 12).

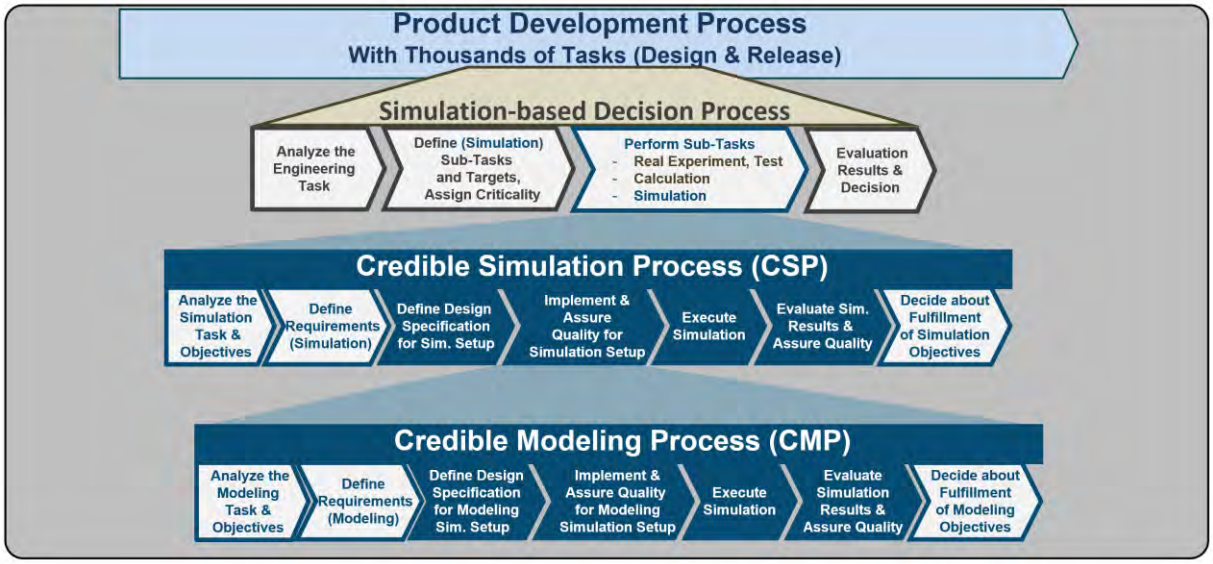


Abbildung 12: Simulationsaufgabe als Teil des Produktentwicklungsprozesses[4]

Anteile des Produktentwicklungsprozesses können dabei vom simulationsbasierten Entscheidungsprozess (Simulation-based Decision Process) übernommen werden. Versuchsanordnungen (Real Experiment), Berechnungsverfahren oder Simulationen im eigentlichen Sinne können dabei durch den Credible Simulation Process (CSP) unterstützt werden. Der CSP geht von einer Analyse der Simulationsaufgabe und einer darauf basierenden Festlegung der Simulationsziele aus. Auf Basis der Simulationsziele werden Simulationsanforderungen festgelegt. Die Definition des Aufbaus der Simulationstools (Simulation Setup) durch die Spezifikation der Simulationsarchitektur ist essenziell und entscheidet über die Implementierung der Simulationstools und aller zugehöriger Module (z. B. des Modells des Systems under Test) im Sinne einer Werkzeugkette. Die Ausführung der Simulation (Execution), die Evaluierung der Simulationsergebnisse und die Lieferung der Simulationsergebnisse an die Entscheidungsebene schließen den CSP ab. Der Credible Modelling Process (CMP) schließlich dient dazu, den für den CSP benötigten Umfang von Modellen (welche meist aus verschiede-nen Quellen bzw. von verschiedenen Lieferanten stammen) systematisch und qualitätsgeprüft bereitzustellen.

Die im Projekt SET Level entstandenen Prozessbeschreibungen wurden als Ideen vom Projekt SmartSE des prostep ivip Vereins (siehe https://prostep.org) übernommen und ausgearbeitet. Für die Wartung und Weiterentwicklung werden sie zurückübertragen an das SmartSE Projekt.

Das Projekt SET Level profitierte entscheidend von der Spezifikation so genannter Simulation Use Cases (SUCs). Nach anfänglich intensiven Diskussionen mit dem Schwesterprojekt VVMethoden (wo ebenfalls an Use Cases gearbeitet wurde und wird) wurde ersichtlich, dass eine Präzisierung der Use Cases unabdingbar ist. Nachdem das Verständnis von Use Cases im Projekt VVMethoden reifte und beispielweise eine Unterscheidung von Use Cases anhand von unterschiedlichen Kreuzungstypen (z. B. T-Kreuzung) implementiert wurde, wurde

[4] siehe https://gitlab.setlevel.de/open/processes_and_traceability/credible_simulation_process_framework/-/raw/main/Credible-Simulation-Process-v1-3.pdf

empfohlen, im Projekt VVMethoden von Functional Use Cases zu sprechen. Use Cases im Projekt SET Level wurden fortan als Simulation Use Cases bezeichnet.

Im Projekt SET Level wird unter einem Simulation Use Case das Zusammenwirken aller Module verstanden, die für die Ausführung einer Simulationsaufgabe erforderlich sind. Ausgehend von einer Simulationsaufgabe mit einem zugehörigen Simulationsziel handelt es sich bei besagten Modulen im Wesentlichen um das Modell eines Testobjektes (System under Test), das Modell der Testumgebung (Operational Environment) und die Simulationsplattform selbst (bestehend aus dem Simulationskern und weiteren Modulen). Auf die für einen Simulation Use Case erforderliche Architektur wird später (siehe Abschnitt 2.1.3.3) in diesem Dokument vertieft eingegangen. Entscheidend für das Projekt SET Level war die Prämisse, aus allen Teilprojekten Beiträge für die exemplarische Umsetzung der Simulation Use Cases einzufordern und somit eine verbindende Klammer über alle Teilergebnisse des Projekts hinweg zu schaffen.
Die Verortung der im Projekt SET Level betrachteten Simulation Use Cases im V-Modell der Produktentwicklung geht aus Abbildung 13 hervor.

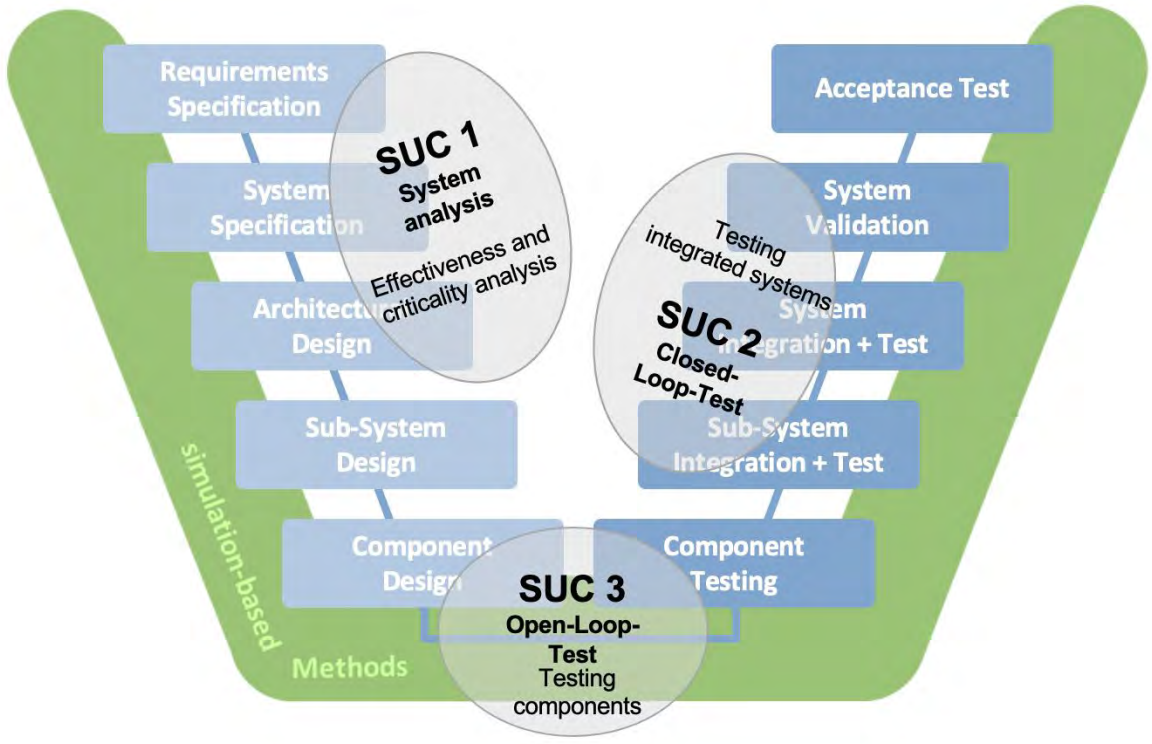


Abbildung 13: Verortung der Simulation Use Cases im V-Modell der Produktentwicklung

Zu jedem Simulation Use Case wurden ein oder auch mehrere Demonstratoren aufgebaut. Simulation Use Case 1 (SUC 1) widmete sich einer Analyseaufgabe, der Kritikalitätsanalyse einer Verkehrssimulation. Simulation Use Case 2 (SUC 2) widmete sich mit mehreren Demonstratoren der Closed-loop Simulation von Subsystemen. Simulation Use Case 3 (SUC 3) widmete sich mit mehreren Demonstratoren der Open-loop Simulation von Komponenten, insbesondere von Kamera-, Lidar- und Radarsensormodellen. Dadurch wurde insbesondere die Nutzbarkeit von Standards im Kontext des simulationsbasierten Entwickelns und Testens im Projekt SET Level mehrfach demonstriert. An den Stellen, an denen sich Defizite in den Standards zeigten, wurden während der Projektlaufzeit entsprechende Rückmeldungen durch Mitarbeiter des Projekts SET Level bei den entsprechenden Standardisierungsgremien eingebracht und bearbeitet.

In diesem Zuge wurde Klarheit über die Verortung von Standards geschaffen. Als Beispiel mag die Operational Design Domain (ODD) dienen, die nach Überzeugung der Mitarbeiter im Projekt SET Level in das Gebiet der Requirements Specification gehört. Entsprechende Ableitungen sollten sich daher in den IT-Standards für die Simulation (z. B. OpenSCENARIO) wiederfinden. Im Projekt SET Level wurden auch Erfahrungen im Umgang mit dem Standard Functional Mockup Interface (FMI) gesammelt. So wurden Interaktionseffekte beobachtet, die erst auftreten, wenn integrierte Subsysteme Effekte zeigen, die in isolierten Anwendungsfällen nicht auftreten. Die Notwendigkeit für den standardisierten Zusammenbau von Functional Mockup Units (FMUs) wurde im Projektverlauf deutlich und wird in Folgeversionen des Standards System Structure and Parameterization (SSP) einfließen. Notwendige Erweiterungen von OpenDRIVE für den Einsatz in urbanen Verkehrsräumen wurden identifiziert und in die Standardentwicklung eingebracht. Außerdem wurden im Projekt SET Level Handlungsbedarfe in der Spezifikation und Erweiterung von Schnittstellen zwischen den Modulen von Simulationstools und von Simulationsmodellen identifiziert und durch Erweiterungen des ASAM Open Simulation Interface (OSI) aufgelöst.

Abschließend ist die im Projekt SET Level identifizierte fundamentale Bedeutung einer vereinheitlichten **Architektur** für Test- und Analyseumgebungen zu erläutern. In der Praxis sind, abhängig von der Simulationsaufgabe, durchaus sehr unterschiedliche Ansätze bekannt und in Anwendung. Im Projekt SET Level wurde auf Basis bestehender Diskussionen in den ASAM Standardisierungsgremien zur Beschreibung der Architektur von Test- und Analyseumgebungen ein hierarchischer Ansatz mit drei Verfeinerungsstufen bzw. Betrachtungsebenen entwickelt (siehe Abbildung 14).

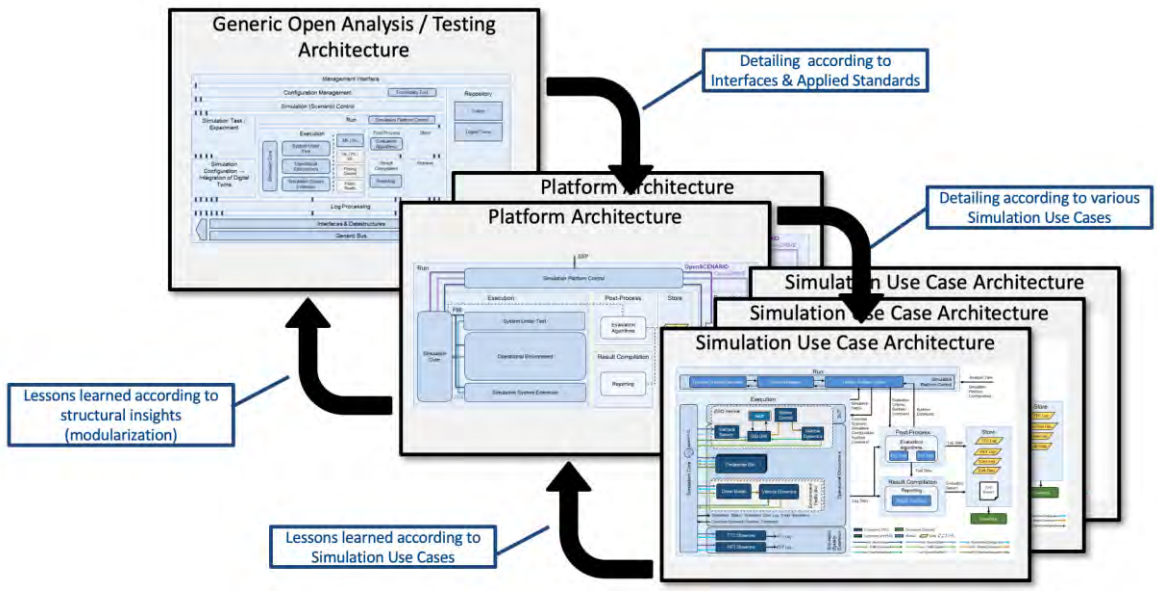


Abbildung 14: Verfeinerungsstufen der Architektur

Gemeinsamkeiten der verschiedenen Verfeinerungsstufen wurden (basierend auf Vorarbeiten im ASAM) auf oberster Betrachtungsebene als **Generic Open Analysis / Testing Architecture** dokumentiert. Der Ansatz sollte sämtliche Verifikations- und Validierungsaufgaben automatisierter Fahrfunktionen einschließen. Dazu zählen neben dem simulationsbasierten Testen (MiL, SiL und HiL) auch das Testen auf Prüfgeländen oder auf öffentlichen Straßen. Durch die Berücksichtigung sämtlicher genannter Testinstanzen (über den Fokus des

Projekts SET Level hinaus) wird der Architekturansatz dem eingangs als „Multi Pillar Approach“ bezeichneten Anspruch gerecht.

Test- und Analyseaufgaben werden von Modulen ausgeführt, die auf einer Ebene mittlerer Granularität in so genannten **Platform Architectures** beschrieben sind. Diese Betrachtungsebene umfasst neben den Modulen auch Hinweise auf den Einsatz von Standards, z. B. an Schnittstellen.

Die bereits erwähnten **Simulation Use Cases** machten deutlich, dass die konkrete Umsetzung der Architektur abhängig von der Simulationsaufgabe, von vorhandenen Tools und Modellen ist und unterschiedliche Ausprägungen annehmen kann. Daher finden sich auf der untersten Ebene der Architekturbeschreibung spezifische Ausprägungen der auf höheren Ebenen beschriebenen Architekturen. Diese Freiheit eröffnet die Möglichkeit anwendungs- und nutzerspezifischer Implementierungen in einem vorgegebenen Rahmen und unter Gewährleistung einheitlicher Simulationsqualität und damit Credibility von Simulationsergebnissen.

1.5 Zusammenarbeit mit anderen Stellen

Dem bereits zu Projektbeginn formulierten Anspruches des Projektes SET Level auf Anbindbarkeit der im Projekt entwickelten Simulationswerkzeuge- und Methoden an die Methoden, Werkzeuge und Daten weiterer leitender Initiativen im Gebiet simulationsbasiertes Testen wurde dadurch Rechnung getragen, dass gezielt informelle Austauschaktivitäten und gemeinsame Konferenzbeiträge mit informellen Partnerschaften angestrebt und im Projektzeitraum realisiert wurden. Diese Partnerschaften sind grundsätzlich so angelegt, dass sie auch über die Projektlaufzeit von SET Level hinaus Bestand haben werden, dies unter dem Label der PEGASUS Familie und auch unter den laufenden Aktivitäten des Projektes VVMethoden.
Dabei wurden für das SET Level Projekt sowie für VVMethoden Systematiken zum frühzeitigen Austausch technischer Vorergebnisse eingeführt, um eine frühzeitige Nutzung und Einholung von Feedback externer Experten auf Vorergebnisse zu erreichen und um gleichzeitig die externen Aktivitäten wie z. B. in der Standardisierung zumindest partiell auf die SET Level Simulationsmethode auszurichten.
Parallel dazu wurden in enger Abstimmung mit dem Projekt VVMethoden auf der internationalen fachlich-strategischen Ebene Allianzen mit Stakeholdern organisiert, um damit den beiden Projekten SET Level und VVMethoden einen kontinuierlichen Raum und Aufmerksamkeit für die dort entwickelten Lösungsarchitekturen zu geben. Das strategische Ziel dieser Allianzen besteht vorrangig darin, Konvergenzen bestehender Simulations- und Testmethoden zu identifizieren und gemeinsame Justierungen zu stimulieren, so dass nach der Projektlaufzeit beider Projekte eine möglichst breite Überführbarkeit der Methoden, genutzter Simulationswerkzeuge aber auch der damit verbundenen Datenräume existiert und damit künftiger Entwicklungsaufwand für die in der Simulationsmethode genutzten Werkzeuge reduziert wird. Somit stehen perspektivisch die Ressourcen für Optimierung der Testverteilung und Design der Parameterräume zur Verfügung.
Unter dieser strategischen Agenda zum frühzeitigen Austausch wurden die folgenden Aktivitäten gestartet und mit den nachfolgend beschriebenen Resultaten durchgeführt.

1.5.1 Nationaler, fachlicher Austausch

Zu dem wichtigsten nationalen Projekt im Gebiet Absicherungsmethoden durch virtuelle Nachweisverfahren, dem vom BMWK geförderten Projekt VVMethoden, wurden bereits während der Antragsphasen beider Projekte auf mehreren Ebenen Querverbindungen geschaffen

(siehe Abbildung 15). Auf organisatorischer Ebene wurden mehrere Arbeitsgruppen eingerichtet und gepflegt. Weiterhin wurden auf beiden Seiten Unterarbeitspakete mit dem vorrangigen Ziel der gegenseitigen Kompatibilität der im Projekt VVMethoden entwickelten Gesamtmethode mit den im Projekt SET Level entwickelten Simulationsmethoden gebildet. Im Zuge der entsprechenden Entwicklungen wurden im Projekt SET Level den drei Simulation Use Cases drei entsprechende Functional Use Cases auf VVM-Seite gegenübergestellt, die auf ihren jeweiligen Ebenen die technologische Verzahnung beider Projekte darstellen (siehe Abbildung 15). Diese „Verzahnungsarchitektur“ SET Level / VVM) wurde auch stets auf den Konferenzen als leitend dargestellt, um so der internationalen Community das Kohärenzbestreben der BMWK Projekte zu erläutern.
Auf fachlicher Ebene wurden zunächst diverse Vorergebnisse der Projekte SET Level und VVMethoden ausgetauscht, z. B. im Rahmen der jeweiligen Arbeitspakete 5 respektive Teilprojekt 5 aber auch in eigens eingerichteten Austauschformaten wie z. B. die gemeinsame Arbeitsgruppe „Komponentenmodelle“ für Sensor- und Funktionsmodelle“ aber auch in der Arbeitsgruppe „Glossar“ zur Begriffsdefinition. Um den Austausch zugangsvereinfacht zu realisieren, wurde für beide Projekte ein vergleichbarer Freigabeprozess für Vorergebnisse entwickelt und angewendet, der einen beschleunigten Austausch von fachlichen Spezifikationen ermöglicht hatte. Dieser Freigabeprozess wurde beispielsweise auf die folgenden Austauschgegenstände angewendet: Sensor- und Komponentenmodelle, Begriffsdefinitionen, Spezifikationen von Simulation Use Case und Functional Use Case, Ontologien und Berichte zum methodischen Vorgehen.
Im weiteren Projektverlauf wurden mit den ersten offiziellen Ergebnissen sowie mit eigens zweckgebunden freigegebenen Vorergebnissen ein Austausch zu weiteren nationalen Projekten gepflegt, z. B. mit dem vom BMWK geförderten Projekt KI-Absicherung zur Ontologie oder mit dem vom BMBF geförderten Projekt VIVALDI zum Glossar. Weiterhin sind informelle Pfade in mehrere Standardisierungsorganisationen bereits vor Projektstart vorhanden gewesen, so z. B. in mehrere ASAM Gruppen sowie in die ISO TC 22 SC33 WG 9 „Scenario-based safety evaluation framework for ADS“. Diese Pfade wurden zur Dissemination für die Zwischenstände genutzt bzw. in den nationalen Vorabstimmungen zur ISO diskutiert. Erweiternd wurde speziell vom Projekt VVMethoden ausgehend eine Dauerpräsenz in der neuen ISO TC22 SC32 WG13 „Safety for automated driving systems — Design, verification and validation“ erwirkt und damit auch ein Disseminationspfad über die Laufzeit des Projekts SET Level hinaus sichergestellt.

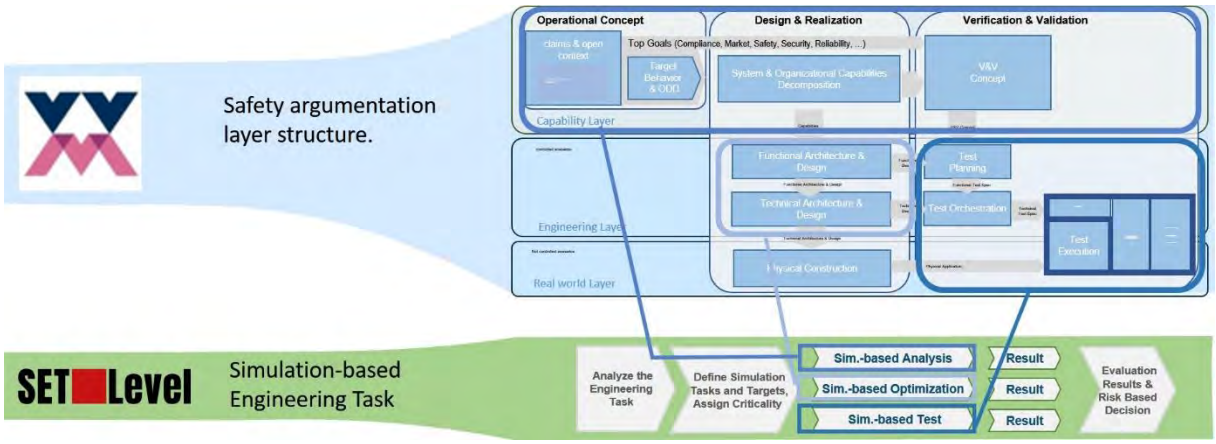


Abbildung 15: Verzahnung der im Projekt VVMethoden entwickelten Sicherheitsargumentation mit dem Arbeitsfluss der im Projekt SET Level entwickelten Simulationsmethode (Galbas, et al., 2022)

Auf nationaler Ebene wurden Treffen zwischen den Koordinatoren der Projekte der VDA Leitinitiative durchgeführt, um die Kohärenz zwischen den Projekten sicherzustellen. Aus diesen Treffen wurden Vorgehensweisen speziell zu internationalen Auftritten abgeleitet sowie die besagte Verzahnung zwischen den Projekten verfolgt. Speziell mit dem Projekt VVMethoden wurden auch Werkzeuge aus dem Projekt SET Level und deren Nutzungen diskutiert und z. B. im Falle des im Projekt SET Level (weiter)entwickelten Credible Simulation Process Frameworks eine Weiternutzung im Projekt VVMethoden sowie eine Verstetigung im Rahmen des prostep ivip Projektes SmartSE erwirkt.
Neben den Aktivitäten in den Standardisierungsgremien ist auch die im Projekt SET Level entwickelte „Forschungsimplementierung“ eine auf Verstetigung und Wachstum angelegte Realisierung der im Projekt SET Level entwickelten Simulationsmethode mit der auch zukünftig eine Verbreitung und ein Einbringen in internationale Partnerschaften angestrebt werden.
Zusätzlich zu den beschriebenen nationalen Aktivitäten wurden die (international ausgerichteten) Halbzeitevents der Projekte SET Level und VVMethoden gezielt genutzt, um die operative Anbindung an internationale Aktivitäten weiter auszubauen. Auf diesen Halbzeitevents wurde neben den Projektergebnissen auch die Kohärenz innerhalb der Projektfamilie dargestellt, indem die Verzahnung des jeweiligen Schwesterprojektes erläutert wurde.
Speziell beim zweitägigen Halbzeitevent des Projekts VVMethoden wurde der gesamte zweite Tag für eine internationale Veranstaltung genutzt, auf der neben Rednern aus den Projekten SET Level und VVMethoden auch namhafte internationale Redner in die Gestaltung eingebunden waren und die in der abschließenden Diskussion die Sicherheitsargumentationen und die entsprechenden Simulationsmethoden diskutiert haben. Neben der Präsentation des SET Level Zwischenergebnisses „A Configurable Simulation Tools for Various V&V Tasks” wurde mit dem weiteren SET Level Vortrag „International perspectives and link to VVM (When) are we ready for deployment?“ der hohe Anwendungsbezug vom Projekt SET Level zum simulationsbasierten Sicherheitsnachweis verdeutlicht.

Internationaler, fachlich-strategischer Austausch

Als eine wichtige Ausgangsposition für den Ausbau der internationalen Abstimmungen unter dem Dach der „PEGASUS Familie“ wurden die bereits bestehenden Netzwerke aus der PEGASUS Familie genutzt. So wurde z. B. der PEGASUS Expert Workshop (PEW) als bewährtes Format einer international gut besetzten Expertengruppe zum Austausch genutzt. Während der Laufzeit des Projekts SET Level fanden unter Beteiligung von Mitarbeitenden im Projekt SET Level im Rahmen dieser PEWs zwei virtuelle Symposien mit den folgenden Titeln statt: „Building a Robust Safety Case“ und „Formalizing the Safety Case“. Dadurch wurde international eine Baseline zu den noch offenen Aufgabenbereichen in der Nachweiskette erarbeitet. Bei weitgehender Übereinstimmung über die noch offenen Anforderungen für diese Nachweiskette konnten u. a. durch diese Symposien die Simulationsarchitekturen und fachlichen Strategie der beiden in Deutschland führenden Projekte in dem Bereich Absicherung automatisierter Fahrfunktionen international erfolgreich dargestellt und in die Diskussion eingebracht werden. Ergänzt wurde diese internationale Anbindung durch die bereits genannten Standardisierungsaktivitäten, vorrangig dabei die Arbeiten in ASAM OpenX und in der ISO TC 22 SC33 WG 9.
Aus diesen Abstimmungen wurde ersichtlich, dass eine Virtualisierung und Formalisierung aller am Sicherheitsnachweis beteiligten Elemente, wie z. B. der ODD, der Kritikalität in abstrakten Szenarien, der Umsetzung in konkret beschriebene und simulationswerkzeugkompatible Szenarien die idealerweise in Szenariendatenbanken zur Verfügung gestellt werden können, die Qualifizierung von Testdaten und Simulationsmethoden sowie die Validierung der Simulation gegenüber der Realität, auch künftig als weisende Elemente für Projekte, um den

simulationsbasierten Sicherheitsnachweis ihre Gültigkeit haben werden. Dazu zählen auch künftige Justierungen und Anwendungen der im Projekt SET Level entwickelten Simulations(werkzeug)architektur. Zudem besteht in der internationalen Community weiterhin Interesse, anwendungsnahe Ergebnisse des Projekts SET Level zu prüfen. Zu diesen Ergebnissen gehören beispielsweise die Referenzszenarien und Komponentenmodelle zur Anbindung an externe Tools.

Aufgrund dieser international relativ einheitlichen Sichtweise wurden mit einigen der Hauptakteuren aus dem PEW bilaterale Treffen zum fachlich strategischen Austausch organisiert, um potenzielle gegenseitige Nutzungsmöglichkeiten von den jeweiligen Vorgehensweisen, Szenarien oder Daten zu besprechen und auch um bewährte Vorgehensweisen gegenseitig zu reflektieren. Im Zentrum des Austauschs standen (und stehen auch nach dem Ende des Projekts SET Level) vor allem Kooperationen mit Partnern aus USA, Japan, Frankreich, UK sowie der EU. Im Einzelnen verteilen sich diese Austauschaktivitäten, zu denen es jeweils mehrere Abstimmungstreffen gegeben hat, wie im Folgenden beschrieben.

Mit der Warwick Universität (WMG) stand insbesondere die Toolkette zum simulationsbasierten Sicherheitsnachweis im Vordergrund. Beide Seiten arbeiten an vergleichbaren Spezifikationen und nutzen ähnliche Werkzeuge und während zwar WMG Warwick für ihre kollaborative Szenariendatenbank als zentrale Drehscheibe für V&V Stakeholder mit seiner Szenarien Description Language „WRAP“ ein eigenes Format entwickelt hat sollen von der WMG Datenbank (WMG DB) nicht zuletzt auch wegen der Kooperationsgespräche mit SET Level und VVMethoden ein Kompatibilität der WMG DB auch zu den dort verwendeten Werkzeugen und Spezifikationen wie z. B. aus ASAM OpenX bzw. CARLA und openPASS hergestellt werden. Zudem stehen mögliche Partizipationsmodelle mit der WMG DB im Raum, wobei insbesondere formal in OpenSCENARIO beschriebene Szenarien sowie dazu kompatible Testdaten im Vordergrund stehen können. Aufgrund der rechtlichen Situation konnte sich das Projekt SET Level nicht formal an der WMG DB beteiligen. Allerdings haben einige Projektpartner diese Beteiligung eigeninitiativ aufgenommen. Weiterhin wird von der über das TPX-Projekt noch laufenden Kooperation mit WMG Warwick ein guter Zugang zur Formaldefinition der ODD für den Sicherheitsnachweis erwartet, weil WMG Warwick dort in der ISO und auch in ASAM eine zentrale Position einnimmt. Insbesondere die im Jahre 2023 erwartete ISO Spezifikation TS 5083 aus der ISO TC 22 SC32 WG 13 wird wertvoll sein.

Eine vergleichbare Initiative zu einer kollaborativen Austauschplattform wird mit ADScene (Kernpartner: Renault, System-X, PSA Group, Vedecom) aus Frankreich ins Leben gerufen. Dort soll nach noch zu definierenden Partizipationsmodellen ein Austausch formaler Szenarien stimuliert werden. Bei den Kooperationstreffen mit ADScene wurden ähnliche Vorgehensweisen zum simulationsbasierten Sicherheitsnachweis erkannt und Durchgängigkeit des Sicherheitsnachweises diskutiert. Durch die erklärte Bereitschaft von Mitarbeitenden im Projekt ADScene u. a. auf dem im Rahmen der PEGASUS Familie entwickelten grundsätzlichen Verfahrens aufzusetzen ist eine grundsätzliche Durchgängigkeit des Sicherheitsnachweises bereits vorhanden. Zudem ist eine Weiterverwendung von im Projekt SET Level erarbeiteten Ergebnissen durch ADScene gewährleistet. Das Projekt ADScene erweitert das bisherige PEGASUS Netzwerk um weitere Partner vor allem aus dem europäischen Raum und öffnet damit auch Perspektiven zu künftigen EU Projekten in dem thematischen Umfeld wie z. B. dem EU Projekt SUNRISE mit seinen Akteuren.

Als weiteres verbindendes Element sowohl für die außereuropäischen Beziehungen als auch für Verbindungen zu europäischen Projekten zum simulationsbasierten Testen wurden neben der bereits bestehenden ISO Working Group TC 22 SC 33 WG9, in der im Zeitraum 2019–2022 in zweiwöchentlichen Webkonferenzen gemeinsam mit Mitarbeitenden im Projekt SET Level an der ISO Standardisierung zum „Scenario-based safety evaluation framework for

ADS“ gearbeitet wurde, weitere Kanäle zu den japanischen Partner eröffnet: Aus den Kontakten durch die stetigen Beiträge durch die PEGASUS Familie auf der jährlichen SIP-adus Konferenz sowie durch Beiträge in dem vom BMBF geleiteten Steuerkreis zur deutsch-japanischen Kooperation wurde im Jahr 2022 mit dem dreitägigen Symposium safeCAD-DJ in Berlin ein eigenes Austauschforum errichtet. Auf diesem Symposium wurden unter Beteiligung von Mitarbeitenden der Projekte SET Level, VVMethoden und VIVALDI sowie mit den Koordinatoren der japanischen Projekte SAKURA und DIVP Keynotes und Podiumsdiskussionen abgehalten.
Zudem wurde am dritten Tag des safeCAD-DJ Symposiums an einer Fachstrategie mit Vertretern der deutschen und japanischen Ministerien (BMWK und BMBF sowie Cabinet Office und METI) gearbeitet. Dabei wurde neben dem Wunsch nach gegenseitiger Nutzung von Sensormodellen, Implementierungen von Standards in Simulationswerkzeugen, Strukturen und Szenarien auch der Bedarf an zukünftigen Optimierungen und Durchgängigkeiten der Werkzeugketten beschrieben, insbesondere im Hinblick auf das Teilen von Modellen und Verständigen auf deren Durchgängigkeiten im Entwicklungs- und Testframework, auf das Generieren von qualifizierten Daten und auf die Überführung manuell bedienter (Test-)Werkzeuge auf teilautomatische Verfahren. Für diese künftigen Schritte wurden die bisherigen Definitionen in der PEGASUS Familie u. a. als gemeinsame Basis identifiziert und speziell die Beträge des Projekts SET Level intensiv aufgenommen. In abschließenden Erklärungen bekräftigten die beteiligten Ministerien aus Deutschland und Japan die auf safeCAD-DJ vorgetragenen Erkenntnisse auch für künftige Förderkulissen berücksichtigen zu wollen und damit den internationalen Austausch weiter zu fördern.
Auf der SIP-adus Konferenz 2022 in Kyoto wurde neben einem Plenumsvortrag über VVMethoden und SET Level auch die abgehaltene Break-Out Session zum Thema „Safety Assurance“ mitmoderiert. Aus den Teilnehmern dieser Session wurde eine neue Allianz zwischen der PEGASUS Familie, dem SAKURA Projekt sowie SUNRISE als europäisches Projekt zum szenarienbasierten Testen vereinbart. Eine mögliche Allianz mit dem US SAE Automated Vehicle Safety Case Consortium wurde ebenfalls diskutiert. Dadurch besteht eine weitere operative Anschlußmöglichkeit an die USA.
Durch diese neuen Aktivitätsstränge in Kombination mit den bisherigen Austausch- und Konferenzformaten (Abbildung 16) sind ideale Rahmenbedingungen geschaffen, um im Jahre 2023 unter dem Label PEGASUS Familie sowohl die Endergebnisse des Projekts SET Level als auch die des laufenden Projektes VVMethoden und die Inhalte der weiteren genannten Initiativen und Projekte in einem gemeinsamen internationalen Netzwerk zu diskutieren. Davon ausgehend können neue Projeke und Aktivitätsstränge im Themengebeit szenarien- und simulationsbasiertes Testen gestartet werden.

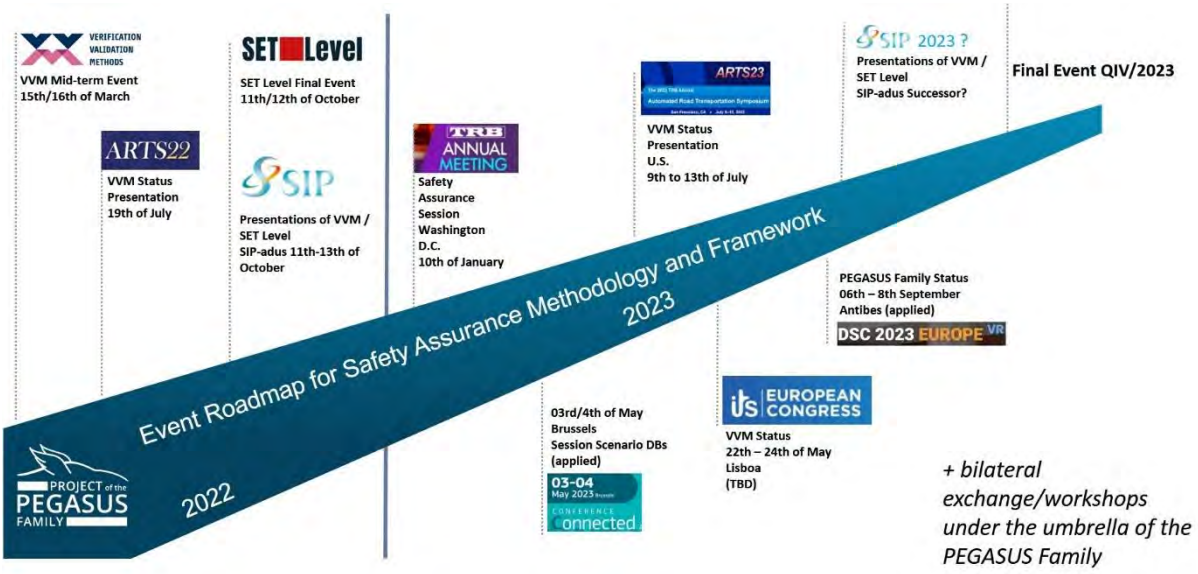


Abbildung 16: PEGASUS Familie - Dissemination Roadmap 2022 und 2023

2 Eingehende Darstellung der fachlichen Teilprojekte

2.1 Teilprojekt 1: Use-Case Management und funktionsorientierte Anforderungsanalyse

2.1.1 Übersicht über Inhalte und Ziele von TP1

TP 1 definierte mit den Anforderungen einen Startpunkt für die Arbeiten der Teilprojekte 2 bis 5. Zudem trug TP 1 übergreifend mit der Beschreibung von Anwendungen der Simulation, mit Konzepten, einem Glossar und beispielhaften Verkehrsszenarien zur Identifikation gemeinsamer Themen und einem kohärenten Vorgehen bei. Darüber hinaus lief auch ein großer Teil der Zusammenarbeit mit anderen Projekten, insbesondere dem Schwesterprojekt VV-Methoden, über TP 1 zusammen mit TP 5. In dieser Rolle unterlegten diese Teilprojekte die organisatorische Projektkoordination des TP 0 auf inhaltlicher Ebene.
Eine prominente Rolle unter den Ergebnissen des TP 1 spielte die Formulierung von Anforderungen an Simulationslösungen, die sich aus der Rolle der Simulation bei Verifikation und Validierung automatisierter Fahrzeuge hoher Automatisierungsstufen im urbanen Raum ergeben. Diese betreffen zum einen Funktionsumfang, Qualität und Validität der Simulationsergebnisse sowie zum anderen nichtfunktionale Anforderungen an ihre Realisierung wie Offenheit und Modularität.
Ergänzt wurden die Anforderungen durch eine Beschreibung von möglichen Anwendungen der Simulation im Entwicklungsprozess. Die ausgearbeiteten Anwendungen spannen in exemplarischer Form ein breites Spektrum auf, das die Simulationsumgebungen, die sich an den Prinzipien des Projektes orientieren, aufgrund ihrer Flexibilität abdecken können.
Für die Verwendung im Projekt SET Level selbst und im Austausch mit anderen öffentlich geförderten Projekten wurden Begriffe in einem Glossar zweisprachig in Deutsch und Englisch zusammengestellt.
Zur Realisierung von Anwendungen der Simulation werden neben den Kernkomponenten einer Simulation und Komponenten, die ein zu testendes Automationssystem in einem Fahrzeug oder einer Verkehrsumgebung modellieren, auch oft spezifische Verfahren benötigt, welche zur Beantwortung der jeweiligen Fragestellung höhere Simulationsfunktionalitäten realisieren. Insbesondere Explorationsverfahren für große Szenarienräume sind Beispiele für höhere Simulationsfunktionalitäten. Ein solches Verfahren, das darauf abzielt, eine Garantie für das Entdecken aller kritischer Szenarieninstanzen geben zu können, wurde in Detaillierung einer Simulationsanwendung spezifiziert.
Weitere Ergebnisse des TP 1 betreffen die Definition von Simulationsaufgaben. Neben vorbereitenden Aktivitäten mit Anforderungscharakter für eine Beschreibungssprache von Simulationsaufgaben, die im TP 2 aufgenommen wurden, erstellte das TP 1 eine Liste beispielhafter Szenarien. Dafür wurden Karten von Verkehrsräumen (Beispiel: vierarmige Kreuzung) und darauf ablaufende Verkehrsdynamiken (Beispiel: Rechtsabbiegen bei kreuzendem Fußgänger) erstellt. Diese Szenarien haben den Charakter von Testabläufen, wo ein automatisiertes Fahrzeug durch Verkehrsraum und Verhalten anderer Verkehrsteilnehmer vor Fahraufgaben gestellt wird. Dabei ist das Verhalten des Testfahrzeugs nicht restringiert, d. h., in einer Simulation kann das Testfahrzeug mit einem Automationssystem instanziiert werden, dessen Leistung dann in der Simulation getestet werden kann. Die Beschreibungen sind auf dem Level logischer Szenarien angesiedelt. Das bedeutet, dass das Verhalten des umgebenden Verkehrs parametrisiert ist und variiert werden kann, so dass aus jedem Beispielszenario viele mögliche Testläufe abgeleitet werden können. Diese beispielhaften Szenarien wurden in den projektinternen Demonstrationen der realisierten Simulationsumgebung verwendet (SUC 1 bis 3). Wie auch andere Projektergebnisse, die für externe Anwender von Interesse sein können, wurden die Beispielszenarien öffentlich zugänglich gemacht.

Über die projektinternen Tätigkeiten hinaus trug TP 1 zusammen mit anderen Teilprojekten zu einer Weiterentwicklung der Standards OpenDRIVE und OpenSCENARIO bei der ASAM bei.

2.1.2 Ergebnisbeiträge von TP 1

2.1.2.1 Anforderungen an die Simulation

Das Arbeitspaket 1.1 erstellte die Anforderungen und das Glossar, beschrieb Use-Cases der Simulation und spezifizierte ein Explorationsverfahren für Szenarienräume.

2.1.2.2 Verkehrsräume und Referenzszenarien

Das Arbeitspaket 1.2 stellte Verkehrsräume und Referenzszenarien zusammen und arbeitete an der Weiterentwicklung von OpenDRIVE und OpenSCENARIO mit.

2.1.3 Detaillierte Darstellung der Ergebnisse von TP 1

2.1.3.1 Anforderungen

Da die funktionalen, qualitativen und strukturellen Anforderungen eng miteinander verknüpft sind, wurden die Anforderungen in Zusammenarbeit der Mitarbeitenden aller drei Unterarbeitspakete erhoben und gemeinsam in einer Tabelle dokumentiert. Als Sprache für die Formulierung wurde Englisch gewählt, um sie in möglichen internationalen Kooperationen unmittelbar verwenden zu können.
Um eine einheitliche Dokumentation aller Anforderungstypen und ein schnelles „Zurechtfinden“ innerhalb der Anforderungstabelle (sowohl von Projektpartnern aus anderen Unterarbeitspaketen als auch von Mitarbeitenden aus dem Projekt VV-Methoden, mit dem eng zusammengearbeitet wurde) zu ermöglichen, wurden die Anforderungen in einer einheitlichen Tabelle beschrieben. In dieser Tabelle sind die nachfolgend genannten sechs Tabellenblätter aufgeführt:

- Das erste Tabellenblatt beschreibt allgemeine Informationen zur Anforderungstabelle, beispielsweise die Definition der Begriffe funktionale, qualitative und strukturelle Anforderungen, Kriterien zur Definition von Zielen (SMART-Kriterien: **S**pecific, **M**easurable, **A**chievable, **R**easonable **T**ime-bound), Gütekriterien für Anforderungen und Schlüsselwörter mit Richtlinien zur Verwendung dieser.
- Das zweite Tabellenblatt beschreibt die Strukturierung der Anforderungen. Dabei soll diese nicht zwangsweise eine Architektur für die Implementierung vorgeben, sondern die Anforderungen selbst strukturieren und in logische Module gruppieren. Die einzelnen Anforderungen können bei der Implementierung anderen „Modulen“ zugeordnet werden. Die Erstellung dieser Architektur wurde im Arbeitspaket 2.1 bearbeitet.
- Das dritte Tabellenblatt beschreibt wichtige Begriffe im Kontext der Anforderungen. Diese Begriffe sind auch im projektweiten Glossar enthalten, welches durch die eingesetzte zentrale Versionsverwaltung GitLab allen Projektbeteiligten zugänglich war.
- Die eigentlichen Anforderungen sind in den drei weiteren Tabellenblättern beschrieben (aufgeteilt in funktionale, qualitative und strukturelle Anforderungen). Der jeweils aktuelle Status jeder Anforderung wurde während der Projektlaufzeit mit Hilfe einer Kennzeichnung als „DRAFT“, „TO REVIEW“, „IN REVIEW“, „TP 1 ACCEPTED“ und „GENERALLY ACCEPTED“ erfasst.

Die funktionalen, qualitativen und strukturellen Anforderungen wurden basierend auf den im Projekt betrachteten Anwendungsfällen systematisch strukturiert und kategorisiert (vgl.

Beschreibung des zweiten Tabellenblatts oben). Während der Projektlaufzeit wurde der jeweilige Stand der Anforderungstabelle in der zentralen Versionsverwaltung GitLab abgelegt, sodass die Tabelle für alle Projektbeteiligten jederzeit verfügbar war. In mehreren Iterationen wurde diese Tabelle einem projektweiten Review unterzogen und die erhaltenen Kommentare eingearbeitet. Dabei erfolgte zusätzlich ein Abgleich der erhobenen Anforderungen mit den im Projekt SET Level verwendeten Simulation Use-Cases. Die Anforderungstabelle wurde auch an das Projekt VVMethoden mit der Bitte um Kommentierung weitergegeben und in Telefonkonferenzen mit Mitarbeitenden aus beiden Projekten diskutiert und abgestimmt. Für jeden Meilenstein wurde der resultierende Stand der Anforderungstabelle schreibgeschützt zentral für alle Projektbeteiligten in der im Projekt eingesetzten Versionsverwaltungssoftware GitLab zur Dokumentation abgelegt.
Insgesamt wurden 147 funktionale Anforderungen erhoben und in die Anforderungstabelle eingetragen (22 Top-Level Anforderungen und 105 davon abgeleitete Anforderungen). Weiterhin wurden 41 qualitative Anforderungen erhoben und in die Anforderungstabelle eingetragen (17 Top-Level Anforderungen und 24 davon abgeleitete Anforderungen). Zudem wurden 23 strukturelle Anforderungen erhoben und in die Anforderungstabelle eingetragen (18 Top-Level Anforderungen und 5 davon abgeleitete Anforderungen).
Die Anforderungen sind dabei alle identisch aufgebaut und bestehen aus bis zu 13 Feldern die jeweils durch eine eigene Spalte in der Tabelle abgebildet werden.

- Die ersten drei Spalten sind vorgesehen für eine eindeutige Identifizierung der Anforderungen. „ID“ ist dabei die ID der Top-Level-Anforderung. „ID2“ und „ID3“ werden für Unteranforderungen vergeben. Jede ID beginnt mit einem Buchstaben: „F“ für funktionale Anforderung, „Q“ für qualitative Anforderung und „S“ für strukturelle Anforderung.
- Die Spalte „Derived from“ dient der Angabe weiterer Anforderungen aus denen die aktuelle Anforderung abgeleitet wurde.
- Die Spalte „Referenced by“ dient im Umkehrschluss zur Referenzierung von Anforderungen, die im Bezug zur aktuellen Anforderung stehen.
- Die Spalte „Name“ enthält einen eindeutigen Namen für die jeweilige Anforderung.
- In der Spalte „Simulation goal“ wird das Simulationsziel, für welches die jeweilige Anforderung relevant ist, angegeben.
- Die Spalte „Requirement“ enthält den eigentlichen Anforderungssatz.
- Die Spalte „Explanation“ kann zusätzlichen Erläuterungstext zur Anforderung enthalten.
- Die Spalte „Category“ dient zur Kategorisierung und Strukturierung der Anforderung hinsichtlich einer groben Simulationsarchitektur.
- Die Spalte „History“ dient zur Nachverfolgbarkeit der Historie der Erstellung und eventuell vorhandener Änderungen.
- In der Spalte „State“ wird der oben genannte Status der Anforderung festgehalten.
- In der letzten Spalte „Comment TP 1“ konnten während der Anforderungserhebungsphase Kommentare für die TP-interne Diskussion der Anforderungen hinterlassen werden.

Zur besseren Illustration sind in Abbildung 17 und Abbildung 18 eine Top-Level-Anforderung mit einer Unteranforderung dargestellt. Aufgrund der Lesbarkeit wurden die Spalten auf zwei Abbildungen aufgeteilt.

ID	ID2	ID3	Name	Simulation goal	Requirement
F-18			**Manual interaction with platform management system**	**Criticality Analysis, Effectiveness Analysis, Verification, Validation**	**The platform management system shall enable manual interaction.**
	F-18-1		Manual concrete scenario selection	Criticality Analysis, Effectiveness Analysis, Validation, Verification	The platform management system shall enable the user to load a concrete scenario manually.

Abbildung 17: Illustration der Anforderungstabelle (1/2)

Explanation	Category	State	Comment TP1
This requirement is necessary to allow the user to interact with the platform management system manually, e.g., with a user interface.	**Platform Management System**	**COMMONLY ACCEPTED**	
	Platform Management System	COMMONLY ACCEPTED	**Scenario:** A scenario describes the temporal development between several scenes in a sequence of scenes. Every scenario starts with an initial scene. Actions & events as well as goals & values may be specified to characterize this temporal development in a scenario. Other than a scene, a scenario spans a certain amount of time.

Abbildung 18: Illustration der Anforderungstabelle (2/2)

Die abgeleiteten Anforderungen wurden als Basis für die Entwicklung einer funktionalen Architektur der szenarienbasierten Simulationsplattform verwendet (siehe AP 2.1). Zudem sind die Anforderungen in die Arbeiten von AP 2.3 eingeflossen.
Nach der projektinternen Abstimmung der Anforderungen (zeitlich kurz nach der Halbzeitpräsentation des Projektes Februar 2021) wurden nur noch Anpassungen an Begrifflichkeiten vorgenommen, um die Formulierungen in den Anforderungstabellen mit den späteren Projektentwicklungen konsistent zu halten.

2.1.3.2 Glossar

Ein zweisprachiges Glossar in Deutsch und Englisch listet eine Vielzahl projektrelevanter Begriffe mit ihren Definitionen auf. Anfangs nur innerhalb des Projektes entwickelt, wurde es später zusammen mit anderen öffentlich geförderten Projekten fortgeschrieben. Besonders eng war die Zusammenarbeit mit VVMethoden. Zum Halbzeitevent des Projektes VVMethoden im März 2022 wurde der damalige Stand des Glossars veröffentlicht (VVMethoden, 2022). Zu diesem Zeitpunkt waren 212 Begriffe aufgenommen. Später trugen auch die Projekte VIVALDI und Gaia-X4PLC zur Weiterführung des Glossars bei. Zum Ende des Projektes SET Level umfasste das Glossar nahezu 300 Begriffe, die in zunehmendem Umfang mit Quellen unterlegt worden waren. Nach dem Ende von SET Level arbeiten die drei genannten Projekte noch aktiv an der Weiterentwicklung des Glossars. Eine Veröffentlichung eines konsolidierten Standes der Begriffssammlung ist in Vorbereitung.

2.1.3.3 Use Cases der Simulation

Zu den Grundannahmen des Projektes SET Level gehört, dass Simulation vielfältige Aufgaben im Entwicklungsprozess hochautomatisierter Fahrzeuge und ihrer Komponenten

übernehmen wird. Entsprechend waren Modularität und Konfigurierbarkeit als Anforderungen an die Simulation formuliert worden. Einige Anwendungsmöglichkeiten der Simulationswerkzeuge wurden zu Demonstrations- und Testzwecken während der Projektlaufzeit realisiert (siehe Abschnitt 2.3.3.2, 2.3.3.3 und 2.3.3.4). Darüber hinausgehend wurden weitere Anwendungen in (Hungar, 2021) systematisch beschrieben.
Beim ersten beschriebenen Anwendungsfall handelt es sich um „Scenario Mining“. Dabei werden in einer frühen Entwicklungsphase Szenarien identifiziert, die während der Automationsentwicklung bei den Kritikalitätsanalysen berücksichtigt werden sollten. Dieses Verfahren basiert auf einer Kopplung von Simulation der Mikro-Ebene (z. B. mit SUMO, http://www.eclipse.org/sumo/) mit Simulation auf der Nano-Ebene[5], wie sie im Fokus des Projektes SET Level stand. Diese Anwendungsmöglichkeit wurde prototypisch realisiert und beim Abschlussevent (siehe https://setlevel.de/en/final-presentation/booths/goals-and-usages-of-simulation) vorgestellt.
Der zweite dargestellte Anwendungsfall betrifft die Validierung eines weit entwickelten Systems – „Deep Validation“. Wenn eine Simulationskonfiguration mit hoher Realitätstreue zur Verfügung steht und der Testraum in genügend gut ausgearbeiteten logischen Szenarienbeschreibungen erfasst ist, kann eine Abschätzung des Sicherheitsrisikos, das mit der Einführung des automatisierten Fahrzeugs in den Verkehr einhergeht, simulativ berechnet werden.
Die dritte Anwendung „Incident Analysis“ adressiert die Monitoringphase einer Fahrzeugautomation, also die Überwachung des Produktes in der praktischen Nutzung. Hier kann Simulation zur Untersuchung der Ursachen von Vorfällen im Realverkehr eingesetzt werden. Über die Standardfunktionalitäten von Simulationsumgebungen sind hierfür Werkzeuge erforderlich, die beobachtete Realverkehrsabläufe in variierbare (also logische) Szenarien umsetzen („Replay-to-Sim“).
Diese Beispiele zeigen auf, dass Werkzeuge, die dem Ansatz des Projektes SET Level entsprechen, weithin und flexibel bei der Entwicklung von Fahrzeugautomationssystemen eingesetzt werden können. Das Spektrum der Anwendungsmöglichkeiten geht sogar noch weit über die ausgearbeiteten Beispiele hinaus.

2.1.3.4 Explorationsverfahren

Im szenarienbasierten Testen wird das Testobjekt – etwa ein automatisiertes Fahrzeug oder eine Automationskomponente – in Anwendungsszenarien geprüft. Die Szenarien können beispielsweise in Form von logischen Szenarien vorliegen. Dies sind parametrisierte Szenarien, die jeweils eine Vielzahl von konkreten Testszenarien beschreiben, die durch Instanziieren der Parameter mit konkreten Werten entstehen. Parameter können diskrete oder kontinuierliche Wertebereiche haben. Kontinuierliche Parameter wie etwa Geschwindigkeit werden bei der Instanziierung diskretisiert. Auf diese Weise spannt ein logisches Szenario einen umfangreichen Testraum auf. Selbst bei einfachen Szenarien ist dieser aufgrund der inhärenten kombinatorischen Explosion schon so groß, dass nur ein Bruchteil der Instanzen in einem realistischen Zeitrahmen durchsimuliert werden kann. Allerdings ist es für vielerlei Testziele erforderlich, den Raum in einem gewissen, zielabhängigen Sinn *vollständig* abzudecken. Um solche Testaufgaben lösen zu können, ist es also erforderlich, Verfahren zur Exploration zu verwenden, die durch geschickte Wahl der durchsimulierten Instanzen das Testziel erreichen.

[5] Bezeichnungen für den Detaillierungsgrad von Straßenverkehrssimulationen werden nicht einheitlich verwendet. Meist wird eine Simulation, in der das individuelle Verhalten von Verkehrsteilnehmern (Geschwindigkeit, Wegewahl etc.) modelliert ist, bereits als Mikro-Simulation bezeichnet, so z. B in (Bungartz, 2013). Für eine Beurteilung der technischen Realisierung einer Automation ist jedoch eine noch detailliertere Modellierung erforderlich, weshalb hier von „Nano-Ebene“ gesprochen wird – es spielt dann nämlich noch die Wirkungsweise der Komponenten eines Fahrzeugs eine Rolle.

Abbildung 19 illustriert den prinzipiellen Aufbau einer Simulation für die Exploration eines logischen Szenarios. Die Basis stellt eine Simulationslösung dar, mit der eine konkrete Szenarioinstanz durchsimuliert wird. Hierfür kann ein geeignetes Werkzeug – je nach Abstraktionsgrad, Testobjekt, Bewertung etc. – eingesetzt werden. Die eigentliche Steuerung der Exploration ist in Abbildung 19 darüber abgebildet. Hier werden die Ergebnisse der einzelnen Simulationsläufe im Gesamtzusammenhang bewertet, und es wird ausgewählt, welche weiteren Läufe noch zu geschehen haben.

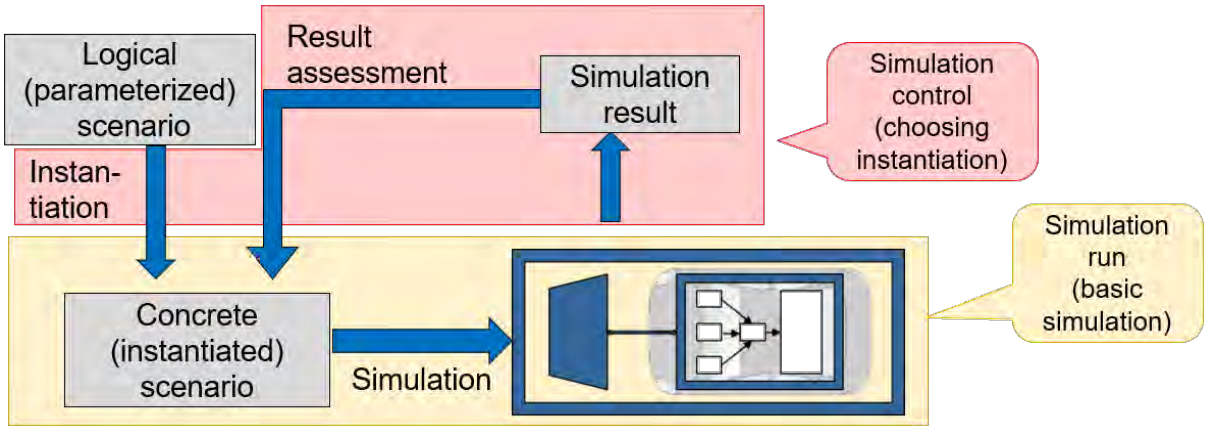


Abbildung 19: Illustration des Aufbaus einer Simulation zur Exploration von logischen Szenarien

Speziell für die Aufgabe, alle möglicherweise kritischen Instanzen eines logischen Szenarios garantiert zu identifizieren, wurde ein Verfahren konzipiert. Dieses beruht darauf, bei geeigneter Fassung der detaillierten Simulationsaufgabe, zuverlässig abschätzen zu können, um wie viel die Kritikalität der Verkehrsabläufe in einer Umgebung (bei Variation der konkreten Parameterwerte) eines durchgeführten Simulationslaufes maximal ansteigen kann. Neben anderen Bedingungen, die für die Abschätzungsmöglichkeit erfüllt sein müssen, spielen die Stetigkeitseigenschaften der Maßfunktion für Kritikalität hier eine wesentliche Rolle. Eine ausführliche Darstellung des Verfahrens wurde (Hungar, 2020) veröffentlicht.
Auch weitere Explorationsverfahren sind in dem Projekt betrachtet worden. Diese Arbeiten waren allerdings nicht im TP 1 verortet. Sie sind im Kapitel “Closed-loop Verkehrssimulation für Analyseaufgaben“ und (Schütt, et al., 2022) beschrieben.

2.1.3.5 Verkehrsräume und Referenzszenarien

Simulationsbasiertes Testen und Validieren von automatisierten Fahrfunktionen kann nur gute und vertrauenswürdige Ergebnisse liefern, wenn in der Simulation die für den Straßenverkehr relevanten, kritischen Szenarien abgetestet werden. Im Rahmen von SET Level wurden Simulationsumgebungen aufgebaut und die grundsätzliche Funktionsweise des simulations- und szenarienbasierten Testens demonstriert. Hierzu wurden beispielhaft Szenarienräume (sog. Verkehrsräume) und Referenzszenarien definiert, die untenstehend dargestellt sind. Für vollständige Tests etwa einer Komponente oder die Absicherung einer automatischen Fahrfunktion sind natürlich eine Vielzahl von Szenarien abzuprüfen. Da es im Projekt SET Level jedoch im Wesentlichen um Technologien und Methoden zur Simulation ging, war das Ziel, exemplarische Verkehrsräumen und ausgewählte Szenarien bereitzustellen, mit denen die Projektentwicklungen erprobt und demonstriert werden konnten. Diese dienten zur Demonstration von Werkzeugen und Methoden in den Simulation Use Cases SUC 1 bis SUC 3.

Als erprobte Beispiele für Test und Demonstration von Simulationswerkzeugen wurden die Ergebnisse unter der Lizenz „Creative Commons Attribution-ShareAlike 4.0 International“ (CC BY-SA 4.0) im öffentlichen GitLab des Projektes interessierten Anwendern zur Verfügung gestellt.

Definition
Der Begriff „Verkehrsraum” beschreibt die Summe der statischen Elemente aus dem Szenario. Dazu gehören beispielsweise die Straßengeometrie, Fahrstreifenmarkierungen, Vorfahrtsregeln, Verkehrsschilder und -ampeln, Gehwege, Zebrastreifen, aber auch die Randbebauung oder die verwendeten Materialien mit ihren Eigenschaften (Radarreflektivität o. ä.). Das „Szenario“ umfasst darüber hinaus auch die dynamischen Objekte – also im Wesentlichen die Verkehrsteilnehmer, wie Pkw, Lkw, Busse und Bahnen, Motorräder, Fahrräder und Fußgänger. Aber auch Skateboarder oder Rollerfahrer, Hunde und Katzen oder ein über die Straße rollender Fußball gehören in der Stadt dazu und dürfen in der Simulation nicht vergessen werden. Dem Verhalten dieser Objekte liegt jeweils ein Modell zugrunde, das zusammen mit den Zielvorgaben (Straße überqueren, abbiegen, stehen bleiben, usw.) und speziell gewählten Parametern (Abstände, Geschwindigkeiten, Zeitpunkte, usw.) mögliche Trajektorien in der Simulation erzeugt. Schließlich sind auch Wetterbedingungen und Lichtverhältnisse sowie evtl. vorhandene digitale Infrastruktur Bestandteil der Szenarienbeschreibung.

Methodische Erarbeitung
Die Grundlage für Verkehrsräume und Szenarien bildet das sog. 6-Ebenen-Modell für die Beschreibung von Szenarien, das bereits vor einigen Jahren im Rahmen des vom BMWK geförderten Vorhabens PEGASUS entwickelt wurde. Wie Abbildung 20 zeigt, beschreibt Ebene 1 die Straßengeometrie und Ebene 2 die Verkehrsregelungen mit Verkehrszeichen und anderen Elementen, Randbebauung und Straßenbegleitgrün. Ebene 3 stellt vorübergehende Änderungen der Streckenführung oder Verkehrsregelung dar, wie sie zum Beispiel durch Baustellen verursacht werden. Die beweglichen Objekte sind in Ebene 4 verankert, Umweltbedingungen in Ebene 5, und Ebene 6 ist reserviert für die Übertragung von digitalen Informationen via Mobilfunk, von ITS-Roadstations oder Car-to-Car-Kommunikation.
Beginnend mit einem einfachen Szenario in einem einfachen Verkehrsraum wurde die Komplexität nach und nach gesteigert und an die Bedürfnisse der Simulation Use Cases angepasst. Daraus ergab sich auch die Erweiterung der Szenarien um sog. Meta-Daten, die eine Klassifizierung der Szenarien erlauben.
Im Verlauf des Vorhabens wurden mehrere Verkehrsräume und dazugehörige Szenarien entwickelt, um mit den Anforderungen der Nutzer Schritt zu halten. Verkehrsraum 0 (null) besteht aus einer geraden, ebenen Straße mit einem Fahrstreifen je Fahrtrichtung, einem Mittelstreifen und je einem Randstreifen auf jeder Straßenseite.

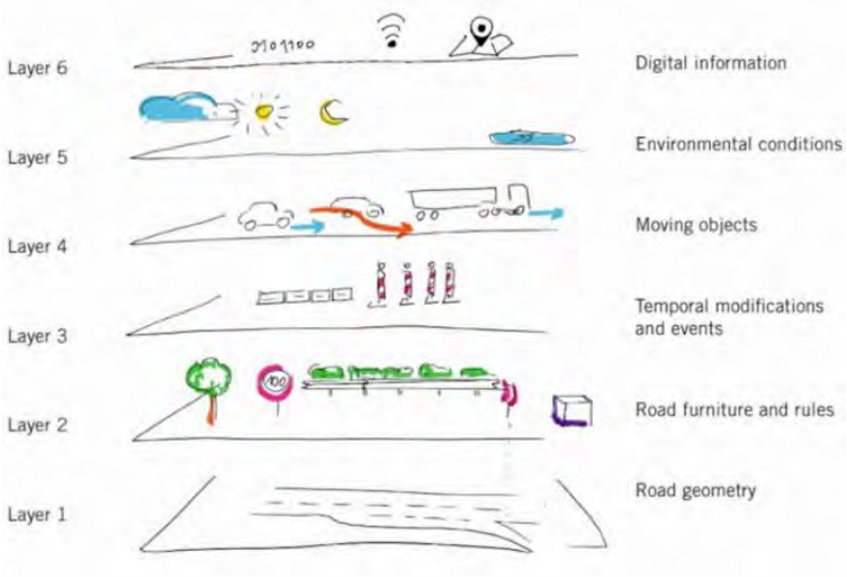


Abbildung 20: 6-Ebenen-Modell aus PEGASUS

In Referenzszenario RS0 (Abbildung 21) fahren zwei Fahrzeuge in entgegengesetzter Richtung, und zwei Fußgänger überqueren die Straße. Damit lässt sich die Reaktion der Fahrfunktion auf Fußgänger testen.

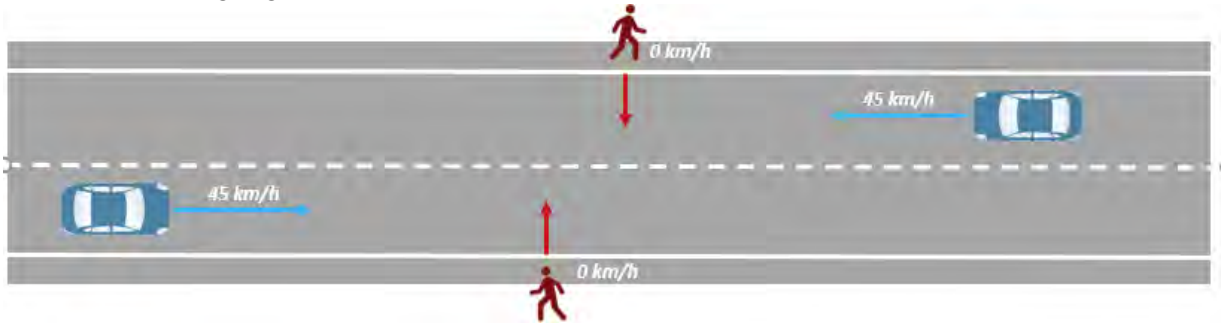


Abbildung 21: Verkehrsraum TS0 mit Referenzszenario RS0

Verkehrsraum 1 zeigt eine städtische Einfallstraße mit einem Fahrstreifen je Fahrtrichtung, die durch eine durchgehende Linie getrennt sind. Von rechts nähert sich eine (Einbahn-) Straße, die über einen Einfädelstreifen in die Einfallstraße übergeht. Mit Referenzszenario RS1 (siehe Abbildung 22) lassen sich einfache Einfädelszenarien simulieren, so dass man z. B. testen kann, wie eine automatisierte Fahrfunktion auf unterschiedliche Abstände und Geschwindigkeiten reagiert.

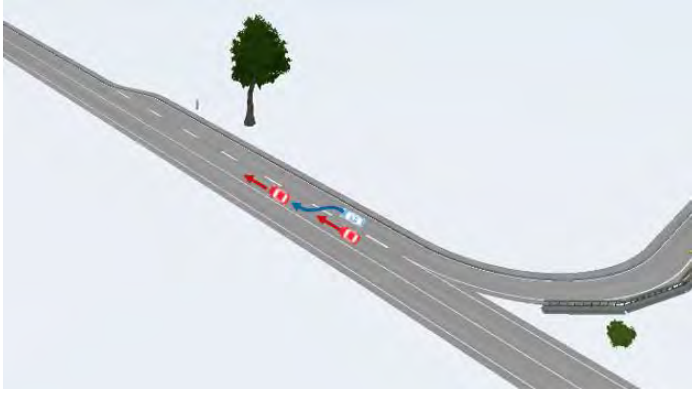


Abbildung 22: Verkehrsraum TS1 mit Referenzszenario RS1

Zu Meilenstein 2, der zeitlich etwa mit der Halbzeitveranstaltung des Vorhabens SET Level zusammenfiel, wurden einfache Abbiegeszenarien dargestellt, bei denen mehrere Verkehrsteilnehmer in Interaktion treten. Hierzu wurde Verkehrsraum TS2 entwickelt (SET Level, 2022). Er besteht aus einer vierarmigen, rechtwinkligen Kreuzung mit einem Fahrtreifen je Richtung und einem Gehweg auf jeder Seite. Die Straße in Ost-West-Richtung ist vorfahrtsberechtigt, während in Nord-Süd-Richtung „Vorfahrt gewähren“-Schilder angebracht sind. Teilweise ist eine Straßenrandbebauung vorhanden; auf der Nordost-Ecke befindet sich eine Parkfläche (siehe Abbildung 23).

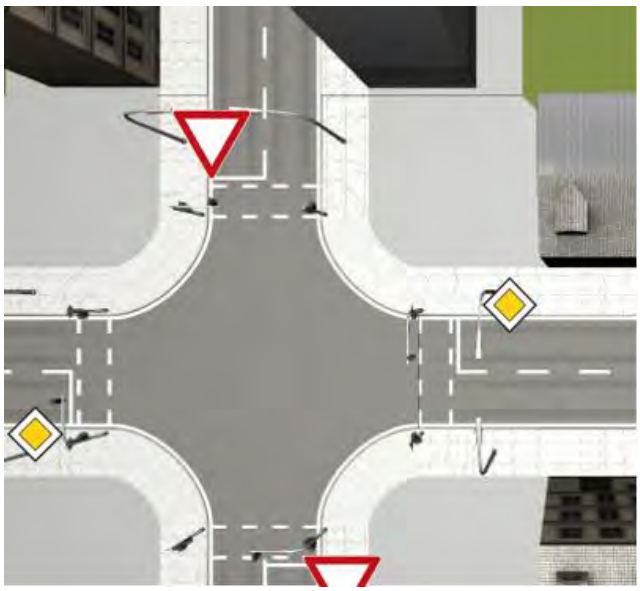


Abbildung 23: Einfache Kreuzung, Verkehrsraum TS2

Für Referenzszenario RS2 (siehe Abbildung 24) wurde dieser Verkehrsraum mit diversen Verkehrsteilnehmern bestückt. Insgesamt fünf Pkw und sieben Fußgänger bilden ein realitätsnahes, städtisches Szenario. Das Ego-Fahrzeug (in blau dargestellt) nähert sich von rechts der Kreuzung und möchte nach rechts in den nördlichen Arm abbiegen. Dabei sind kreuzende Fußgänger zu beachten, und die Sicht nach rechts ist durch einen parkenden Pkw erschwert.

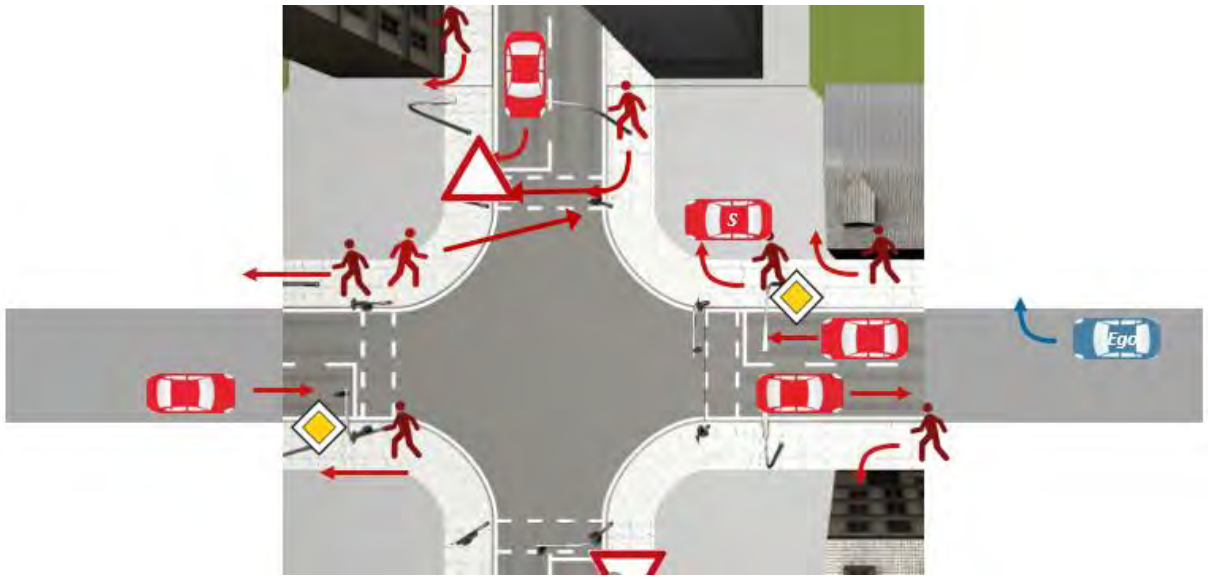


Abbildung 24: Referenzszenario RS2: Rechtsabbieger mit querenden Fußgängern.

Für Meilenstein 3, der zeitlich kurz vor Projektabschluss lag, wurde die Forschungskreuzung des DLR in Braunschweig als Verkehrsraum ausgewählt. Wie in Abbildung 25 zu erkennen ist, handelt es sich um eine vierarmige Kreuzung mit bis zu fünf Fahrstreifen je Fahrtrichtung. Daneben sind Fahrradwege und Gehwege sowie Fußgängerüberwege vorhanden. Der Verkehr wird über eine Lichtsignalanlage geregelt.

Abbildung 25: Komplexe Kreuzung, Verkehrsraum TS3

Für die Simulation wurde ein Szenario erarbeitet (siehe Abbildung 26), bei dem das Ego-Fahrzeug von links in die Kreuzung einfährt und nach links abbiegt, ohne dass hierfür eine spezielle Linksabbieger-Ampel vorhanden wäre. Der Gegenverkehr ist somit zu beachten, ebenso wie Fußgänger und Radfahrer. Das Szenario wurde in verschiedenen Ausprägungen umgesetzt, bei denen die Fahrzeuge im Gegenverkehr unterschiedliche Abstände haben oder beispielweise auf der Kreuzung noch einen Fahrsteifenwechsel durchführen. Auf diese Weise entstehen unterschiedliche Situationen, die ggf. eine Herausforderung für die zu testende automatisierte Fahrfunktion darstellen könnten.

Abbildung 26: Referenzszenario RS3: Linksabbieger auf komplexer Kreuzung

Die wesentlichen Verkehrsräume und Szenarien, die im Verlauf des Vorhabens erarbeitet wurden, sind auf dem SET Level Public GitLab (SET Level, 2022) öffentlich zugänglich. Dort findet sich auch eine ausführliche Beschreibung der einzelnen Szenarien.
Auf der SET Level Abschlussveranstaltung am 11. und 12. Oktober 2022 wurden sämtliche Ergebnisse präsentiert und anschließend auf der SET Level Webseite (SET Level, 2022) dokumentiert. Weiterhin sind dort (SET Level, 2022) unter anderem auch Videosequenzen abgelegt, die einzelne Verkehrsräume und Szenarien zeigen – dort als Grundlage für die Kritikalitätsanalyse im Rahmen von SUC 1.

2.1.3.6 Konzepte der Test- und Szenarienbeschreibung

Weitere Beiträge des AP 1.2 zu den Projektresultaten betrafen die Erfassung von Simulationszielen und die Sprachmittel zur Beschreibung von Szenarien.
Die Arbeiten zu den Simulationszielen wurden zusammen mit dem AP 2.2 und AP 4.1 durchgeführt, und die Ergebnisse sind im Zusammenhang im Abschnitt 2.3.3.1.2 „Simulationsziele“ dieses Berichtes dargestellt.
Die wesentlichen Formate zur Beschreibung von Szenarien sind OpenDRIVE und OpenSCENARIO. Beide Formate werden kontinuierlich weiterentwickelt, um den wachsenden konzeptionellen und technischen Anforderungen gerecht zu werden. Seit etlichen Jahren koordiniert der ASAM e.V. (Association for Standardization of Automation and Measuring Systems) diesen Prozess. Während der Projektlaufzeit wurden von OpenDRIVE die Versionen 1.6 und 1.7 bei der ASAM veröffentlicht, von OpenSCENARIO die Versionen 1.0, 1.1 und 1.2. Diese Standards spielen eine große Rolle für die Simulationswerkzeug-unabhängige Formulierung von Szenarien. An anderer Stelle in diesem Bericht wird deren Anwendung im Projekt SET Level beschrieben, vornehmlich im Kapitel „Teilprojekt 4: Instanziierung von Werkzeugketten für Entwicklung und Testen“. Daneben wurde noch der Standard OpenSCENARIO 2.0 im Juli 2022 veröffentlicht. Dabei handelt es sich nicht um eine Weiterentwicklung der OpenSCENARIO-1.x-Reihe, sondern um einen Ansatz, der Beschreibungen auf einem höheren Abstraktionsgrad ermöglicht. Langfristig sollen die Möglichkeiten der 1.x-Reihe auch darin abgedeckt werden können. Derzeit ist die 2.0-Version noch nicht für die im Projekt SET Level betrachteten und entwickelten Simulationswerkzeuge relevant, da die Konkretisierungsmöglichkeiten noch ausgebaut werden müssen. Langfristig sollen beide Entwicklungsreihen zusammengeführt werden, indem die Ausdrucksmöglichkeiten der 1.x-Versionen mit abgedeckt werden.
An der Ausarbeitung dieser Standards haben sich Mitarbeitende des Projektes SET Level beteiligt. Das Schwergewicht der Beteiligung betraf die OpenSCENARIO-1.x-Versionen. Darüber hinaus wurden noch anhand der praktischen Arbeiten zu den Verkehrsräumen und Referenzszenarien (Kapitel 2.1.3.5 „Verkehrsräume und Referenzszenarien“) vielfältige Erfahrungen gemacht. Insbesondere wurde deutlich, dass die Definitionen der Formate noch verbessert werden müssen, da divergierende Interpretationsmöglichkeiten in der Praxis noch nicht zu der gewünschten Werkzeugunabhängigkeit führen. In den Referenzszenarien findet man diese Erfahrungen in der gewählten Fassung widergespiegelt – möglicherweise problematische Konstrukte sind dort weitgehend vermieden. Eine weitere Beobachtung betrifft die Parameterisierungsmöglichkeiten von OpenSCENARIO. Diese sind noch auszubauen, z. B., was die Definition von Trajektorien betrifft.

2.1 Teilprojekt 2: Simulationsbasiertes und virtuelles Entwickeln / Testen

2.1.1 Übersicht über Inhalte und Ziele von TP 2

TP 2 sollte dazu dienen, die bereits im Projekt PEGASUS entwickelten Methoden und Werkzeuge zum simulationsbasierten Testen entsprechend der im Projekt SET Level betrachteten Anforderungen für höhere Automatisierungslevel und in urbanen Umgebungen weiterzuentwickeln. Zudem mussten neuartige Methoden entwickelt werden, die eine Automatisierung im Bereich der Anforderungsermittlung in Entwicklungsprozessen umsetzen, verschiedene Aufgaben zur Zwischenprüfung von Entwicklungsartefakten durch automatisierte Prozessabläufe unterstützen und im Zusammenhang mit der Testdurchführung in simulationsbasierten Umgebungen ebenfalls automatisierte Abläufe in Werkzeugen (IT-Tools) spezifizier- und verfügbar machen.
Eine besondere Herausforderung lag im Bereich der Integration verfügbarer bzw. im Projekt zu erstellender Simulationsmodelle (insbesondere auch für die Umwelt und für andere Verkehrsteilnehmer) in komplexen Gesamtsystemen. Beim Aufbau der Methoden wurde eine integrierte Nutzung mit Prüfständen, dem Testen im nicht-öffentlichen Raum und der Testdurchführung im öffentlichen Raum, berücksichtigt, da sich hieraus wichtige Hinweise für die Wahl der Modellgranularität bzw. des Detailgrads der Modelle ergeben, die in Simulationen gehandhabt werden müssen.
Es wurden in diesem Teilprojekt drei Arbeitspakete definiert:

- AP 2.1 (Aufbau einer einheitlichen Integrationsarchitektur): Bei diesem Arbeitspaket lag der Fokus auf Anforderungen, Konzepten und Werkzeugen zur Beschreibung von kopplungsrelevanten Modelleigenschaften und deren werkzeugbasierter Handhabung/Nutzung sowie relevanter Schnitte mit Fokus auf die Anwendungsfelder Entwicklung und Testen.
- AP 2.2 (Integrationsmechanismen und Methoden): Bei diesem Arbeitspaket standen generische Mechanismen zur Modellkopplung in Simulationen im Mittelpunkt, die insbesondere das Auftreten potenzieller Artefakte im Zusammenhang mit der Modellkopplung beobachten bzw. diese kompensieren helfen.
- AP 2.3 (Anforderungen an Ausführungsumgebungen): Bei diesem Arbeitspaket standen verteilte und performante Ausführungsumgebungen im Sinne von Werkzeug-/Technologieplattformen sowie deren Anforderungsentwicklung im Mittelpunkt.

2.1.1.1 Aufbau einer einheitlichen Integrationsarchitektur

In diesem Arbeitspaket lag der Fokus auf Anforderungen, Konzepten und Werkzeugen zur Beschreibung von kopplungsrelevanten Modelleigenschaften und deren werkzeugbasierter Handhabung und Nutzung sowie relevanter Schnitte mit Fokus auf die Anwendungsfelder Entwicklung und Testen. Die Anforderungen detaillierten die Vorgaben aus TP 1. Eine Anwendbarkeit auf die Simulation für das Entwickeln von Komponenten oder Systemen, auf die Sicherheitsbewertung und die Kritikalitätsanalyse musste möglich sein. Trotz der hohen Komplexität wurde darauf geachtet, dass die entstehende Lösung möglichst einfach und robust in der Handhabung bleibt.
Gegenstand der Betrachtung war das Gesamtsystem, also neben dem Ego-Fahrzeug mit all seinen Komponenten (Sensoren, Aktoren, Software, Fahrdynamik) auch die Umgebung des Fahrzeugs (Straßen, Infrastruktur, Verkehrsteilnehmer, Wettersituation etc.). Die zu entwickelnde Integrationsarchitektur musste es ermöglichen, dass innerhalb einer einheitlichen Signal- und Schnittstellenarchitektur dennoch Fahrzeugarchitekturen unterschiedlicher Ausprägung dargestellt werden können. Dies betraf sowohl die Einbindbarkeit von unterschiedlichen

Komponenten als auch die Nutzung von Komponentenmodellen mit unterschiedlichen Modellierungstiefen. Die Beschreibung der Integrationsarchitektur wurde in einer Tool-neutralen Form erstellt, die es ermöglicht, einen Austausch zwischen unterschiedlichen Tools durchzuführen (Übertragbarkeit). Es wurden die Möglichkeiten vorhandener Standards wie SSP oder ASAM OSI auf ihre Verwendbarkeit hin überprüft und gegebenenfalls fehlende Aspekte in den entsprechenden Gremien eingebracht. Zudem wurde die Durchgängigkeit der Integrationsarchitektur erreicht. Dadurch wurde es möglich, eine konkrete Implementierung einer Fahrzeugarchitektur in verschiedenen Simulationsplattformen einzusetzen, wie z. B. Entwicklungsrechner als auch Rechencluster (SiL) oder Prüfstände (HiL). Die Schnitte des Systems konnten auch so gelegt werden, dass eine Gesamtsimulation verteilt über mehrere Rechnereinheiten stattfinden kann. Von der Schnittstellenarchitektur war auch die Trennung der Simulation von der Umwelt auf der einen Seite und der Sensorik auf der anderen Seite betroffen, die je nach Sensortechnologie (Kamera, Lidar, Radar, Ultraschall, etc.) unterschiedliche Ausprägungen haben kann. Um auch in der Lage zu sein, das reale Kommunikationsverhalten von komplexen Systemen, deren Software und Algorithmen auf mehreren Steuergeräten verteilt ist, zu simulieren, wurden entsprechende Artefakte und Latenzzeiten in der Integrationsarchitektur berücksichtigt. Bestehende Lösungen wurden so weit wie möglich genutzt und bei Bedarf erweitert. Die entwickelte Lösung sollte auch für Simulationen nutzbar sein, die schneller als Echtzeit laufen. Implikationen, die sich aus dieser Forderung ergeben, wurden entsprechend berücksichtigt. Die Integrationsarchitektur wurde in einer ausführbaren Art geliefert. Dadurch können konkrete Applikationen direkt durch Einbindung von der ausgewählten Simulationseinbindung und Definition aller benötigen Parameter in die ausführbare Integrationsarchitektur zu einer lauffähigen Simulation erstellt werden.
Sowohl das Projekt SET Level als auch andere nationale und internationale Projekte nutzen szenarienbasierte Simulationen. Ein Szenario beschreibt dabei auf einer abstrakten Ebene zum einen die notwendige Umgebung für den zu simulierenden Test, also Straßennetz, Infrastruktur, Verkehrsteilnehmer mit ihren Intentionen oder Bewegungsabläufen, und zum anderen auch die Fahraufgabe des Ego-Fahrzeugs. Für die Szenarienbeschreibung wurde auf dem Standard ASAM OpenSCENARIO aufgesetzt, der schon im Projekt PEGASUS genutzt wurde. Dabei war zu prüfen, ob bereits alle Anforderungen erfüllt werden, die sich aus den Simulationsaufgaben für höhere Automatisierungslevel im urbanen Raum ergeben. Bei Bedarf wurde der Standard innerhalb der entsprechenden Gremien dahingehend erweitert. Die wichtigsten Einsatzzwecke der Simulation mussten abgedeckt werden. Es wurde auch geprüft, ob es darüber hinaus noch weitere Elemente gibt, die für eine umfassende Simulierbarkeit höherer Automatisierungslevel benötigt werden. Da auch für die Umsetzung der Szenarien galt, dass sie sowohl übertragbar (verschiedene Simulationstools) als auch durchgängig (verschiedene Simulationsplattformen) sein soll, wurde eine hinreichend genaue Beziehung zwischen Elementen der Szenarienbeschreibung und den umsetzenden Simulationskomponenten sichergestellt. Standards wurden auf Spezifikationslücken geprüft, die ebenfalls zu unterschiedlichen Ergebnissen führen können. Es musste gewährleistet werden, dass in allen Fällen annähernd gleiche Simulationsergebnisse erzeugt werden. Um die Simulation für die Absicherung und Freigabe von Systemen auf höheren Automatisierungsstufen einsetzen zu können, musste sichergestellt werden, dass die Ergebnisse ausreichend valide sind. Um dies zu erreichen, mussten mehrere Bestandteile der Simulation ihre Validität nachweisen:

- Die verwendeten Simulationstools mussten geeignet nachweisen, dass die berechneten Ergebnisse hinreichend korrekt sind unter der Annahme, dass die eingesetzten Simulationsmodelle ausreichend valide Ergebnisse liefern. Es wurden Methoden zur Qualifizierung der Simulationstools benannt. Dies beinhalteten auch die Verwendung der Simulationssolver innerhalb der Simulationstools.

- Für jedes eingesetzte Simulationsmodell musste geeignet nachgewiesen werden, dass die berechneten Ergebnisse für den angedachten Einsatzzweck des Simulationsmodells und bei zweckgemäßer Anwendung in der Simulationsumgebung ausreichend valide sind. Es wurden Methoden genannt, wie die Simulationsmodelle validiert werden können im Hinblick auf eine zweckgemäße Anwendung im Simulationsverbund validiert werden können. Zusätzlich wurden Anforderungen an andere Aspekte der Validierung von Simulationsmodellen definiert.
- Es wurden Methoden zur Quantifizierung von Simulationsfehlern aus den numerischen Eigenschaften der Simulationssolver entwickelt.
- Unumgängliche Grenzen der Simulation wurden aufgezeigt und Empfehlungen abgeleitet, wie damit umzugehen ist. Soweit Normen zu diesem Themengebiet existieren, wurden diese möglichst übernommen. Da die Integrationsarchitektur im gesamten Entwicklungsprozess eingesetzt werden sollte und auch die finale Freigabe an konkrete Entwicklungsstände geknüpft werden musste, war ein exaktes Änderungsmanagement unerlässlich. Es musste daher überprüft werden, ob sich daraus zusätzliche Anforderungen an die Integrationsarchitektur ergaben. Diese Anforderungen fanden entsprechend Berücksichtigung.

Dazu beschäftigten sich Mitarbeitende im Projekt SET Level mit dem Aufbau einer einheitlichen Integrationsarchitektur und definierten alle an der Simulation beteiligten Module und die Informationsflüsse zwischen den einzelnen Teilen. Damit einhergehend wurden unterschiedliche Schnittstellen zu bestehenden und verwendeten Simulationsumgebungen definiert und Beschreibungen der Teilmodelle erstellt. Die Definition verwendeter Begriffe sowie gegebenenfalls unterschiedliche Definitionen wurden untersucht und geklärt. Die erstellte Simulationsarchitektur bildete damit die Grundlage für alle anderen Teilprojekte.
Außerdem wurden als Teilbereich vom Gesamtsystem verschiedene Simulationsmethoden von Kommunikationsartefakten erarbeitet und dokumentiert. Daneben wurde eine Referenzarchitektur erarbeitet und somit eine funktionale Systemarchitektur aus austauschbaren Komponenten mit der entsprechenden Darstellung beschrieben.
Spezifikationen zur Szenarienbeschreibung und die damit verbundenen Implementierungen von Test- und Analyseaufgaben sowie die Formulierung zugrundeliegender Anforderungen wurden untersucht. Dabei wurden die verwendeten Verkehrsszenarien und -räume dokumentiert und beschrieben sowie einfache Karten Szenarien umgesetzt. Daneben wurden die Beziehungen zwischen Szenarienbeschreibung und Verhaltensmodellen untersucht, um einer standardisierten Einsetzbarkeit von verschiedenen Agentenmodellen näher zu kommen. Da solche Modelle für die unterschiedlichen Aspekte von bestimmten Simulationen nur bedingt geeignet sein könnten, mussten Beschreibungen für die Eignung dieser Modelle für den vorgesehenen Einsatzzweck gefunden werden. Dazu kamen die unterschiedlichen Informationsflüsse von und zu den eingesetzten Agentenmodellen.
Verschiedene Methoden zur Validierung von Modellen, welche in den Simulationsumgebungen eingesetzt werden sollten, wurden untersucht und die erarbeiteten Ergebnisse bewertet. Dies beinhaltet die Anforderung an Validierung und Qualifizierung generell. Die betrachteten Verifikationstechniken sollten genutzt werden, um einen Grad der Vertrauenswürdigkeit beschreiben zu können. Zusätzlich wurde der Umgang von Grenzen der Simulation untersucht, um Probleme feststellen zu können und die unterschiedlichen Anforderungen zu definieren, welche an die beteiligten Komponenten der Simulation gestellt werden können.
Verschiedene Methoden zur Qualifikation von Tools und Software, welche in der Simulation verwendet werden, wurden ebenfalls untersucht, um diese hinsichtlich der Eignung für den jeweiligen Anwendungsfall bewerten zu können.

Eine Methode zur Quantifizierung von Fehlern wurde erarbeitet, welche bei der Modellierung und Simulation von Systemen auftreten können. Schlussendlich wurde die prozesssichere Gesamtbeschreibung einer Simulationsaufgabe erarbeitet. Sie dokumentierte die Simulation Use Cases (SUCs) in Hinsicht auf den Credible Simulation Process.

2.1.1.2 Entwicklung von Mechanismen für die Kopplung von verschiedenen Simulationsmodellen (Integrationsmechanismen und Methoden)

Das zweite Arbeitspaket in TP 2 beschäftigte sich mit der Kopplung und den Schnittstellen von Modellen zur simulativen Absicherung von Systemen höherer Automatisierungslevel. Dieses Arbeitspaket lieferte zum einen Ansätze für die konkrete numerische Kopplung und zum anderen Aussagen über Anforderungen an die Kopplung, um gewisse Qualitätsziele zu erreichen und auf Limitierungen hinzuweisen.
Diese Kopplung betrifft einerseits die Kopplung von unterschiedlichen einzelnen Modellteilen für eine Simulation des Gesamtsystems (Boundary/Core-Modelle) unter Berücksichtigung der jeweiligen Simulationsstufe (z. B. MiL, SiL, HiL) sowie des Simulationsziels. Andererseits waren Methoden zur Abstraktion von gleichartigen Modellen bzgl. unterschiedlicher Granularität sowie Reifegrad zu untersuchen. Hierzu wurden allgemeine Ansätze zur Spezifikation und formalisierten Beschreibung von Modellen für unterschiedliche Einsatzzwecke erarbeitet. Als Grundlage dienen Anforderungen und Rahmenbedingungen aus dem Arbeitspaket 2.1 „Integrationsarchitektur“. Die zu erarbeitenden Ansätze dienten im Anschluss als Grundlage und Strukturierung für die Umsetzung einzelner Modelle im Rahmen von TP 3.
Potenziell auftretende Artefakte im Zusammenhang mit der Modellkopplung wurden über zu erarbeitende Mechanismen erkannt beziehungsweise gegebenenfalls kompensiert. Zu deren Quantifizierung wurden Gütekriterien und Maße für Modelle und Simulationsplattformen entwickelt. Notwendige und ausreichende Detaillierungsgrade von Modellen in Abhängigkeit der Architekturen wurden analysiert und daraus bestimmt, welche Modelle für welches Simulationsziel geeignet sind.
Ein wichtiger Aspekt ist die Übertragbarkeit von Modellen und Simulationen zwischen unterschiedlichen Werkzeugen bzw. Ausführungsplattformen. Hierbei stehen generische Ansätze für die Kopplung von Modellen im Vordergrund, um von der frühen Entwicklungsphase bis hin zur Absicherung mit einer Zielplattform eine Übertragbarkeit zu gewährleisten. Ein enger Austausch mit übergreifenden Ansätzen aus dem Projekt VVMethoden hat stattgefunden.
Als Basis für die Schnittstellenbeschreibung wurde auf dem etablierten Standard Functional Mockup Interface (FMI) aufgesetzt. Dieser ist im Hinblick auf notwendige Erweiterungen der Simulation für Systeme höherer Automatisierungslevel zu untersuchen. Relevante Lösungsansätze wurden gemeinschaftlich erarbeitet und können im Rahmen der geplanten Standardisierungsaktivitäten zu einer Erweiterung von FMI führen. Angegliederte Prozesse zur Modellentwicklung und -qualifizierung wurden angemessen berücksichtigt.
Dazu wurde das AP 2.2 in 3 Unterarbeitspakete aufgeteilt, in welchen sich jeweils mit den verschiedenen Aspekten der Kopplungen und Validierung von Simulationsmodellen beschäftigt wurde. Im Rahmen des ersten Unterarbeitspakets wurden die verschiedenen Mechanismen zur Kopplung von Modellen untersucht und bewertet, damit diese in den zugehörigen anderen Teilbereichen Anwendung finden konnten.
Im Rahmen des zweiten Unterarbeitspakets wurde sich mit generischen Ansätzen für Simulationszielen und Methoden beschäftigt, in denen es insbesondere um die Validierung und Kritikalitäts- und Wirksamkeitsanalysen ging. Dabei ging es auch um Kopplungseffekte zwischen Einzelmodellen und dem Gesamtsystem, welche sich auf die Leistungsfähigkeit von verschiedenen Modellzusammenstellungen auswirken können.

Im Rahmen des letzten Unterarbeitspakets wurden Gütekriterien für die Kopplung von Simulationsmodellen erarbeitet. Dabei wurden die erarbeiteten Kopplungsmechanismen verwendet. Als Grundlage zur Identifikation bestimmter Gütekriterien wurden die Beispiele aus SUC 1 und MS 1 verwendet, mit denen auch die Effekte von Modellkopplungen sichtbar gemacht wurden.

2.1.1.3 Definition von Anforderungen an die Ausführungsumgebungen

Für eine effiziente Abarbeitung der Simulationsaufgaben werden geeignete Ausführungsumgebungen im Sinne von Werkzeug- und Technologieplattformen benötigt. Dieses Arbeitspaket definierte Anforderungen an diese Ausführungsumgebungen, die zum einen aus den Ergebnissen von TP 1 abgeleitet wurden und sich zum anderen aus den anderen beiden Arbeitspaketen ergaben. Zusätzlich ergaben sich weitere Anforderungen aus der Nutzung der Simulationsinfrastruktur sowohl für die Validation von automatisierten Fahrfunktionen als auch für die kontinuierliche Entwicklung. Im Speziellen wurden Mechanismen für ein Änderungs- bzw. Konfigurationsmanagement benötigt, um sicherstellen zu können, dass alle benötigten Komponenten (Simulationsmodelle, Funktionssoftware, Simulationswerkzeuge etc.) eines bestimmten Entwicklung- oder Freigabestands, reproduzierbar aus verschiedenen Datenquellen abgerufen werden können.
Die Ausführungsumgebungen können aus verschiedenen Simulationsplattformen (MiL, SiL, HiL) oder auch aus Mischsystemen bestehen. Daraus leiten sich weitere Anforderungen ab, im Speziellen die Anforderungen zur Durchgängigkeit. Um die Performance bezüglich Geschwindigkeit der Ausführungsumgebung zu erhöhen, wird ein Mittel auch das Verteilen von Rechenaufgaben auf viele Rechner zur gleichen Zeit sein (Rechencluster). Hierdurch ergaben sich zusätzliche Anforderungen, die hier definiert wurden. Auch die Forderung, Simulationen schneller als Echtzeit durchführen zu wollen, führte zu weiteren Anforderungen.
Dazu gehörte eine Übersicht über die einzuhaltenden Standards und eine Bewertung derselben auf ihre Relevanz und Auswirkungen auf das Projekt SET Level. Ein Teilbereich davon beschäftigte sich mit Mindestanforderungen an Solver, welche im Projektkontext verwendet werden konnten.
Zusätzlich wurden Empfehlungen bezüglich Performance erarbeitet und Kriterien für Ziele und Güte von Simulationen geschaffen. Dadurch sollten nach Möglichkeit verschiedene Kriterien zur Beurteilung der verschiedenen Teilschritte einer Simulation in Hinblick auf deren Performance geschaffen werden. Diese haben dann in anderen Arbeitspaketen ihre Anwendung gefunden.
Außerdem sollten die Anforderungen an SiL, HiL und vernetzte Systeme untersucht werden.

2.1.2 Ergebnisbeiträge von TP 2

Im Wesentlichen sind in TP 2 vier Ergebnispakete entstanden.

2.1.2.1 Standards

Eine Übersicht über die verwendeten Standards mit Relevanz für SET Level folgt in Abschnitt 2.1.3.1

2.1.2.2 Anforderungen an Ausführungsumgebungen

Es wurden zu folgenden Teilgebieten der Ausführungsumgebungen Anforderungen formuliert:

- Solver in Abschnitt 2.1.3.2.1

- Performance in Abschnitt 2.1.3.2.2
- Simulationsumgebungen SiL, HiL und vernetzte Systeme in Abschnitt 2.1.3.2.3

2.1.2.3 Integrationsarchitektur

Es wurde eine grundlegende Architektur entwickelt, um auf der einen Seite eine spezifische Modellierung von Komponenten und auf der anderen Seite die Ausführung des Verbundes in einer Simulationsinfrastruktur zu ermöglichen. Die tatsächliche Beschreibung des Tests basiert auf den hier entwickelten Konzepten. Dazu kommen Schnittstellen zwischen verschiedenen Teilen der Simulationsarchitektur sowohl nach innen als auch nach außen und eine formalisierte Beschreibung der einzelnen Komponenten und deren Daten.
Teile dieser Integrationsarchitektur beinhalten folgende Themen

- Simulationsarchitektur als Übersicht in Abschnitt 2.1.3.3
- Spezifikationen Szenarienbeschreibungen in Abschnitt 2.1.3.3.1
- Formalisierte Beschreibung Testkonfiguration in Abschnitt 2.1.3.3.2
- Verwendung der ASAM OSI Schnittstelle in Abschnitt 2.1.3.3.3
- Zusammenhänge mit anderen OpenX Standards in Abschnitt 2.1.3.3.4

2.1.2.4 Integrationsmechanismen und Methoden

Integrationsmechanismen beinhalten folgende Teilaspekte, die im Projektkontext untersucht wurden:

- Entwicklung von Methoden in Abschnitt 2.1.3.4.1
- Tool Confidence Level in Abschnitt 2.1.3.4.2
- Modellintegration und -kopplung in Abschnitt 2.1.3.4.3

2.1.3 Detaillierte Ergebnisbeiträge von TP 2

2.1.3.1 Standards

Zunächst wurde eine Übersicht über bekannte und von den Projektpartnern verwendete Standards erstellt, welche anschließend bezüglich ihrer Relevanz und Berücksichtigung im Projektkontext bewertet wurden. Abbildung 27 zeigt einen frühen Arbeitsstand der gesammelten Standards in der gewählten Darstellung als Radardiagramm, um die aufgeführten Standards entsprechend der Relevanz für das Projekt SET Level und den Reifegrad der jeweiligen Standards übersichtlich darstellen zu können. Während der Projektlaufzeit wurden alle aufgelisteten Standards analysiert und in den jeweiligen Versionen bewertet.

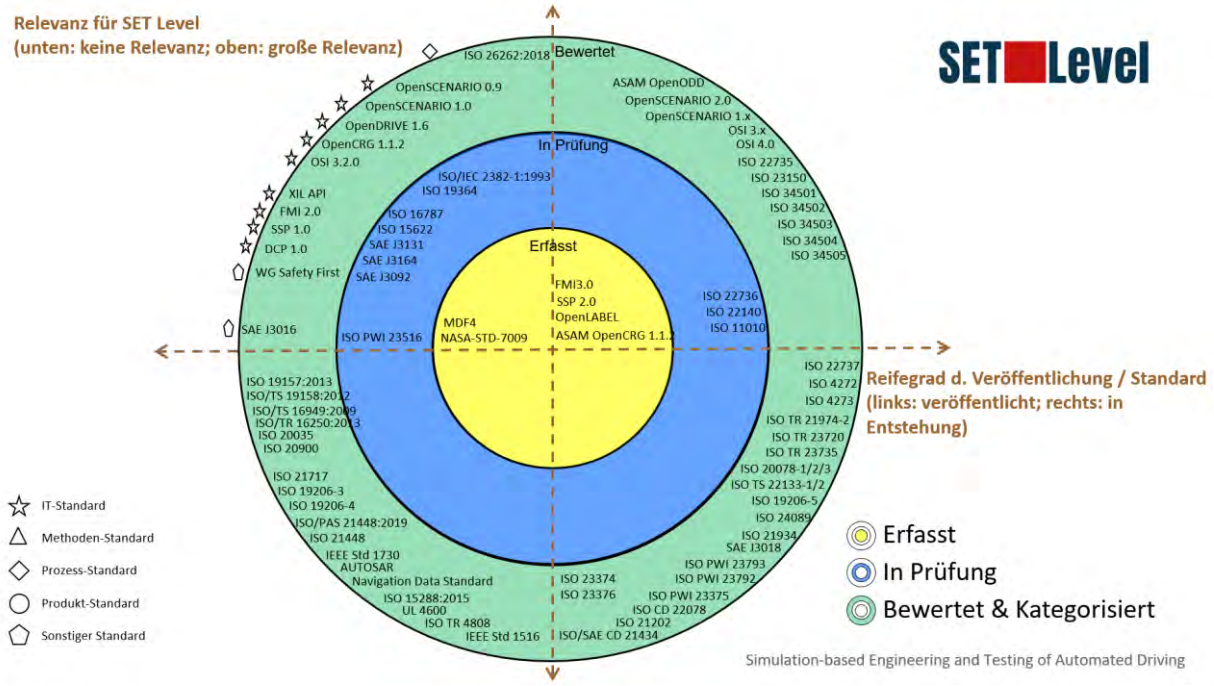


Abbildung 27: Früher Arbeitsstand der gesammelten Relevanz verschiedener Standards für das Projekt SET Level

Nachfolgend wurden die für das Projekt SET Level besonders relevanten Standards in verschiedenen Arbeitspaketen genauer untersucht. Insbesondere durch die Verwendung offener Standards wurde ein Mehrwert generiert, da eines der Hauptziele des Projekts SET Level eine Austauschbarkeit unterschiedlicher Aspekte und Architekturbereiche war. Durch die Verwendung von weit verbreiteten, offenen und häufig genutzten Standards erhöhte sich auch die Nutzbarkeit von und das Interesse an einer weiteren Verwendung der im Projektkontext erarbeiteten Ergebnisse. In Abbildung 28 sind die Standards dargestellt, die für das Projekt SET Level hauptsächlich von Interesse waren. Insbesondere in Standards wie ASAM OSI und ASAM OpenSCENARIO wurden benötigte Schnittstellen geschaffen und im Laufe des Projekts immer weiter verfeinert und erweitert, um verschiedene Aspekte der Simulation beschreiben und beeinflussen zu können sowie die Ausführung spezifischer Befehle zur Laufzeit zu überprüfen zu können. Zusätzlich wurden diese Standards durch die Projektarbeiten um Funktionalitäten erweitert. Einige projektspezifische Schnittstellen und Funktionalitäten, die derzeit nicht direkt in den allgemeinen Standard übernommen werden, stehen im öffentlichen Bereich des GitLabs des Projekts SET Level als Ergebnis zur Verfügung.

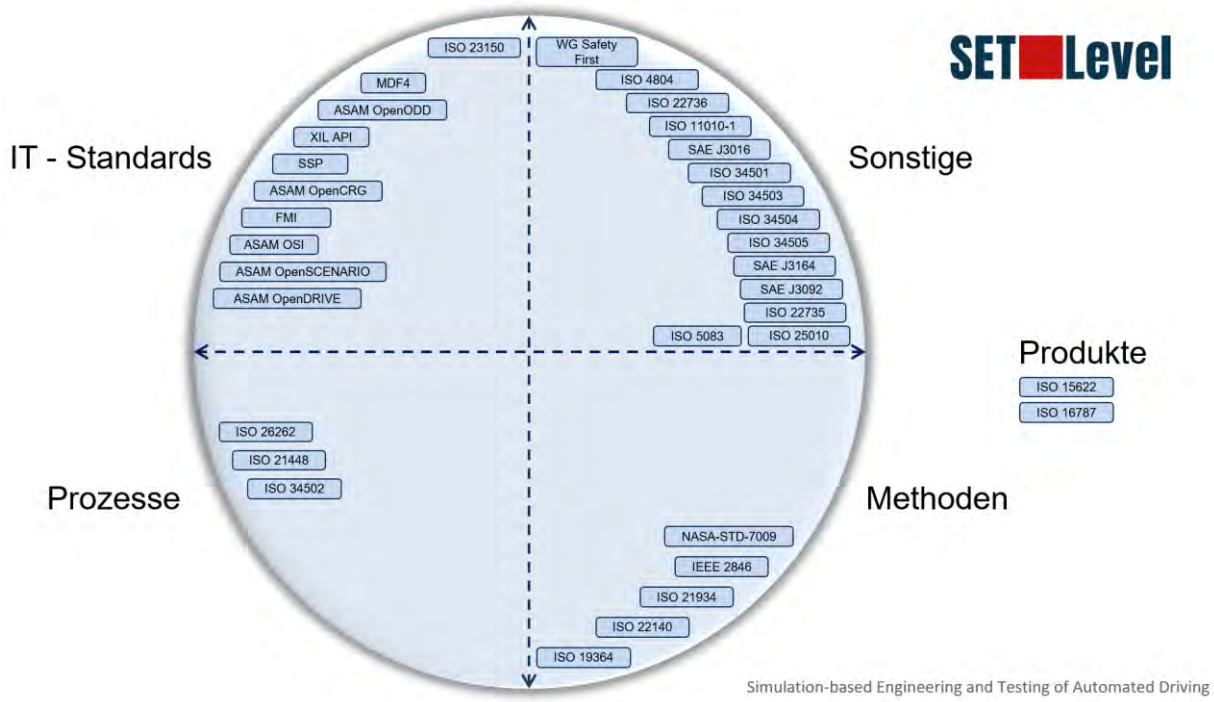


Abbildung 28: Übersicht der relevanten Standards im Projekt SET Level

2.1.3.2 Anforderungen an Ausführungsumgebungen

2.1.3.2.1 Teilgebiet: Solver

Die Anforderungen an Ausführungsumgebungen befassten sich unter anderem mit dem Thema „Solver“, welches sehr umfassend und komplex ist. Dies gilt schon für Regelungssysteme („Control-Systems“) in denen nur eine aktive Einheit (Prozess, Task) beteiligt ist. Dies gilt aber insbesondere bei parallel ausgeführten Modellen in Verbindung mit anderen aktiven Einheiten (Prozessen, Tasks, Threads) der Simulationsumgebung. Diese Ausgangssituation war praktisch in jeder Simulation im Bereich des Projekts SET Level vorhanden.
Im Projektverlauf wurde die Aufgabe analysiert und diskutiert, sowie welche Möglichkeiten bestehen, um sicherzustellen, dass die Simulationsaufgabe(n) mit der erforderlichen Genauigkeit in einer akzeptablen Zeit ausgeführt werden können.
Im Allgemeinen kann festgehalten werden, dass erforderliche Simulationen in ausreichender Zahl, ausreichende Güte und in einer akzeptablen Zeit durchgeführt bzw. vorhanden sein müssen, um die drei Simulationsziele „Analyse“, „Optimierung“ und „Test“ (für Freigaben, für die System-Abnahme oder auch mittelbar als Beitrag für die Homologation) bestmöglich erfüllen zu können.
Für die Zielerreichung ist es erforderlich, geeignete Solver auszuwählen und zu parametrieren. Hierbei kann keine pauschale Empfehlung ausgesprochen werden, da jeder eingesetzte Solver individuell optimiert werden muss. Die Optimierung erfolgt in einer hierarchischen Reihenfolge:

1. Verwendung des am besten für die Komponente geeigneten Solvers
2. Auswahl und Parametrierung der verschiedenen Solver bei einer Kopplung von Komponenten
3. Optimierung der gesamten Absicherungsstrategie

Der Begriff Solver wurde von fast jedem Beteiligten unterschiedlich interpretiert. Zu Projektbeginn waren hauptsächlich Kenntnisse im Bereich von Solvern für Regelungen (Differenzialgleichungen) vorhanden. Es wurde somit untersucht, wie diese Erfahrung auf den Aspekt der

Solver in einer Simulation anwendbar ist. Das erwies sich als schwierig. Die Trennung von Solvern für Regler physikalischer Systeme und Solvern für Simulationsmodelle stellte eine große Hürde dar. Diese gilt insbesondere bei Berücksichtigung der Simulations-Güte bei der Kopplung verschiedener Simulationsmodelle.
Nach (Wikipedia, 2023) wird ein Solver folgendermaßen definiert:
„Solver (engl., dt. Löser) ist eine Sammelbezeichnung für spezielle mathematische Computerprogramme, die mathematische Probleme numerisch lösen können."
In der Regel ist die Funktion des Solvers auf manuelle und automatische Entscheidungen verteilt. Man wählt einen geeigneten Solver, parametriert diesen und versieht ihn nach Möglichkeit mit einer automatischen Schrittweitensteuerung.

Durch Parametrisierung ist es auch möglich, den Simulations- und Lösungs-Raum geschickt einzuschränken. Wenn z. B. bekannt ist, dass 10 x 10^9 km erforderlich sind, um ein KI-Netzwerk ausreichend zu trainieren, dann werden eventuell nur 1000 „kritische" km zum Training des KI-Netzes benötigt. Die anderen 9.999.999.000 km werden hierbei vernachlässigt. Für jeden Bereich dieser 10 x 10^9 km wird die Schrittweite (Genauigkeit) so selektiert, dass unwichtige Kilometer nicht berücksichtigt werden, da sich dort die Lösung nicht befindet. Nur dort, wo die Lösung vermutet wird, wird die Genauigkeit so gewählt, dass die Lösung (Teil-Lösung) gefunden wird. Dieses Vorgehen fällt in den Bereich "Definition der notwendigen Genauigkeit" und nicht in den Bereich "Technische Umsetzung einer Genauigkeitsvorgabe" in der Simulation.

Der notwendige Solver im Projekt SET Level ist in der Regel ein manueller Auswahl-Prozess. In anderen Bereichen, z. B. model-predictive control auf ECUs, müssen während des Betriebs des Fahrzeuges und somit der ECU, ständig Minima, bzw. Maxima gefunden werden. Wie aus der Mathematik bekannt, ist dies kein triviales Problem. Wenn die Schrittweite falsch gewählt wurde, so werden im ungünstigsten Fall weder ein „Gipfel" (Maximum) noch ein „Tal" (Minimum) gefunden. Eine Abtastung des gesamten Lösungsraumes mit einer besonders geringen Schrittweite ist aus Performancegründen wiederum auch nicht möglich, um alle Gipfel oder Täler zu finden. Somit werden, meist problemspezifisch, intelligente und effiziente Solver benötigt, die mit hoher Wahrscheinlichkeit die Minima bzw. Maxima finden. Dieser Anwendungsfall (Optimierung) wurde nicht näher verfolgt. Zur Abgrenzung wurde und wird dieser immer wieder herangezogen werden.

Auch für das Projekt SET Level wäre es denkbar, dass Modelle in der Simulation die Auflösung (zeitlich, räumlich) selbst beeinflussen können. Das hat jedoch negative Auswirkungen auf Determinismus und die Eindeutigkeit der Simulationsergebnisse und erhöht die gesamte Komplexität. Deshalb erfolgt die Auswahl und Parametrierung des Solvers im Projekt SET Level manuell durch den Simulationsausführenden.

Diese manuelle Auswahl des geeigneten Solvers, oder der geeigneten Solver, erfolgt in der Regel gestützt durch Expertenwissen. In der Simulation laufen in der Regel mehrere Modelle parallel. Diese Modelle sind durch geeignete Mechanismen miteinander zu koppeln. Gerade diese Kopplungen haben einen großen Einfluss auf die Güte der Simulation. Deshalb müssen die Art der Kopplungen sowie die eingesetzten Solver näher untersucht werden.

Gerade bei Verwendung einer Co-Simulation werden mehrere unabhängige Modelle (oftmals FMU) gemeinsam simuliert. Hierbei setzen die verschiedenen Modelle jeweils eigene Solver ein. Eine globale Optimierung, z. B. durch die Verwendung des gleichen Solvers in verschiedenen Modellen, wird in der Regel einen großen Einfluss auf die Güte der Simulationsergebnisse haben.

Somit sind die Prozessschritte zur Definition des Solvers folgendermaßen definiert:

1. Die in einer ersten Analyse definierten Kriterien an die Simulationsziele (Erwartung, Güte, Ziele, etc.) für die Zusammenstellung der detaillierten (genaueren) Simulationen verwenden. Im Projekt SET Level wurde im SUC 1 das Konzept überprüft und validiert.
2. Die detaillierten (genaueren) Simulationen, unter Berücksichtigung der in der ersten Analyse definierten Kriterien, konfigurieren. Im Projekt SET Level sind das SUC 2 und SUC 3.
    a. „Konfigurieren“ meint hier verschiedene Dinge: Anzahl der Modelle, Umweltbedingungen (Ebene 5 des 6-Ebenen-Modells zur Beschreibung von Szenarien), Genauigkeit (Zeit, Raum), Aufzeichnungsdichte etc.
3. Geschwindigkeit und Trajektorien der Modelle einstellen
    a. Falls die Modelle selbst Einfluss auf die Geschwindigkeit und Trajektorien haben, sollte der mögliche Wertebereich für die Wahl eines geeigneten Solvers (Genauigkeit) berücksichtigt werden.
4. Experiment starten und beobachten
5. Ergebnisse auswerten
    a. Feststellen, ob die Genauigkeit des Solvers in einem sinnvollen (effizienten und effektiven) Bereich liegt und gegebenenfalls nachjustieren. Diese Verifikation kann nur erfolgen, wenn die Simulationsziele ausreichend genau definiert wurden. Oft wird dieser Schritt von einer Kosten-Nutzen-Abwägung geprägt sein. Das bedeutet dann auch, dass die Simulationsziele entsprechend geändert werden müssen. Die Auswirkungen auf das Gesamtziel, z. B. einer Freigabe durch positive Simulationsergebnisse, muss dann wiederum durch die Ausführung weiterer Simulationen verbessert werden.
6. Iteration zu Schritt 1. Unter Berücksichtigung von Schritt 5a.

Generell muss überlegt werden, ob es notwendig ist, dass jedes Modell Einfluss auf das Experiment („Closed-loop“) haben sollte. Dies würde im Umkehrschluss bedeuten, dass die aktuell vorgesehene manuelle Berechnung und Konfiguration des „Solvers“ berücksichtigt werden muss und dass sich Ort, Zeit, Geschwindigkeit während des Experiments „unvorhergesehen“ ändern können. Dies wäre dann ein rückgekoppeltes System.

Entscheidend beim Integrationsschritt (Solving) ist der numerische Fehler des rückgekoppelten Systems. Darüber hat ein einzelnes Teilmodell (z. B. eine FMU) keine Kenntnis. In der Regel werden Teilmodelle in sehr viele verschiedene Kontexte integriert, wobei innerhalb des einzelnen Modells keine Information über das jeweilige Gesamtmodell vorliegt. Somit gestaltet sich die Auswahl geeigneter Solver für rückgekoppelte Systeme deutlich komplexer. Eine Lösung hierzu ist die dynamische Anpassung des Solvers (Abtastrate, Zeitintervall, Genauigkeit). Ein solcher automatischer Solver ist nicht trivial, da in einem rückgekoppelten System mehrere Solver automatische Änderungen durchführen würden, was wiederum zu unvorhersehbarem, nicht deterministischem Verhalten führen kann.

Die für das Projekt SET Level wahrscheinlich beste Lösung der „Anforderungen an Solver“ ist, für jeden konkreten Simulation Use Cases die Genauigkeit des entsprechenden Solver manuell und statisch festzulegen.

Im Projekt SET Level werden die Anforderungen an Solver im SUC 2 (Gesamtfahrzeug) weniger genau aufgelöst als die Anforderungen an Solver im SUC 3 (Sensoren). Das entspricht

dem normalen Paradigma in der modellgetriebenen Softwareentwicklung oder SW-Integration: Je größer (hierarchisch höher) das System, desto weniger granular kann und wird aufgelöst (Solver).

Die Entwicklung der Sensormodelle in SUC 3 wurde betrieben, um diese nachfolgend auch im Kontext Gesamtfahrzeugmodell in SUC 2 verwenden zu können.

Automatische Konvertierungen von Modellen zwischen Hierarchieebenen gibt es aktuell und wahrscheinlich auf weite Sicht hin nicht. Somit erfordert es immer einen manuellen Eingriff eines Simulationsexperten. Das gleiche gilt für die Wahl und Parametrierung des hierfür benötigten Solvers.

Insbesondere bei der Kopplung von Modellen spielen Solver eine entscheidende Rolle. Daher wurde analysiert und diskutiert, welche Möglichkeiten bestehen, um sicherzustellen, dass die jeweilige Simulationsaufgabe mit der erforderlichen Genauigkeit in einer akzeptablen Zeit ausgeführt werden kann.
Hierfür können nach aktuellem Kenntnisstand und nach aktuellem Stand der Forschung in der Regel keine allgemeingültigen Berechnungen, Ableitungen und Formeln angegeben werden. Ganz im Gegenteil, es müssen aktuell immer individuelle geeignete Solver gefunden und parametrisiert werden.

2.1.3.2.2 Teilgebiet: Performance

Im Bereich der Simulation ist die Performance wohl der wichtigste Parameter, der zwischen Erfolg und Misserfolg unterscheidet. Aus diesem Grund fordert das SET Level Projekt, dass Empfehlungen gegeben werden müssen, die allen Beteiligten die Möglichkeit geben, die Simulationen effizient zu definieren, einzurichten und die Simulationen auf die performanteste Weise durchzuführen.

Das für die Optimierung der Leistung betrachtete System besteht aus Sicht der Hardware im Wesentlichen aus den Komponenten CPU(s), Arbeits-/Massenspeicher und Datenschnittstellen. Darüber hinaus gibt es Optimierungspotenziale im Bereich der Software. Es gibt mehrere Herausforderungen in Bezug auf die Leistungsoptimierung. Diese Herausforderungen ergeben sich oft nicht nur aus der Simulationsaufgabe selbst, sondern als Folgen unzureichender Erfahrung und einer im Allgemeinen zu unspezifizierten Teststrategie (Validierung und Verifikation) und somit einem weitestgehend unbekannten Simulationsziel.
Dazu wurden Im Rahmen des SET Level Projekts eine Checkliste mit 22 Punkten erstellt, deren Themen recht allgemein gehalten sind und in der Regel als „bekannt“ wahrgenommen werden. Es ist jedoch nicht einfach, all diese Themen gleichzeitig zu überblicken und sie anschließend auf das Design der Simulation, die Ausführung der Simulation und die Analyse der Simulationsergebnisse anzuwenden.
Dabei soll die Checkliste regelmäßig im Laufe der Planung, Design und Ausführung von Simulationen angewendet werden, um sicherzustellen, dass die Handlungsanweisungen möglichst angemessen berücksichtigt werden.
Eindeutige Leistungsanforderungen bedingen klar formulierte Zieldefinitionen. Letztere können z. B. über die Anwendung der bekannten SMART-Regel nach (Wikipedia, 2023) aufgestellt werden:
S(pezifisch):

- Was soll erreicht werden?
- Welche speziellen Schritte/Maßnahmen sollen ergriffen werden?

M(essbar):

- Definition eindeutiger Metriken zur Bewertung der Ergebnisse

A(kzeptiert):

- Handelt es sich um ein sinnvolles und motivierend?

R(ealistisch):

- Ist das Ziel unter realistischer Sichtweise erreichbar?

T(erminierbar):

- Existieren klare und zeitliche Vorgaben für die Durchführung?

Es kann festgehalten werden , dass das Ergebnis der Leistungsanforderungsanalyse in Form einer Checkliste gegeben ist, die es den verschiedenen Rollen, die an der Konzeption, Definition, Einrichtung, Implementierung, Ausführung und Bewertung der Simulationsläufe beteiligt sind, ermöglicht, die Performance systematisch zu verbessern.
Es ist möglich, einen Zeitbedarf für eine Simulation zu definieren. Falls das Simulationsziel nicht bekannt ist, kann die erforderliche Performance allerdings nicht ermittelt werden. Zum Beispiel: Es wird definiert, dass die Simulation höchstens eine Stunde benötigen darf. Falls nicht bekannt ist, wie viele Simulationen und mit welcher Genauigkeit diese Simulationen durchgeführt werden müssen, kann nicht berechnet werden wie leistungsfähig („performant“) die Simulation sein muss.
Tabelle 1 zum Thema „Performance“ kann in Simulationsprojekten regelmäßig und gewinnbringend als Arbeitsdokument zur Überprüfung der Performancequalität eingesetzt werden.

Tabelle 1: Checkliste Performance

Nr.	Cluster	Thema	Erklärung
1	Theorie	Conways Law	Die Organisation des Projekts diktiert die Architektur des Systems
2	Theorie	Amdahls Law	Es erlaubt die Berechnung der Erhöhung (oder Senkung) der Ausführungsgeschwindigkeit bei Parallelisierung
3	Architektur	SL-Architektur	Welchen Einfluss hat die SET Level-Architektur auf die Performance?
4	Architektur	Architektur von Modellen	Welchen Einfluss haben Modelle auf die Performance?
5	Architektur	Architektur der Interfaces von Modellen	Wie reagiert die Performance auf die Verdopplung (oder Erhöhung) der Anzahl gleicher Modelle? Z. B. OSI-Fußgänger
6	Architektur	Architektur der HADf	Welchen Einfluss hat die Architektur der HADf auf die Performance?
7	Interfaces	Interfaces	Wann ist es sinnvoll das System in Teilsysteme zu trennen? Wann lässt man das (Teil-)System monolithisch
8	Interfaces	ASAM OSI	Wann ist ASAM OSI geeignet?
9	Interfaces	OSI Flatbuf	Trade-off zwischen Speicherbedarf und Übertragungsgeschwindigkeit (optimiert für Einzel-Systeme)
10	Interfaces	OSI Protobuf	Trade-off zwischen Speicherbedarf und Übertragungsgeschwindigkeit (Google: optimiert für verteilte Systeme)

11	Zwänge	Zwänge und die Performance	Welche Zwänge haben einen negativen Einfluss auf die Performance?
12	Toolkette	Logging und Tracing	Oft bremst eine Datenaufzeichnung (tracing, logging) die Ausführungsgeschwindigkeit.
13	Toolkette	Durchgängigkeit von MiL nach SiL, HiL, vernetzte Systeme	Wie kann eine gute Durchgängigkeit (Wiederverwendbarkeit) zwischen Artefakten erreicht werden? Und sind Performance-Aussagen auf die alle XIL-Varianten übertragbar?
14	Toolkette	"Under the Hood"	Wie kann erreicht werden, dass Wissen (z. B. über Performance) für alle Mitarbeiter, z. B. Benutzer von MTP (Methoden, Prozessen, Tools) persistent ist. D.h. wie kann erreicht werden, dass nicht jeder "das Rad neu erfinden" muss.
15	Implementierung	Wie erreicht man "single system" Performance?	Die Performance ist dann am höchsten, wenn alles auf dem gleichen System / Prozessor läuft
16	Implementierung	Optimale Parallelisierung	Wie erreicht man optimale Parallelisierung, bezüglich Performance?
17	Implementierung	Parallelisierung von Szenarien	Wie kann bei einer Parallelisierung maximale "Unabhängigkeit" erreicht werden?
18	Implementierung	Höchste Auflösung (gleichbleibend in jeder Simulation)	Wie erreicht man, dass nur wenige Simulationen mit höchster Auflösung gerechnet werden müssen? Lösung: Iterativer Ansatz mit intelligenter Modifikation der Test-Strategie.
19	Implementierung	Variable Auflösung (innerhalb der Simulation)	Nur in kritischen Bereichen muss/soll hoch aufgelöst werden. Solver können „intelligente“ Verfahren sein, welche die Auflösung flexibel variieren können, ohne das Simulationsziel (Aufgabe) aus dem Fokus zu verlieren.
20	Implementierung	Unterschiedliche Takte	Verwendung unterschiedlicher Takte bei der Ausführung der verschiedenen Modelle erlaubt eine Performance-Steigerung.
21	Implementierung	Niedere Auflösung (rudimentäre Modelle)	Wie stellt man sicher, dass "ungenaue" Modelle nützliche Aussagen ermöglichen?
22	Implementierung	UDP / TCP (Reliability vs Performance)	Wie wichtig sind die Datenübertragungsprotokolle für die Performance? asynchron gegenüber synchron blockierend gegenüber nicht blockierend

2.1.3.2.3 Teilgebiet: Simulationsumgebungen SiL, HiL und vernetzte Systeme

In der Entwicklung durchlaufen die zu entwickelnden Artefakte, ob Hardware oder Software, verschiedene Fertigstellungsgrade. Je früher das Stadium in der Entwicklung desto weniger sind reale Artefakte vorhanden. Diese nicht vorhandenen realen Artefakte werden virtuell geschaffen. Somit ist auch klar, dass alle möglichen Kombinationen aus virtuellen und realen

Artefakten verwendet werden können und in vielen Projekten auch tatsächlich Verwendung finden.

Mit der Anforderung, sicher automatisiert zu fahren, ist es notwendig, eine Teststrategie zu entwickeln, um Tests auf verschiedene Testmethoden aufzuteilen („Multi Pillar Strategy“). Unter anderem wird es nötig sein einen großen Teil in Simulation durchzuführen. Daraus lassen sich verschiedenste Use Cases ableiten, die durch simulative Testmethoden realisiert werden sollen.

Um den Betrachtungsumfang von der Simulation weg auch auf HiL und vernetzte Systeme auszudehnen, wurde im Rahmen von SET Level eine Befragung der beteiligten OEM und Zulieferern durchgeführt. Das Ziel dieser Befragung auf der Basis von Fragebögen war es, die wichtigsten Anforderungen an Simulationsumgebungen im industriellen Einsatz zu beleuchten. Hierbei war es eine Herausforderung, eine möglichst breite Abdeckung unterschiedlicher Simulationsmethoden mit einzubeziehen. Das bedeutet, dass neben den rein virtuellen Simulationsmethoden (MiL, SiL) auch Methoden mit Hardwarebestandteilen (HiL) beachtet wurden.

Im Fokus der Befragung standen bereits etablierte Simulationsmethoden. Darüber hinaus wurden auch Methoden betrachtet, die das Potenzial besitzen, sich in den kommenden Jahren für den Test von Funktionen des hochautomatisierten Fahrens zu etablieren.

Ein sehr wichtiges Themengebiet ist die Durchgängigkeit und die Wiederverwendung. Das heißt, die Betrachtung wie gut und effizient die verschiedenen Simulations-/Testartefakte wie z. B. Umgebungsmodelle, Karten, Szenarienbeschreibungen, Testspezifikationen oder auch unterschiedliche Testumgebungen weitergegeben werden können. Also beispielsweise die Verwendung eines Fahrdynamikmodells eines Pkw mit identischer „Verpackung“ und Schnittstelle im MiL, SiL und HiL Test. Dieses Themengebiet (Durchgängigkeit und Wiederverwendung) konnte durch die Auswertung der Fragebögen nicht beantwortet werden.
Im Projekt SET Level wurden Simulationen hauptsächlich als MiL (Model in the Loop) aufgebaut. Die Fragebögen wurden deshalb vorrangig an Simulationsprojekte in der Industrie gegeben, welche SiL und HiL verwenden. Die dritte Ausprägung einer Simulation, die „vernetzten Systeme“, auch Mischsysteme genannt, besteht in der Regel aus einer Kombination von MiL-, SiL- oder HiL-Systemen. Diese Mischsysteme sind meist aus organisatorischen und/oder technischen Gründen und Gründen der Verfügbarkeit der Komponenten entstanden. Mischsysteme sind heterogen und somit schon intrinsisch komplexer als reine MiL-, SiL- oder HiL-Systeme. Bei näherer Betrachtung sind allerdings die meisten Simulationen Mischsysteme.
Die Ergebnisse einer durchgeführten Befragung im Projektzeitraum zeigten, dass das Feld der Simulation sehr viel größer ist als der im Projekt SET Level definierte Projektumfang. Insgesamt konnten folgende Ergebnisse durch die Auswertung formuliert werden:
Sehr plakativ und einfach formuliert, müssen schnell, differenziert und effizient „Aufgaben“ an die Simulation gestellt werden können, um eine verlässliche, belastbare und lehrreiche „Ergebnisse“ erhalten zu können. Dies wird durch flexibel und modular aufzubauende Simulationen erreicht.
Diese Aufgaben beziehen sich auf das SuT („System under Test“). Die Ergebnisse werden vom Simulations-System (Ausführungsumgebung) erzeugt.

Im Fragenbogen wurden drei Themen näher analysiert, die gleichzeitig auch drei der 8 Handlungsfelder entsprechen, die im Einleitungsteil des Schussberichtes bereits beschrieben wurden (siehe Abbildung 7).

- Simulation-based Engineering Tasks
- Standards
- Modelle

Eine andere Zusammenstellung der Handlungsfelder in den Fragebogen wäre möglich gewesen, wurde aber im Rahmen der Untersuchung für nicht notwendig gehalten.

In Abbildung 29 ist eine mögliche Teststrategie abgebildet, um eine automatisierte Fahrfunktion (Automated Driving Function, ADF) validieren zu können. Die Simulationstechnologie wird dazu auf verschiedenen Ebenen eingesetzt. Die Spanne reicht von Komponententests auf der Softwareebene bis hin zum Gesamtsystem innerhalb einer Verkehrssimulation.
Da die Simulationstechnologie in einem so großen Feld eingesetzt wird und zu jeder Zeit die Ergebnisse im Entwicklungsprozess eine spezifische Qualität zu erfüllen haben, müssen Anforderungen definiert werden, damit die verschiedenen Ausführungsumgebungen nutzbare Ergebnisse erzeugen. Dazu gehört, welche Nutzergruppen die Simulationstechnologie benutzen, welches Simulationsziel verfolgt wird, welche Simulationselemente eingesetzt werden und welche Standards anzuwenden sind.
Im Projekt SET Level konnten nur punktuelle, repräsentative Schwerpunkte bearbeitet werden. Dazu wurde die Gesamtmethodik an spezifischen Simulation Use Cases (SUCs) demonstriert. Mit dem Ziel der Überprüfung, ob die angewandten SUCs alle notwendigen Anforderungen abdecken, wurde ein Fragebogen entwickelt. Dieser sollte zeigen, ob zusätzliche Anforderungen an die jeweiligen Simulationsaufgaben notwendig sind. Die Ergebnisse können für zukünftige Projekte und Forschung herangezogen werden.

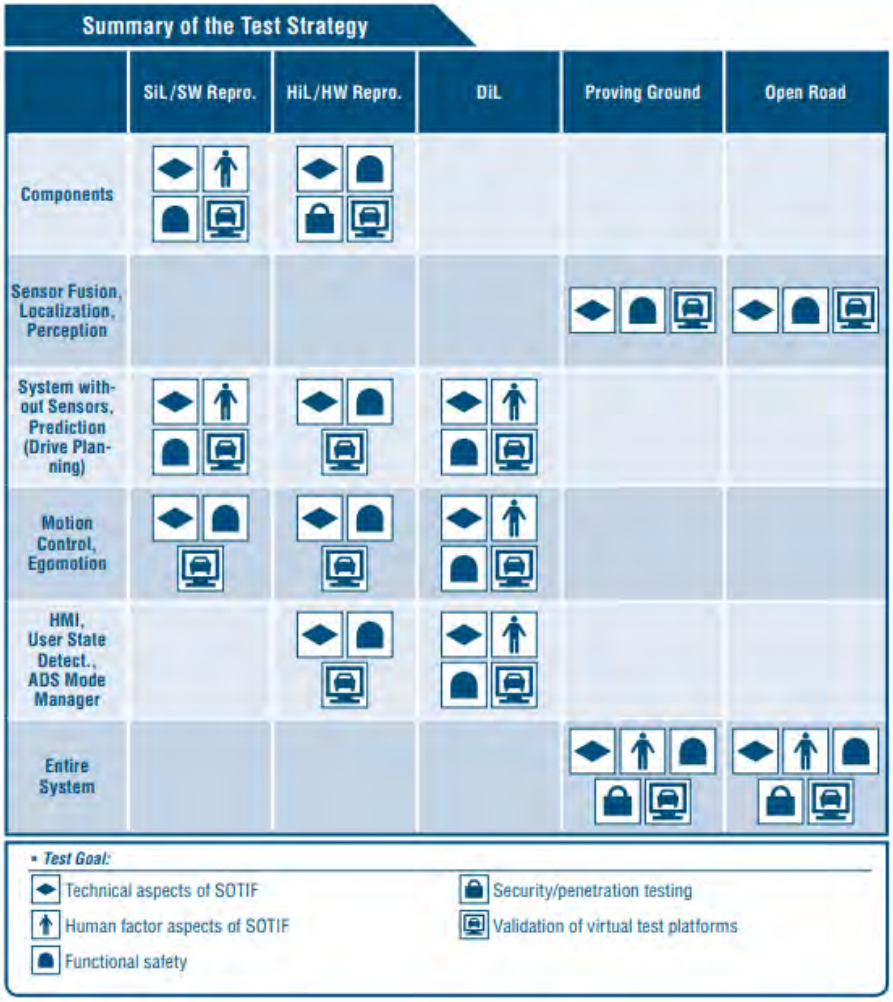


Abbildung 29: Zusammenfassung einer Teststrategie (SaFAD, 2019)

Insbesondere der Bereich „Durchgängigkeit und Wiederverwendbarkeit“ hat sich bei der Auswertung der Fragebogen als zentrales Thema herausgestellt. Das Thema Durchgängigkeit und Wiederverwendung meint hierbei die Betrachtung, wie gut und effizient die verschiedenen Simulations-/Testartefakte wie z. B. Umgebungsmodelle, Karten, Szenarienbeschreibungen, Testspezifikationen oder auch Tools von Testumgebung zu Testumgebung weitergegeben werden können. Also beispielsweise die Verwendung eines Fahrdynamikmodells mit identischer „Verpackung“ und Schnittstelle im MiL-, SiL- und HiL-Test.
Aspekte zum Thema Durchgängigkeit und Wiederverwendung sind nicht direkt in den Fragen des Fragebogens adressiert worden. Es folgt daher eine Betrachtung aus eigenen Erfahrungen heraus.
In einem Whitepaper zum Thema ViL (IPG, 2019) wird Folgendes propagiert: „Die VIL-Methode ermöglicht dabei eine sehr zuverlässige Durchführung der Tests auf einem hohen Integrationsgrad. Die Entwicklung kann von allen Vorteilen der virtuellen Welt profitieren: der Reproduzierbarkeit in dynamischen Szenarien, einem reduzierten Testaufwand, der **Wiederverwendbarkeit** von Testfällen und einer automatisierten Testauswertung. Damit alle Funktionen und die Performanz im realen Fahrzeug in den gleichen Szenarien und Umgebungen getestet werden können wie bei den anderen XIL-Methoden, hat die **Durchgängigkeit** der eingesetzten Simulationsumgebung höchste Bedeutung.“
Abbildung 30 zeigt exemplarisch, welche Artefakte der „virtuellen Welt“ in den verschiedenen Testumgebungen eine Rolle spielen. Da ist es naheliegend, die Artefakte identisch

wiederzuverwenden, unabhängig davon, in welcher Testumgebung simuliert wird. Ideal wäre somit, beispielsweise das identische Fahrdynamikmodell für MiL, SiL und HiL zu verwenden.

V Virtuelle Welt R Reale Welt	MIL	SIL	ECU HIL	System HIL	PT HIL/ VIL stat.	VIL dynamisch	Reale Welt
ECU-Code / -Funktion	V	R	R	R	R	R	R
ECU	V	V	R	R	R	R	R
System (z.B. Lenkgetriebe)	V	V	V	R	R	R	R
Fahrzeug	V	V	V	V	R	R	R
Straße und statische Umgebung	V	V	V	V	V R	R	R
Fahrdynamik	V	V	V	V	V	R	R
Systemerfahrung	V	V	V	V	V	R	R
Fahrer	V	V	V	V	V	V R	R
Umgebungsverkehr	V	V	V	V	V	V	R

Abbildung 30: Artefakte in den verschiedenen Testumgebungen (IPG, 2019)

Auf die zentralen Fragen

- Warum ist diese naheliegende Idee trotzdem eine Herausforderung in der Umsetzung?
- Was kann getan werden, um diese Idee umsetzbar(er) zu machen?

wird in den nächsten Abschnitten näher eingegangen.

Vielfältigkeit des SuT

Test- und Simulationsumgebungen müssen so flexibel sein, dass idealerweise jede Art von SuT angekoppelt werden kann: Modell, ECU-Code, ECU-HW, ECU-HW-Verbund, Gesamtfahrzeug, Verkehrssystem etc. Dabei können die unterschiedlichsten Schnittstellen gefordert sein: OTA (over-the-air) Sensortest, Feldbusse (CAN, LIN …), Analoge I/O, Digitale I/O, Serviceschnittstellen, SW-Schnittstellen etc. Für die Ankopplung der SuTs existieren aktuell oft jeweils verschiedene Adaptertechnologien, welche meist spezifisch für die Testumgebung (die auf das SuT zugeschnitten ist) sind. Diese lassen sich daher oft nicht für weitere Entwicklungsstufen wiederverwenden. Mit Hilfe von Standards sollen diese Inkompatibilitäten in Zukunft behoben werden.
Allerdings ist eine allgemeine vereinheitlichte Integrationsschnittstelle eines SuTs aus Gründen der Vielzahl unterschiedlicher Simulationsaufgaben und korrespondierender SuTs nicht möglich. Aus praktischer und ökonomischer Sicht muss somit entschieden werden, ob alle Testplattformen in eine globale Simulationsumgebung integriert werden können. Hier gilt es zielgerichtete Lösungen zu finden, die je nach Test Ziel variieren können.

Unterschiedliche Anforderungen an Testumgebungen

Hier ist vor allem das Echtzeitverhalten der Simulationsmodelle zu nennen. Für MiL und SiL wünscht man sich Simulationsmodelle, die mehrfach schneller als Echtzeit rechnen können, für Anwendungen mit realen SuTs, wie sie bei einem HiL-Prüfstand vorhanden sind, werden hingegen Simulationsmodelle benötigt, die in Echtzeit ablaufen. Durch diesen Zielkonflikt kann eine Wiederverwendung von Simulationsmodellen eingeschränkt sein.

Ähnliches gilt auch für Szenarienbeschreibungen. Simulationstools müssen in der Lage sein, eine Szenario-Engine sowohl mehrfach schneller als Echtzeit rechnen als auch in Echtzeit ablaufen lassen zu können.
Technologische Unterschiede in der verwendeten Simulationshardware und des verwendeten Betriebssystems der verschiedenen Testumgebungen können die Durchgängigkeit und Wiederverwendung von Modellen erschweren. Für MiL und SiL kommen typischerweise Windows oder LINUX Betriebssysteme zum Einsatz, für HiL echtzeitfähige Betriebssysteme. Sind Modelle nicht für andere Betriebssysteme kompilierbar beziehungsweise vorbereitet, scheitert eine Wiederverwendung und es müssen andere Modelle aus anderen Quellen verwendet werden.

Unterschiedliche Testmethoden

Für eine Gesamtstrategie (Testmatrix) ergibt sich ein etwas komplexeres Bild. Neben den verschiedenen Testumgebungen (Test-Environment), die sich vor allem durch die Art des SuTs ausprägen (Modell, ECU-Code, ECU-HW, ECU-HW-Verbund, Gesamtfahrzeug etc.) existieren auch verschiedene Testmethoden wie anforderungsbasierter Test, Schnittstellentest, Fehlereinbringungstest, Performance Test und szenarienbasierter Test. Der Aspekt „Testen für KI (Künstliche Intelligenz)“ ist hier nicht berücksichtigt.
Abbildung 31 (angelehnt an die ASAM Test Specification Study Group Test Matrix) listet die beiden Dimensionen Testumgebung und Testmethode mit Beispielen und einer Einordnung der Fragebögen auf.

Example of a Testmatrix for Functional Safety		Test Environment								
		MIL	SIL Software Reprocessing/ Data Replay	SIL	Hardware Reprocessing/ Data Replay	HIL	Vehicle-in-the-Loop	Driver-in-the-Loop	Proving Ground	Open Road Field Monitoring
Test Method	**Requirements-based Test** (Functional Test) *Software architectural design / Specified functionality*	**#9 System Architektur als Simulationsarchitektur überprüfen**	**#3 SW Integration Test ECU-Code Fahrfunktion with Restbus data**	**#2 Runtime-Test ECU-Code Fahrfunktion #11 Kontinuierliche Überprüfung des ECU SW Standes**		**#8 Test ECU Verbund (E/E Gesamtverbund)**	e.g. over-the-air sensor stimulation		e.g. testing complete AD function under real conditions	e.g. Shadowing
	Interface Test *Software unit implementation/ Hardware-software interface specification*			e.g. Software Integration Tests	e.g. sensor perception tests	e.g. higher level integration tests (e.g. bus communication)	e.g. test complete AD effect chain on system level			
	Fault injection Testing of safety mechanism/ Robustness	e.g. checking failure effects on controllers	e.g. robustness against camera image pixel faults	e.g. verification of safety mechanisms	e.g. testing safety mechanism robustness incl. HW	e.g. testing of safety mechanisms	e.g. testing of system-level safety	e.g. testing of controllability of driver takeover	e.g. testing of system-level safety	
	Resource Usage/ Performance Test *Sufficiency of resources/ Hardware architectural design*					**#10 Performance Assage über Systemzustand ECU-Verbund**				
	Scenario-based Test *Validation of real-life use cases/SOTIF validation*	**#6 Verifikation der Fahrfunktion**		**#1 Freigabe Fahrfunktion #7 Integration AD Verbund SW (Kontinuierliche Integration und Front-Loading)**		e.g. Validation of electronic integration	e.g. Validaton on system level	e.g. Vehicle interaction with driver	e.g. testing complete AD function under scenario-defined conditions	e.g. test drives with the goal to catch specific scenarios

#4 Generierung realistisches Radarbild mit Hilfe von Sensormodellen	Hier handelt es sich nicht direkt um einen Test, sondern die Erzeugung von hochwertigen Daten für einen späteren Test, z.B. für das SW oder HW Reprocessing.
#5 Generierung synthetischer Radar Sensordaten mit Hilfe von Sensormodellen	

Abbildung 31: Testmatrix (in Anlehnung an die ASAM Test Specification Study Group)

Es kann nicht davon ausgegangen werden, dass zwischen allen Zellen der Testmatrix (eine Zelle als eine Kombination aus Testumgebung und Testmethode) eine Durchgängigkeit und Wiederverwendung von Daten, Modellen oder Tools möglich bzw. sinnvoll ist.
Die hier genannten Testumgebungen SW/HW Reprocessing (Data Replay) nehmen z. B. eine Sonderstellung ein. Bei diesen Testumgebungen werden keine Szenarienbeschreibungen und i.d.R. auch keine Umgebungsmodelle verwendet. Allerdings können die Replay Daten aus dem SW Reprocessing für das HW Reprocessing wiederverwendet werden, wobei diese Daten beispielsweise aus Aufzeichnungen realer Messfahrten gewonnen wurden.

Modellgüte

Je nachdem, welche Sicherheit (absence of unreasonable risk) gefordert ist, wird eine bestimmte Güte beziehungsweise Genauigkeit an die Test-/Simulationsergebnisse gefordert. Somit muss mit unterschiedlichem Anspruch an die SW Tools (Tool Qualification Level) aber auch Anspruch an den Verifikations- und Validierungsgrad der Modellabsicherung (z. B. NASA Technology Readiness Level (TRL)) gearbeitet werden. Pauschal kann dies nicht an der Testumgebung oder der Testmethode festgemacht werden. Entscheidend ist die konkrete Phase im Entwicklungsprozess, welcher meistens im V-Modell abgebildet ist. Entwicklungsbegleitende Tests sind eher im unteren Bereich, Freigabetests im oberen Bereich des V-Modells verankert. Falls eine mögliche Verwendung in verschiedenen Testumgebungen angestrebt ist, haben Modelle mit geringer Absicherung potenziell weniger Wiederverwendungswert als Modelle mit sehr hoher Absicherung (V&V). Falls sich die Wiederverwendbarkeit über verschiedene Funktionsevolutionen verteilt, ist eventuell die geringe Absicherung und damit die schnellere Anpassung ein großer Vorteil in frühen Entwicklungsphasen.

Szenarien

Für den szenarienbasierten Test ist eine sehr große Menge an Szenarien erforderlich, um dem „offenen Kontext“ des realen Verkehrs gerecht zu werden. Für diese Szenarien entstehen momentan Ansätze für Szenariodatenbanken mit dem Ziel der unternehmensübergreifenden Sammlung und Verwendung von Szenarien zu ermöglichen. Es wird davon ausgegangen, dass ein einzelnes Unternehmen nicht in der Lage sein wird, eine ausreichend große Sammlung an Szenarien zu erstellen. Dies könnte allein schon aus betriebswirtschaftlicher Sicht für ein Einzelunternehmen nicht realistisch darstellbar sein.
Für den Aufbau und die Wiederverwendung der Szenarien sind folgende Anforderungen sinnvoll:

- Darstellungsfähigkeit in einem standardisierten Format (z. B. ASAM OpenSCENARIO)
- Kennzeichnung der Szenarien (z. B. ASAM OpenLABEL), die eine automatisierte Suche ermöglicht
- Zugänglichkeit über Unternehmensgrenzen hinweg (Lizenzrechte, Zugriffsrechte, etc.)
- Trennung der Szenarien von testrelevanten Aspekten (z. B. Bewertungsmetriken, Akzeptanzkriterien) zur Wiederverwendung in beliebigen Testfällen.

Testspezifikation

Sind die Testfälle Keyword-/Step-based abstrakt umgesetzt (Trennung von abstrakten Steps und der dahinterliegenden Step-Implementierung), ist eine Wiederverwendung von Steps für verschiedene Testumgebungen und Testmethoden erleichtert.
Besteht eine klare Trennung zwischen Testfall und Szenario, ergibt sich eine gute Möglichkeit der Wiederverwendbarkeit von Testfällen.

Schnittstelle zwischen Testfall und Szenario

Während der Simulation oder des Tests kommt es zu Interaktionen zwischen dem Testfall und der Szenarioausführung in der Szenario-Engine. Zum einen muss es eine konsistente Konfiguration von Parametern geben (z. B. Anfangspose der Agenten), zum anderen müssen Ereignisse im Szenario (z. B. Ausscheren erkannt) von der Simulation verarbeitet werden können.
Die ASAM Test Specification Study Group sieht hier eine bidirektionale Schnittstelle zwischen Szenario und Testfall, die analysiert, spezifiziert und idealerweise standardisiert wird. Dieser

Schnittstellenstandard dient als Bindeglied und wäre unabhängig von einem Standard zur Beschreibung von Szenarien- oder Testfällen zu definieren und zu verwenden.

Standards

Die Nutzung von Standards unterstützt die Wiederverwendung von Modellen, Karten, Szenarien und Testspezifikationen in hohem Maße. Hier sind insbesondere folgende Standards hervorzuheben, für die auch das Projekt SET Level Zuarbeit geleistet hat:

- ASAM OpenDRIVE
- ASAM OpenSCENARIO
- ASAM Open Simulation Interface (ASAM OSI)
- Functional Mock-up Interface FMI

Der FMI Standard (Version 2.0.2) spielt eine wichtige Rolle bei der Modellweitergabe. FMI 2.0 hat allerdings eine Reihe von Limitierungen, die in derzeitigen Implementierungen nur durch Behelfslösungen beherrscht werden (z. B. Pointer auf arrays bei der Nutzung von ASAM OSI). Die Version FMI 3.0, die kürzlich veröffentlicht wurde, hat hierfür einige wichtige Erweiterungen mitgebracht. Um virtuelle Steuergeräte abbilden zu können, dient FMI 3.0 für künftige Projekte als Basis. Für die Standardisierung von Schnittstellen für den SiL Test existiert die VDA Projektgruppe „Software-in-the-Loop (SiL) Standardisierung“. In diesem Projekt werden wesentliche Schnittstellen betrachtet, die im Zusammenspiel einer Toolkette für den SiL-Test erforderlich sind. Im Zentrum stehen die Aspekte SiL Architektur, SiL Framework, Simulationsartefakte (V-ECU), Virtuelle E/E (Abbildung der Fahrzeugelektronik in rein virtuellen Simulations- und Testumgebungen), Test Automation und MCD (Measurement, Calibration & Diagnostics). Zudem wird der Begriff und eine API für ein „SiL Virtual Interface“ (SiLVI) vorgeschlagen. Dadurch soll eine standardisierte virtuelle Bus Kommunikation (CAN, LIN, …) zwischen V-ECUs, unter Berücksichtigung von z. B. Bandbreiten und Fehlerfällen, ermöglicht werden.

Es folgt eine Liste von Handlungsempfehlungen, die als Ergebnis der Befragung mittels Fragebögen durch eine Arbeitsgruppe im Projekt SET Level formuliert wurden:

1) Ohne übergreifende Koordination, Planung und Organisation kommt es zu unterschiedlichen Lösungen in den unterschiedlichen Teams, die sich u.U. nicht einmal untereinander kennen.
2) Ein SuT oder die SuT-Schnittstellen können und sollen auf Standards basieren. Allerdings ist eine allgemeine, allumfassende Standardisierung eines SuT aus Gründen der Effizienz nicht möglich.
3) Sind Modelle nicht für andere Betriebssysteme kompilierbar oder vorbereitet reduziert es die Möglichkeit der Wiederverwendung oder es erhöht den Aufwand für eine Verwendung und es müssen Modelle aus anderen Quellen verwendet werden.
4) Die hohe Anzahl unterschiedlicher Testmethoden, die eine Gesamtstrategie ausmachen (Testmatrix), ergeben ein sehr komplexes Bild, welches gut durchdacht, analysiert und strukturiert sein sollte.
5) Es sollte angestrebt werden, Szenariendatenbanken mit dem Ziel einer unternehmensübergreifenden Sammlung und Verwendung von Szenarienbeschreibungen zu ermöglichen.
6) Für den Aufbau und die Wiederverwendung von Szenarien sind folgende Anforderungen sinnvoll: Standard-Formate, Metadaten, Verfügbarkeit und Modularität.
7) Wurde bei der Umsetzung der Testspezifikation in Step-based Testfälle, d.h. der Trennung von abstrakten Steps und der dahinterliegenden Step-Implementierung

unterschieden, so ist eine Wiederverwendung von Steps für verschiedene Testumgebungen und Testmethoden erleichtert.
8) Besteht eine klare Trennung zwischen Testfall und Szenarienbeschreibung ergibt das eine gute Wiederverwendbarkeit.
9) Die ASAM Test Specification Study Group empfiehlt eine bidirektionale Schnittstelle zwischen Szenario und Testfall, die analysiert, spezifiziert und idealerweise standardisiert wird.
10) Die Nutzung von Standards unterstützt die Wiederverwendung von Modellen, Karten, Szenarien und Test-Spezifikationen in hohem Maße.
11) Aktuell fehlt ein Standard für Weitergabe von virtuellen ECUs, inklusive deren Aspekte, Anforderungen und Definitionen. Dieser sollte möglichst bald definiert und verwendet werden.
12) Bei den Befragten existiert der Bedarf, aus einer komplett virtuellen Umgebung heraus eine Gesamtsimulation durchzuführen, die eine Aussage über die Korrektheit der Software und der Architektur zulässt.
13) Bei den Befragten existiert der Bedarf, aus virtuellen Umgebungen heraus Sensoreingangsdaten zu erzeugen, welche für Sensormodelle verwendet werden können, um die Erkennungsgüte von Umgebungselementen zu überprüfen.
14) Einen darüberhinausgehenden Beitrag zur Freigabe eines Systems würde ein Gesamt-Absicherungskonzept erfordern, in dem der Anteil der durch Simulation abgedeckten Fragestellungen im Gesamtzusammenhang klar beschrieben ist. Diese ist aktuell noch nicht in Sicht, insbesondere, da die theoretische Grundlage dafür fehlt.
15) Es ist erforderlich, dass eine SiL-basierte Simulation effizient und modular aufgebaut wird, dass die verwendeten Begrifflichkeiten von allen Rollen gleich verstanden werden und dass die Ergebnisse strukturiert, transparent und leicht nachvollziehbar dokumentiert werden.
16) Es ist erforderlich, dass eine HiL-basierte Simulation effizient und modular aufgebaut wird, dass die verwendeten Begrifflichkeiten von allen anderen Rollen gleich verstanden werden und dass die Ergebnisse strukturiert, transparent und leicht nachvollziehbar dokumentiert werden.
17) Es ist zu erwarten, dass mehr und mehr „reale“ Daten und „reale“ Komponenten in Simulationen für das autonome Fahren verwendet werden. Damit werden verteilte Systeme in Zukunft vermehrt eingesetzt werden.
18) Am häufigsten erwähnt wurde der Standard *Functional Mockup Interface* (FMI). Dieser Standard findet bei fast allen Elementen der Testumgebungen seinen Einsatz und zeigt, wie vielfältig der Standard einsetzbar ist.

Noch ist es für jedes einzelne Simulationsprojekt eine Herausforderung, die beste Simulationstechnologie auszuwählen. Die Handlungsempfehlungen sollen dem Simulationsarchitekten, -entwickler und -verantwortlichen eine Basis geben, um eine leichtere Entscheidung bezüglich einer sinnvollen Eingrenzung der großen Anzahl von Freiheitsgraden einer Simulation treffen zu können.
Aus der durchgeführten Befragung im Projektkontext wurde folgendes abgeleitet:
1) Eine große Bandbreite der Simulationsziele:
    a. Sensorsimulation
    b. Verifikation der Systemarchitektur
    c. Gesamtfreigabe
2) Verwendung einer Vielzahl von Standards
    a. Der FMI Standard war stark vertreten

   b. Der ASAM OSI Standard war nicht stark vertreten
   c. Der SSP Standard war nicht stark vertreten
3) Insbesondere aus der seltenen Verwendung übergreifender Standards wie SSP kann geschlossen werden, dass es noch einige Jahre dauern wird, bis allgemeine Austauschbarkeit unter Simulationsartefakten möglich ist.
4) Es gibt eine große Anzahl verschiedener Tools, verschiedener Modelle, verschiedener Standards, sodass aktuell praktisch immer individuelle Lösungen aufgebaut werden. Dies unterstreicht die Bedeutung des Projekts SET Level: Weg von „home grown“-Lösungen und hin zu standardisierten Simulationsaufbauten. Die Antworten der Industrie haben gezeigt, dass Standardisierung aktuell zu den wichtigsten Themen gehört.
5) Da in vielen Simulationen Sensormodelle mit unterschiedlichen Schnittstellen verwendet wurden, ist eine Standard API wünschenswert, welche die Anforderungen einer Kopplung von Sensoren an Funktionsmodelle erfüllt.
6) Insbesondere für eine gute Durchgängigkeit muss die Frage beantwortet werden, wie Sensordaten im realen Fahrzeug ausgetauscht werden können.

2.1.3.3 Simulations- und Integrationsarchitektur

Die grundlegende Architektur wurde entwickelt, um eine einheitliche Integration spezifischer Komponenten zu ermöglichen. Außerdem sollte damit eine Beschreibung für die Ausführung der Komponenten in einer Simulationsinfrastruktur geschaffen werden. Dazu wurden Konzepte entwickelt, welche alle Teilbereiche und Komponenten identifizieren und entsprechend ihren Aufgaben zuordnen können.
Die Simulationsstruktur wurde in Enterprise Architekt abgebildet und alle Module mit den entsprechend benötigten Informationen und Schnittstellen spezifiziert, wie in Abbildung 32 zu sehen ist. Alle Komponenten sollten austauschbar sein und mussten daher entsprechend dokumentiert werden.
Außerdem wurden ein PoC für Testumgebung und -spezifikation erarbeitet, auf dem automatisierte Testbeschreibungen, -durchführung und -auswertung entwickelt und getestet werden konnten. Darauf aufbauend wurde zusätzlich eine weitergehende Definition eines Testspezifikationsformates für die Beschreibung von Test Tools entwickelt. Hinzu kommt die formalisierte Beschreibung eines Test Setup Containers, um die Austauschbarkeit und Reproduktion solcher Tests zu ermöglichen und formalisieren zu können.

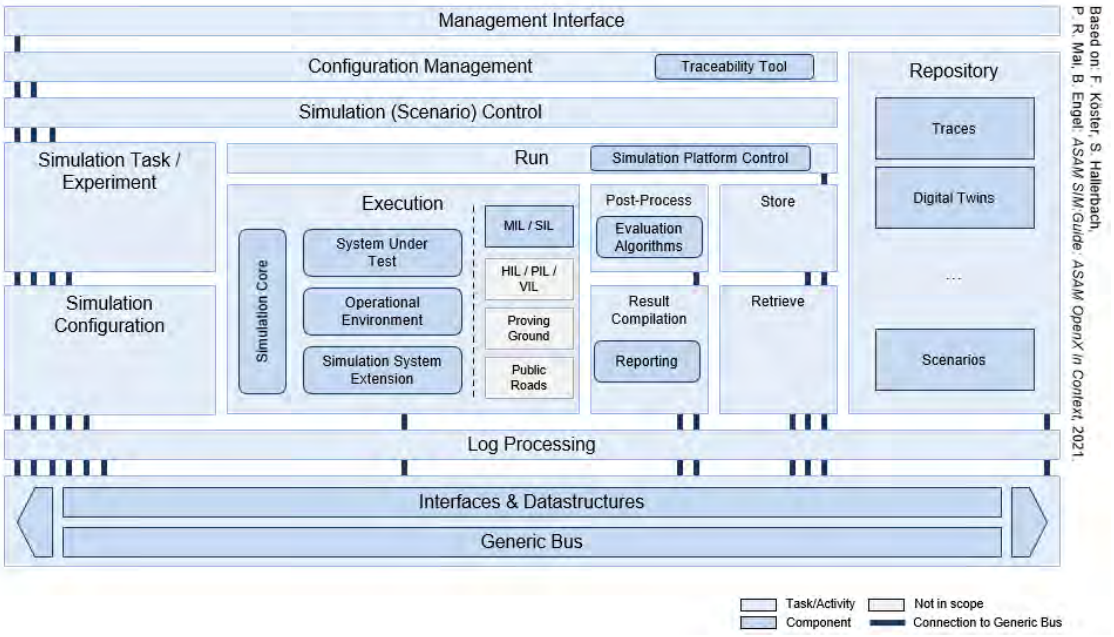


Abbildung 32: Übersicht der Simulationsarchitektur

2.1.3.3.1 Spezifikationen der Szenarienbeschreibungen

Im SET Level Projekt wurden Spezifikationen für die vereinheitlichte Nutzung von Szenarienbeschreibungen, genauer die Einbindung des szenarienbasierten Testens und Analysierens in bestehende Prozesse der Automobilindustrie, evaluiert und entsprechende Integrationskonzepte erarbeitet (siehe Abbildung 33 und Abbildung 34).

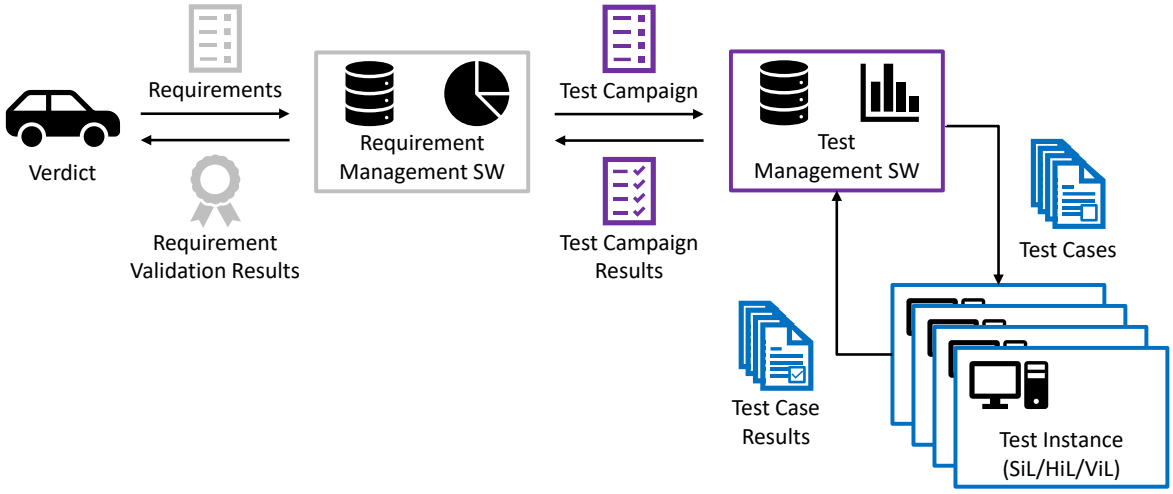


Abbildung 33: Anforderungsbasiertes Testen in der Automobilindustrie

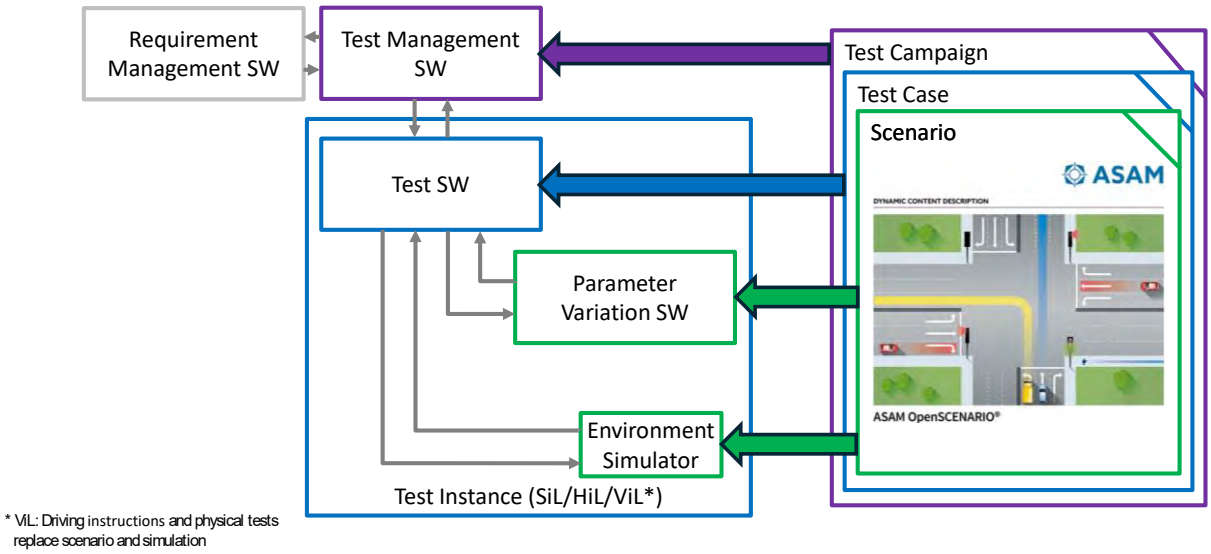


Abbildung 34: Szenarienbasiertes Testen im Kontext anforderungsbasiertes Testen[6]

Hierbei wurde von den Projektpartnern ein Konzept zur Implementierung von durchgängig (über die Automobilindustrie und alle zugehörigen Testinstanzen hinweg) nutzbaren Spezifikationen für Test- und Analyseaufgaben entwickelt. Der Ansatz fußt auf einer klaren Separation von Szenarien- und Testbeschreibungen (siehe Abbildung 35).

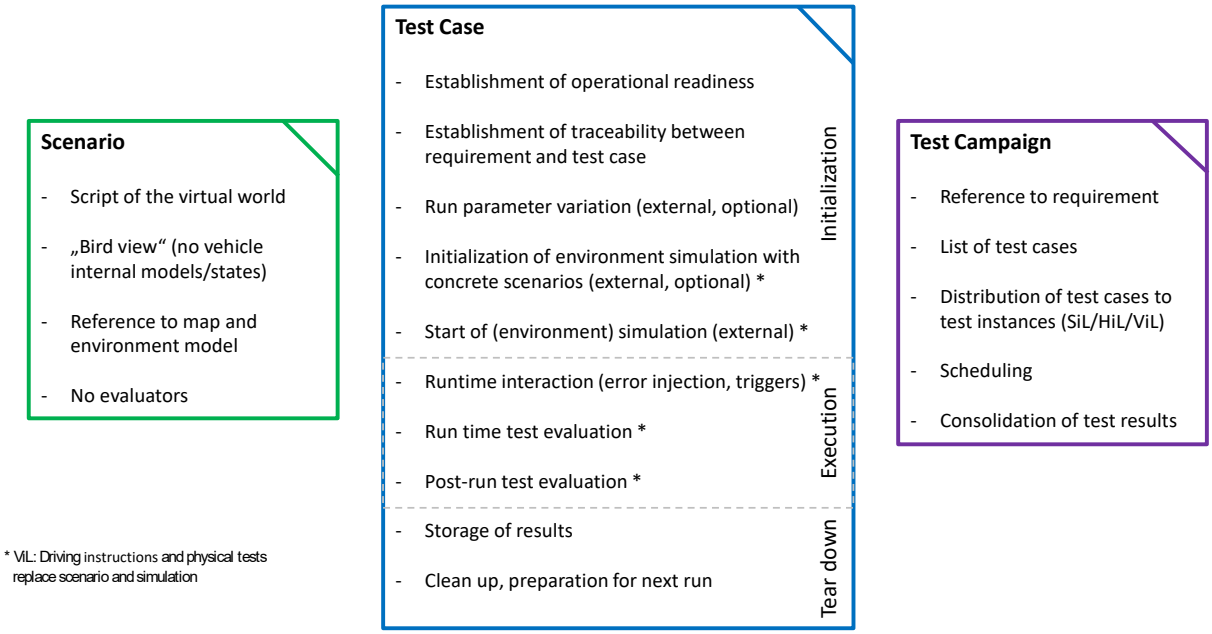


Abbildung 35: Spezifikation von Szenarien, Tests und Testkampagnen

Basierend auf dieser Trennung wurde für die Spezifikation konkreter Test- und Analyseaufgaben ein XML-basiertes Format entwickelt, das mit Hilfe generischer Schlüsselwörter (z. B. „Initialization of Environment Simulation“) parametrierbare, Testinstanz-spezifische Umsetzungen referenziert (siehe Abbildung 36). So wird die Eindeutigkeit von Test- und Analysebeschreibungen bei gleichzeitiger Übertragbarkeit gewährleistet.

[6] Abbildung zum Szenario sowie das ASAM Logo entstammen ASAM OpenSCENARIO 1.2.0, ASAM e.V., 2022: (https://www.asam.net/index.php?eID=dumpFile&t=f&f=4908&token=ae9d9b44ab9257e817072a653b5d5e98ee0babf8)

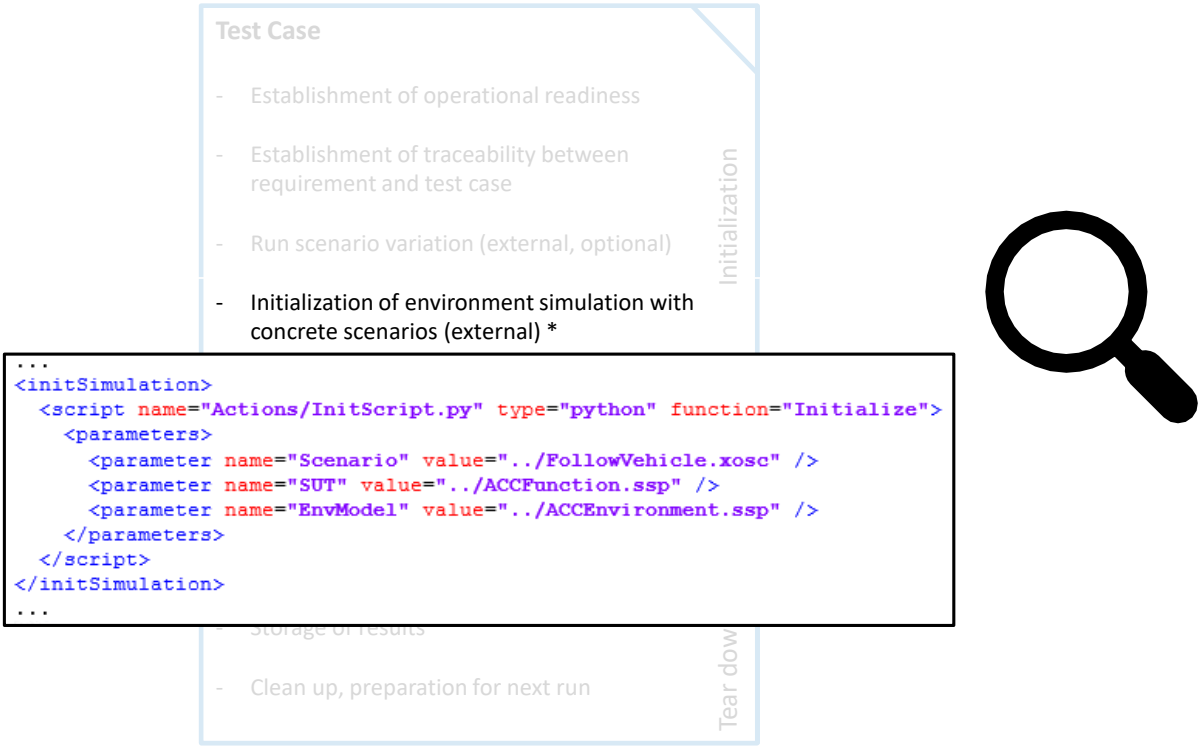


Abbildung 36: Schlüsselwortbasierter Ansatz zur Spezifikation von Test- und Analyseaufgaben

Der entwickelte Ansatz zur Spezifikation von Test- und Analyseaufgaben wurde von den Projektpartnern zum Halbzeitevent des Projekts in einem szenariobasierten Demonstrator präsentiert. Anhand eines auf einer SiL Testinstanz ausgeführten, einfachen Tests, in dem ein automatischer Abstandsregler (Adaptive Cruise Control, ACC) in der Folgefahrt von zwei Fahrzeugen als SuT auf seine Wirksamkeit hin bewertet wurde, konnte die Nutzbarkeit der Testspezifikation und die Vorteile des schlüsselwortbasierten Ansatzes glaubhaft demonstriert werden (siehe Abbildung 37).

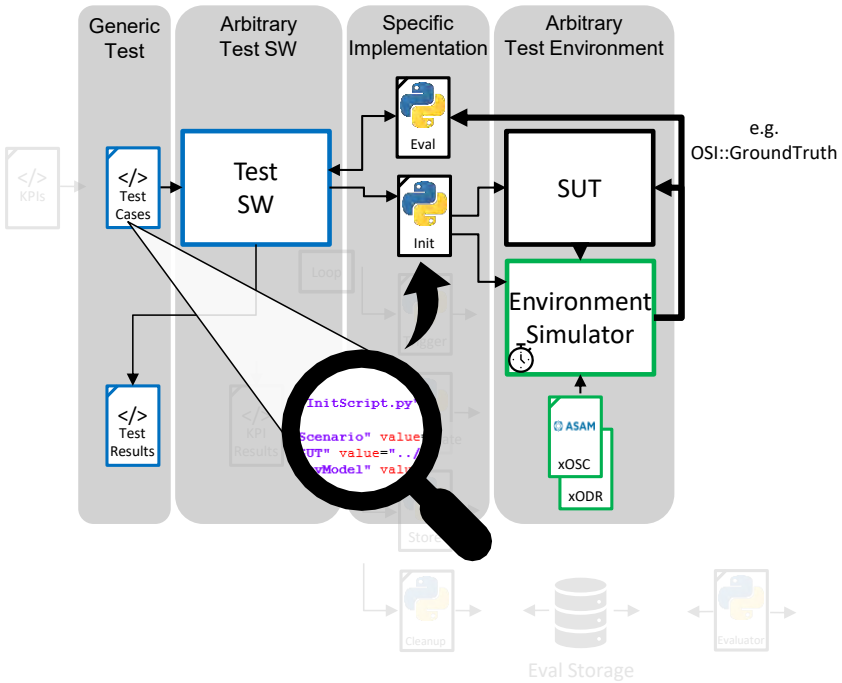


Abbildung 37: Demonstrator für schlüsselwortbasierte Spezifikation von Tests- und Analyseaufgaben

Im Rahmen des Demonstrators wurde der Test sowohl auf einer eigens im Projekt entwickelten Simulationsumgebung als auch auf einer industriell erprobten Lösung des Projektpartners dSPACE ausgeführt. Dank des generischen Ansatzes, einer im Rahmen des Projekts entwickelten einfachen Ausführungssoftware und standardisierten Schnittstellen konnte die einfache Übertragbarkeit des Tests und damit der Nutzen des Ansatzes glaubhaft dargestellt werden. Die Implementierungen wurden mitsamt den Beispieldateien veröffentlicht.
Auf Basis der entstandenen Ergebnisse wurde von den Projektpartnern beim ASAM e.V. ein Projekt für die Bewertung der Notwendigkeit einer standardisierten Spezifikation von Test- und Analyseaufgaben ins Leben gerufen, die ASAM Test Specification Study Group (https://www.asam.net/project-detail/test-specification/). Das im Dezember 2021 abgeschlossene Projekt mündete in einen Report, der anhand einer Analyse umfangreicher Use-Cases den Bedarf an einer standardisierten Spezifikation für Test- und Analyseaufgaben und die Separation von Test und Szenario klar darlegt. Ein Folgeprojekt für die Initiierung der Standardisierungsaktivitäten ist bereits im Aufbau (siehe https://www.asam.net/project-detail/c-2023-01-asam-opentest/).

Ein wichtiges Ziel der Entwicklung einer durchgängig nutzbaren Spezifikation für Test- und Analyseaufgaben bildete außerdem die Gewährleistung von deren eindeutiger Herleitung aus zugrundeliegenden (Haupt-) Anforderungen (sog. Key Performance Indikatoren, KPIs) unter Gewährleistung Ihrer Übertragbarkeit. Die Ableitung eindeutiger, konkreter Tests und Analysen aus teils generisch und natürlichsprachlich formulierten Anforderungen als wichtiges Spannungsfeld in der automobilen Entwicklung und Absicherung bildete dabei die Haupt-Herausforderung (siehe Abbildung 38).

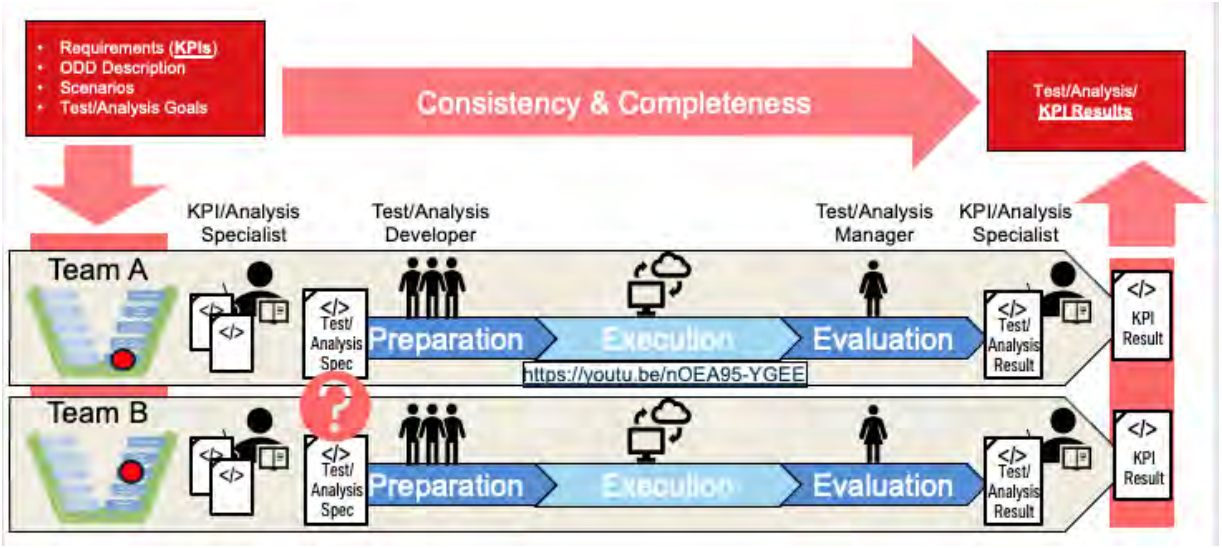


Abbildung 38: Übersicht zum Gesamtkonzept

Als vielversprechendste Ansätze hinsichtlich ihrer Eignung für die formalisierte Beschreibung konkreter Tests und Analysen sowie den zugehörigen Anforderungen wurden die Ontologie des ASAM OpenXOntology Standardisierungsprojekts und die Automotive Global Ontology (AGO) im Rahmen des SET Level Projekts von den Projektpartnern integriert und evaluiert. Bei Ontologien handelt es sich um formalisierte Beschreibungen des Wissens innerhalb einer bestimmten Domäne. Die zugrundeliegenden, standardisierten Definitionen von Konzepten ermöglichen die konsistente Darstellung des Wissens sowohl in einer menschen- als auch in einer maschinenlesbaren Form. Sie erlauben weiterhin die Identifikation von logischen

Inkohärenzen und Konflikten und die Identifikation von nicht explizit spezifizierten Zusammenhängen und bilden daher ein mächtiges Werkzeug für die konsistente Beschreibung von Test- und Analyseaufgaben sowie den zugehörigen Anforderungen auf verschiedenen Abstraktionsebenen.
Leider beschränken sich die beiden Ontologien bislang auf die Domäne der Umfeldbeschreibung, so dass sie um test- und analysespezifische Aspekte ergänzt werden müssen.
Im Projekt SET Level wurde aufgrund der Synergien mit den bereits im Projekt verankerten ASAM OpenX Standards und der bereits bestehenden Erweiterungskonzepte die im ASAM OpenXOntology Projekt entwickelte Ontologie als Grundlage für die Entwicklung test- und analysespezifische Erweiterungen gewählt (siehe Abbildung 39).

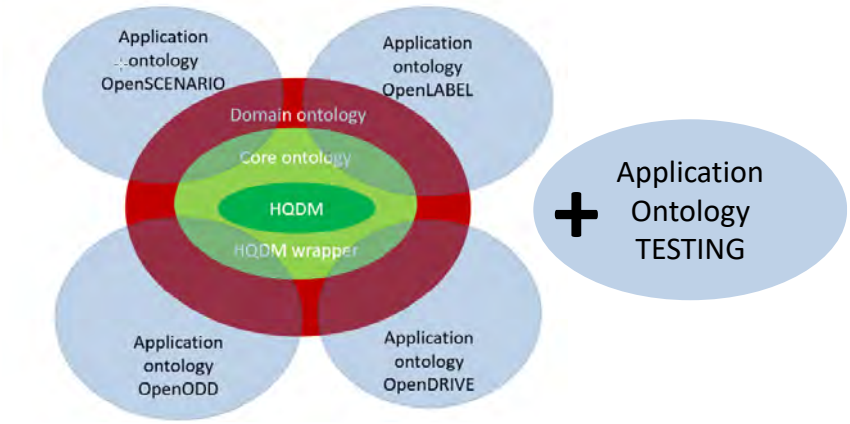


Abbildung 39: Integration der Applikation Test in ASAM OpenXOntology

Basierend auf dem bereits zum Halbzeitevent verwendeten Beispiel eines ACC, getestet auf einer SiL-Testinstanz in einer einfachen Folgefahrt, wurde die Ontologie des OpenXOntology Projekts durch die Projektpartner exemplarisch um die Applikation Testing erweitert. Hierzu wurde mit Hilfe des frei verfügbaren Ontologie-Editors Protégé (https://protege.stanford.edu/) die bestehende Ontologie des Projekts OpenXOntology geladen und exemplarisch um alle erforderlichen testspezifischen Parameter ergänzt. Bei der Umsetzung wurde das Konzept der so genannten Ressource Description Framework (RDF) Triplets angewandt, das Beziehungen zwischen Objekten (Things) und deren Beziehungen (Relations) stets in der Form Subjekt-Prädikat-Objekt abbildet. Dieses Vorgehen erlaubte exemplarisch die Ableitung eines szenariobasierten Tests aus einer natürlichsprachlichen Anforderung (siehe Abbildung 40).

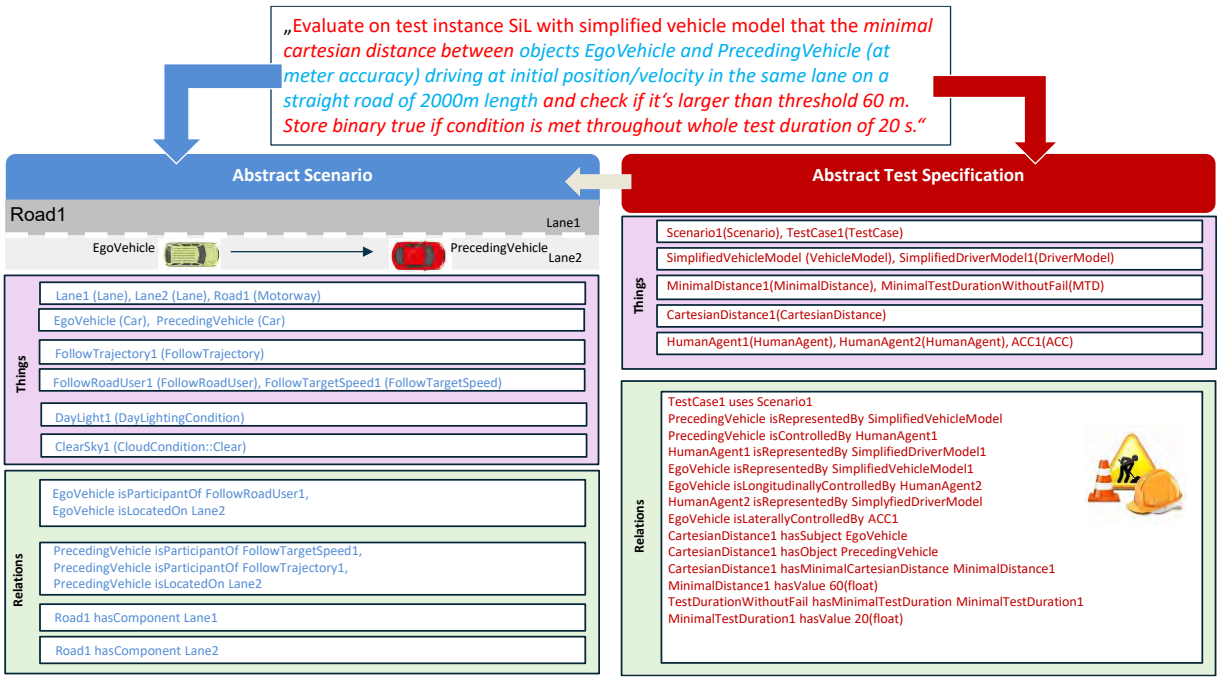


Abbildung 40: Ontologiebasierte Ableitung eines szenariobasierten Tests auf Basis einer natürlich-sprachlichen Anforderung

Der ontologiebasierte Ansatz bildet die Grundlage für die Ableitung eindeutiger, durchgängig nutzbarer Test- und Analysebeschreibungen aus natürlichsprachlichen Anforderungen. Damit zeigt er das Potential für folgende Anwendungen

- Einheitliches Verständnis von Test- und Analyseaufgaben für
  - Standardisierungsgremien
  - Regulatoren
  - Fahrzeughersteller (OEMs)
  - Zulieferer
- Automatisierte Prüfung der Spezifikationen von Test- und Analyseaufgaben hinsichtlich Semantik und Vollständigkeit
- (Semi-)Automatisierte Erstellung konkreter Spezifikation von Test- und Analyseaufgaben
- Ontologie-gestützte Datenbanksuchen zu Modellen, Szenarien, Test- und Analyseaufgaben

Die entstandenen Ergebnisse dienten der Validierung des im Projekt SET Level entwickelten Credible Simulation Process (CSP), als Proof-of-Concept für die schlüsselwortbasierte Spezifikation und als Konzeptvorlage für eine ontologiebasierte abstrakte Beschreibung von Anforderungen. Sie bauten auf die Ergebnisse des ASAM OpenXOntology Projekts, und fanden dank des Engangements der Projektpartner zu Teilen Eingang in die ASAM Test Specification Study Group. In bereits geplanten Folgeprojekten beim ASAM werden sie weiter ausgearbeitet.

2.1.3.3.2 Formalisierte Beschreibung der Testkonfiguration

Es gab bisher kein Konzept, eine detailliertere Grundstruktur einer Architektur für eine Testkonfiguration für das Simulieren automatisierter Fahrfunktionen abzubilden. Dies wurde unter dem Begriff der Plattformarchitektur entwickelt. Ein. Beispiel für eine Plattformarchitektur ist in Abbildung 41 dargestellt. Dies wird unter anderem benötigt, wenn eine Simulation zwischen verschiedenen Partnern ausgetauscht werden soll. Alternativ wäre eine Übergabe sonst auch über Dateien bzw. Ordnerstrukturen möglich, damit ist aber immer nur eine Übergabe eines

bestimmten Standes möglich. Außerdem müssen hierfür bei Änderungen häufig wieder große Datenmengen übertragen werden.

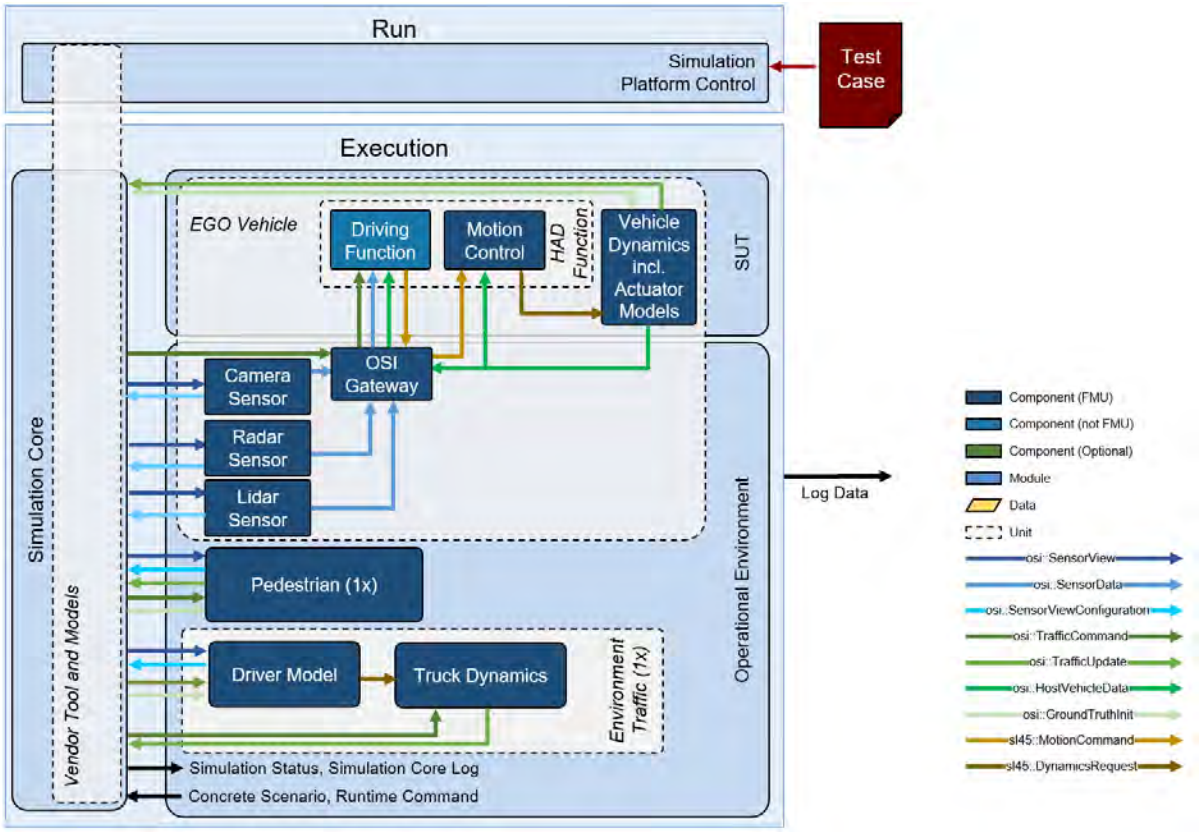


Abbildung 41: Beispiel für eine Plattformarchitektur

Zunächst wurden aus User-Stories die Anforderungen und notwendigen Inhalte für ein solches Format erarbeitet. Anschließend wurden die exakten Inhalte des Formats basierend aus der bereits entwickelten Architektur abgeleitet. Bei der Untersuchung haben sich drei verschiedene Konstellationen ergeben, bei denen eine Übergabe einer Testsetup-Konfiguration eintreten kann:

- Übergabe firmenintern (z. B. zwischen verschiedenen Abteilungen)
- Übergabe zwischen OEM und Zulieferer
- Übergabe zwischen OEM und Prüforganisation

Folgende Besonderheiten ergeben sich aus den verschiedenen Anwendungsfällen:

Übergabe firmenintern

Firmenintern ergeben sich zwei mögliche Anwendungsfälle. Zum einen die Übergabe zwischen verschiedenen Abteilungen, z. B. um verschiedene Stufen im Entwicklungsprozess (SiL / HiL …) mit der gleichen Testkonfiguration testen zu können. Zum anderen kann an verschiedenen Stellen eine Dokumentation über die durchgeführten Simulationen zum späteren Nachweis einer Validierung und Verifikation notwendig bzw. sinnvoll sein.
Oben beschriebener Anwendungsfall bedingt, dass die verschiedenen Stufen der Entwicklung stets die gleichen Schnittstellen aufweisen. Weiterhin können, falls bei beiden Abteilungen verfügbar, Verlinkungen (Netzlaufwerke etc.) anstatt Dateiübergabe genutzt werden. Falls geteilte Simulationshardware zur Verfügung steht, kann sogar auf die gesamte Simulation verwiesen werden. Bei der Übergabe bzw. Ablage zu Dokumentationszwecken ist dies

nur möglich, falls sichergestellt ist, dass im angestrebten Zeitraum die jeweiligen Orte immer noch erreichbar sind.

Übergabe zwischen OEM und Zulieferer

Neben der Übergabe zwischen OEM und Zulieferer zu Bearbeitungs- bzw. Verwendungszwecken ergibt sich auch hier als zweiter Anwendungsfall die Nutzung zur Dokumentation erfolgter Simulationen.
In beiden Fällen können Verlinkungen auf Inhalte nur erfolgen, wenn die Inhalte öffentlich oder im Rahmen der üblichen Zusammenarbeit zugänglich sind. Falls eine Bearbeitung einzelner Modelle vorgesehen ist, muss in diesen Fällen der Quellcode mit übergeben werden.

Übergabe zwischen OEM und Prüforganisation

Im Falle der Übergabe zwischen OEM und Prüforganisation sind drei mögliche Anwendungsfälle denkbar. Im ersten Fall soll die Funktion des OEMs bei der Prüforganisation validiert bzw. verifiziert werden. Hierzu muss die vollständige Testkonfiguration sowie das SuT übergeben werden. Wie schon bei der Übergabe zwischen OEM und Zulieferer sind Verlinkungen nur möglich, wenn die jeweiligen Inhalte für beide Seiten zugänglich sind. Eine Übergabe des SuTs kann aus schutzrechtlichen Gründen auch in Binärform erfolgen.
Der zweite Fall sieht eine Dokumentation der Validierung und Verifizierung vor. Hierbei muss sichergestellt sein, dass eventuell vorhandene Verlinkungen über den vorgesehenen Zeitraum verfügbar sind.
Ein dritter möglicher Fall ist eine Vorgabe einer Testkonfiguration seitens der Prüforganisation. Hier werden, wie in allen anderen Fällen auch, die Schnittstellen zum SuT mit angegeben und der OEM kann sein SuT in einem definierten Test überprüfen.
Es gab verschiedene Ideen, wie man eine solche Testkonfiguration beschreiben bzw. speichern kann. Eine davon war, einen vollständigen Container (z. B. Docker) inklusive aller in der aktuellen Testkonfiguration vorhandener Elemente zu nutzen. Aus Gründen der Praktikabilität und der zu erwartenden Speichergrößen wurde diese Idee aber wieder verworfen. Insbesondere die Toolabhängigkeit bzw. die fehlende Möglichkeit schnell die zu verwendende Simulationsumgebung oder einzelne Simulationsmodelle auszutauschen, spricht gegen diese Lösung. Als Archivierungsmöglichkeit könnte eine solche Lösung gegebenenfalls eine Rolle spielen.
Auch die populären Simulationstools bieten jeweils eine Möglichkeit einzelne Projekte z. B. in Form eines Projektordners zu archivieren. Hier fehlt jedoch wiederum die Möglichkeit schnell das jeweilige Tool zu wechseln. Außerdem enthalten die Projektordner nicht unbedingt die Voraussetzungen (Grundsetup) die notwendig sind, um die Simulation erneut ausführen zu können. Auch sonstige externe Tools, die gegebenenfalls installiert werden müssen, sind im Projektordner typischerweise nicht enthalten.
Letztendlich wurde sich für ein übergeordnetes beschreibendes XML-Dokument entschieden. Zusätzlich zu diesem Dokument können weitere Dateien, die in diesem Dokument erwähnt werden, mit übergeben werden. Wie schon erwähnt, können auch Verlinkungen auf Netzlaufwerke bzw. Repositories erfolgen, hierbei muss aber stets gewährleistet sein, dass beide Partner auf den jeweiligen Speicherort zugreifen können.
Ausgehend von einem früheren Stand, der weiter oben bereits vorgestellten Abbildung 41 zur Simulationsarchitektur, sind für jeden Block die zu beschreibenden Details festgelegt worden. Die genauen Details werden im Folgenden vorgestellt und erläutert. Das System under Test (SuT) wird in diesem Ansatz nicht explizit berücksichtigt, da sich dieses je nach Anwendungsfall unterscheiden kann. Um das Verhalten der anderen Simulationskomponenten jedoch

abbilden zu können, soll zumindest der Typ des SuTs sowie dessen Einbindung in die Gesamtsimulation beschrieben werden.
Alle Elemente werden als Freitext ausgeführt. Für eine eventuelle spätere Automatisierung könnte dies angepasst bzw. die möglichen Eingaben eingeschränkt werden.
Der Hauptknoten des XML-Schemas, wie in Abbildung 42 gezeigt, beinhaltet die Blöcke, die auch in der erwähnten Simulationsarchitektur zu finden sind.

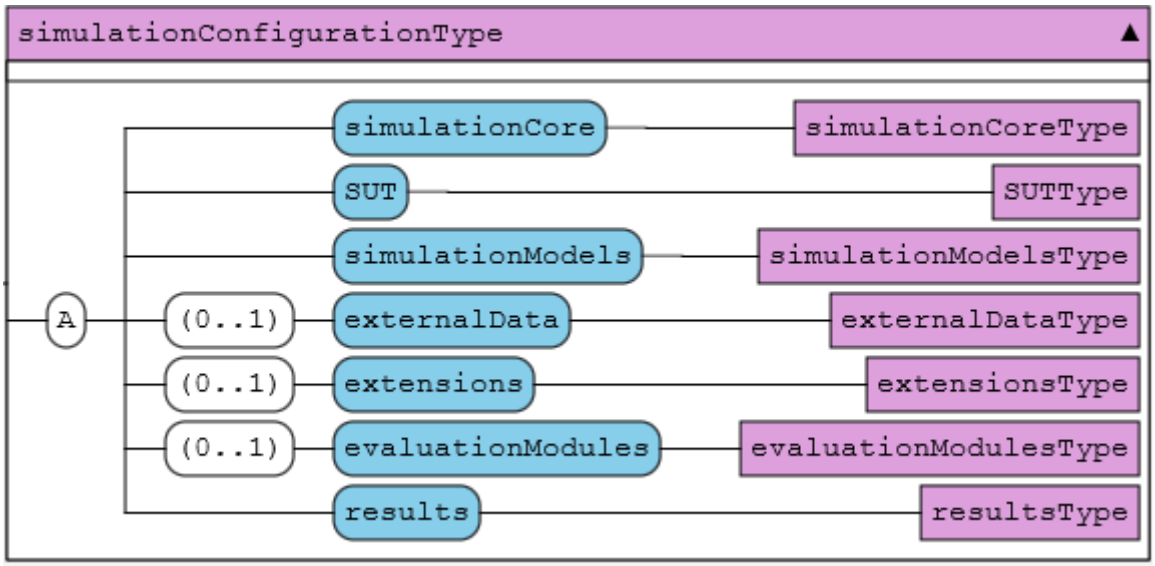


Abbildung 42: Subknoten des simulationConfigurationType

Jeder Subknoten darf dabei exakt einmal vorkommen, wobei die Subknoten *externalData*, *extensions* und *evaluationModules* optional sind und nur verwendet werden müssen, wenn die Konfiguration entsprechende Elemente enthält. Dies wird in der Darstellung durch die Kardinalität *(0..1)* vor jedem entsprechenden Subknoten dargestellt. Auch bei den anderen Submodulen sind einige Elemente optional, insbesondere wenn diese nur zusätzliche Beschreibungen enthalten. Die Struktur der einzelnen Subknoten sowie eventuelle Besonderheiten werden im Folgenden erläutert.
Wie in Abbildung 43 zu sehen, sind die Subknoten *startCommand* und *additionalPackages* des Simulation-Core optional, alle anderen Subknoten müssen exakt einmal vorkommen. Dies sind vor allem die verwendete *simulationSoftware* zusammen mit der entsprechenden *version*, welche auf einem aufgeführten *operatingSystem* ausgeführt werden kann. Damit ist die Grundstruktur der Simulation-Core hinreichend beschrieben.

Abbildung 43: Subknoten des simulationCoreType

Zusätzliche Pakete können über mehrere *additionalPackages* und deren Subknoten eingebunden werden. Im Einzelnen können bzw. müssen je zusätzlichem Paket verpflichtend der *name* (Name des zusätzlichen Pakets) angegeben werden. Der *path* (Pfad, mit dem das Paket eingebunden ist), *author* (Author), *shortDescription* (Kurzbeschreibung des Zwecks), und die *version* (verwendete Version) sind optional.
Die Beschreibung des SuT enthält hauptsächlich die Definition der zugehörigen Interfaces. In den sonstigen Feldern soll eine kurze Information über das in dieser Konfiguration vorgesehene SuT gegeben werden, ohne dass das SuT selbst verlinkt wird. Dazu sind *name* (Name oder Bezeichnung), *version*, *shortDescription*, *interfaces* (Schnittstellenbeschreibung) vorgesehen, optional kann man noch ein *comment* (Kommentar) angeben.
Die zugehörigen Schnittstellen werden über den sogenannten *interfacesType* abgebildet. Hier gibt es die Möglichkeit, eine beliebige Kombination einzelner Verbindungen oder aber beschreibender Dateien anzugeben. Die jeweiligen Interfaces werden über *from* (Quelle), *to* (Ziel) und *dataType* (Datentyp) beschrieben, auch hier gibt es optional die Möglichkeit, einen comment hinzuzufügen.
Zur Angabe weiterer externer Daten (z. B. Trajektoriendaten) kann ein *externalDataType* verwendet werden, der aus einem *dataType* mit Angaben zu *path* und optional *shortDescription* und/oder *comment* bestehen kann. Außerdem können allgemeine Parameter zur Konfiguration angegeben im *parametersType* werden. Letztere können sowohl einzeln als *parameter* mit *name*, *value* (Wert), *unit* (Maßeinheit) und optional *shortDescription* bzw. *comment*, als auch als *parameterFile* (Parameter-Datei) angegeben werden.
Die zur Simulation gehörigen Modelle werden, falls vorhanden, einzeln aufgeführt. Zusätzlich zu den einzelnen Modellen werden auch die Verbindungen zwischen den Modellen mit abgebildet. Der hier genutzte *interfacesType* ist identisch zu dem beim SuT genutzten und wird daher nicht noch einmal beschrieben.
Die einzelnen Modelle werden, wie in Abbildung 44 dargestellt, jeweils über den *simulationModelType* beschrieben, welcher *name*, *path*, *version* und *shortDescription* benötigt. Falls zusätzliche Parameter angegeben werden müssen, kann dies analog zum bereits bei *externalData* beschriebenen Parametertyp geschehen.

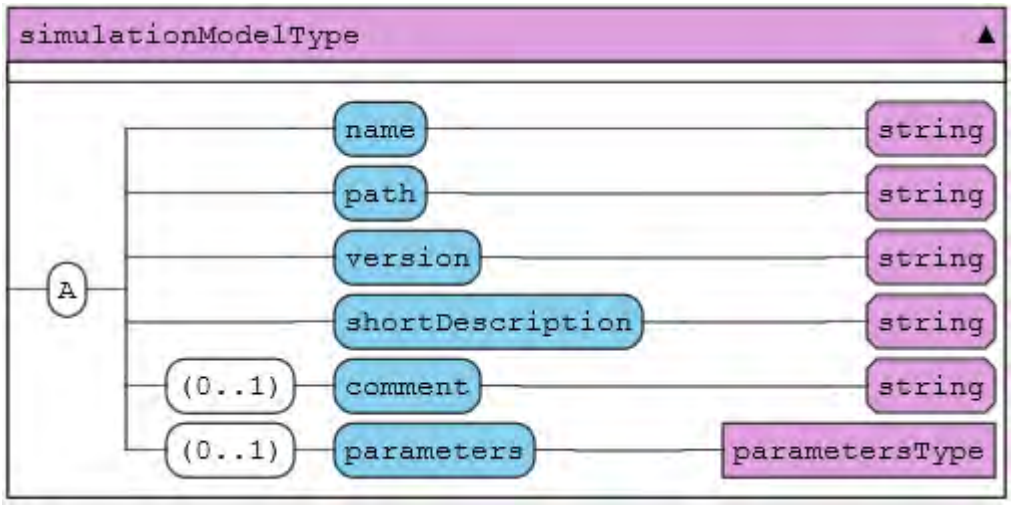


Abbildung 44: Subknoten des simulationModelType

Falls sonstige Erweiterungen in der Konfiguration genutzt werden sollen, können diese hier angegeben werden. Es können beliebig viele Instanzen des *extensionsTypes* genutzt werden. Die für die einzelnen Extensions genutzten *parameters* und *interfaces* entsprechen den schon bereits beschriebenen Formaten. Außerdem sind einige Subknoten wieder optional.

Die Details zu den *EvaluationModules* können über zusätzliche Informationsblöcke abgebildet werden. Es können wiederum beliebig viele einzelne *Evaluatoren* über den zentralen Knoten abgebildet werden. Wie in Abbildung 45 zu sehen ist, sind nur *name*, *path*, *version*, *shortDescription* und die *interfaces* für die einzelnen Evaluatoren verbindlich. Optionale Angaben beinhalten *comment*, *parameters*, *expectedResult* (erwartetes Ergebnis) und *passFailCriteria* (Erfolgskriterien)

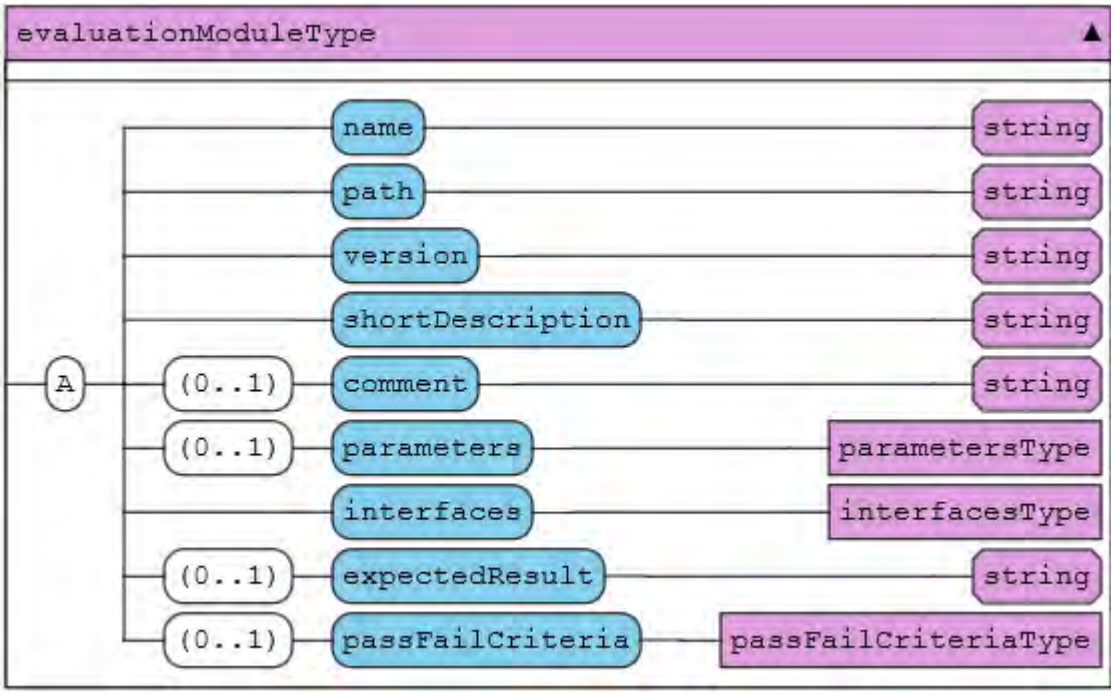


Abbildung 45: Subknoten des evaluationModuleType

Der letzte zu beschreibende Informationsblock beschreibt das Format der Ergebnisse der Simulation. Hierzu können einzelne *signal* (Signale) oder gesamte *signalDescriptionFile* (beschreibende Dateien) angegeben werden. Falls die einzelnen Signale beschrieben werden sollen, so kann dies über die *name*, *unit* und *shortDescription* erfolgen. Zusätzlich kann eine *minimumFrequency* (Mindestfrequenz) und *comment* für das jeweilige Signal angegeben werden.

2.1.3.3.3 ASAM OSI Schnittstellen

Während der Projektlaufzeit wurden viele Schnittstellen zu und für ASAM OSI geschaffen, um bestimmte Informationen vor Initialisierung bereitstellen oder zur Laufzeit beeinflussen zu können. Diese sind auch nach Ende des Projektes im öffentlichen Bereich des GitLab (siehe https://gitlab.setlevel.de/open/osi) zugänglich.

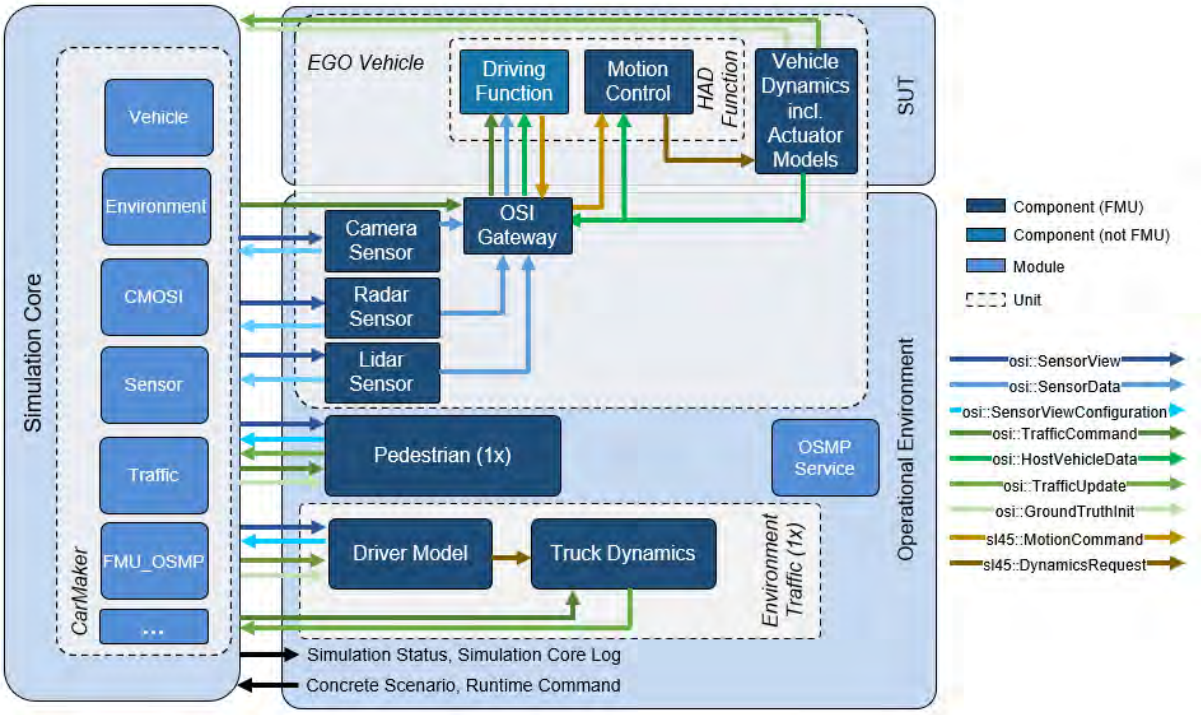


Abbildung 46: ASAM OSI Schnittstellen anhand des Beispiels der Simulationsarchitektur von SUC 2

Einige der Nachrichten waren bisher nicht vorhanden und wurden erst durch die Arbeiten im Projektkontext in den OSI Standard eingebracht. Abbildung 46 zeigt einige dieser Nachrichten im Kontext des SUC 2, bei dem verschiedene Module Informationen über diese Nachrichten austauschen. Dazu gehören unter anderem:

OSI::TrafficCommand

Top-Level Nachricht zur ereignisbasierten Steuerung von Verkehrsteilnehmermodellen (TrafficParticipant Modellen) durch die Szenario-Engine.
Dies können Vekehrsaktionen (*TrafficActions*) wie z. B. Folgen einer Trajektorie (*FollowTrajectoryAction*), Geschwindigkeitsbefehle (*SpeedAction*) oder ähnliche beinhalten und werden zur Laufzeit gesendet. Dabei ist zu beachten, dass diese Nachricht eine Multiple-Choice-Auswahl ist, d. h. es sollen beispielsweise aus Plausibilitätsgründen nur bestimmte Kombinationen von atomaren Verkehrsaktionen in bestimmten Zeitintervallen übertragen werden. Die diesbezüglichen Einschränkungen sind nicht Bestandteil dieser Meldung, werden aber als Aufgabe der Szenarienbeschreibung oder -Engine gesehen.
Alle Verkehrsaktionen werden nur einmal kurz vor ihrem Start gesendet. Dies gilt auch dann, wenn zu erwarten ist, dass ihre Ausführung Simulationszeit in Anspruch nehmen wird. Um dem Verkehrsteilnehmermodell mitzuteilen, dass bestimmte Aktionen beendet werden müssen oder sollen, sind innerhalb dieser Nachricht explizite Aktionen verschachtelt (*AbortActionsAction*, *EndActionsAction*), die einen Verweis auf die jeweiligen Aktionen enthalten. Darüber hinaus gibt es eine *TrafficCommandUpdate*-Nachricht für den Verkehrsteilnehmer, um potenziell verworfene Aktionen zurückzumelden.

OSI::TrafficCommandUpdate

Top-Level Message zur Rückmeldung des TrafficParticipant Modells an die Szenario-Engine über die Nicht-Ausführung eines empfangenen *TrafficCommand*.
Diese Nachricht ermöglicht es, dem Verkehrsteilnehmermodell, Aktualisierungen über die Ausführung seiner empfangenen *TrafficCommand*-Eingabe an die Szenario-Engine zu

senden. Während Verkehrsaktionen in der Regel erfolgreich vom Verkehrsteilnehmer ausgeführt werden, kann es Aktionen geben, die der Verkehrsteilnehmer entweder aus fähigkeits- oder situationsbedingten Gründen nicht ausführen kann.
Diese Nachricht ermöglicht es, einem Verkehrsteilnehmer Feedback zu senden, wenn eine Aktion nicht, wie vom *TrafficCommand* angefordert, ausgeführt werden kann. Derzeit ist es nicht möglich, den genauen Grund für die Nichtausführbarkeit oder die fehlgeschlagene Ausführung zu standardisieren, da die Ursache vielfältig sein kann. Die Verantwortung für die Entscheidung über erfolgreiche oder nicht erfolgreiche Ausführung des Szenarios liegt vollständig bei der Szenario-Engine.

OSI::TrafficUpdate

Top-Level Nachricht zur Rückmeldung eines Verkehrsteilnehmers über seinen aktuellen Status bezüglich Position und Zustand.
Die Nachricht ist für die Aktualisierung der Daten genau eines Verkehrsteilnehmers inklusive eines optionalen Anhängers bestimmt. Aus Gründen der Bequemlichkeit und Konsistenz werden die aktualisierten Informationen als *MovingObject* bereitgestellt. Bestimmte Felder dieser Teilnachricht brauchen nicht gesetzt zu werden und werden von der Simulationsumgebung ignoriert, da es sich um statische Informationen handelt. Um die Erstellung eines separaten Nachrichtentyp nur für die nicht statischen Informationen zu vermeiden, wird auf eine bereits bestehende Nachricht zurückgegriffen.

OSI::SensorViewConfiguration

Top-Level Nachricht zur Konfiguration der Eingangsdatenvon *osi3::SensorView* an den jeweiligen Sensor.
Die *SensorViewConfiguration*-Meldung wird in der Initialisierungsphase einer Simulation verwendet, um die *SensorView*-Konfiguration für einen bestimmten *SensorView*-Eingang auszuhandeln. Sie ist auch als Unternachricht in *SensorView*-Nachrichten enthalten, um anzuzeigen, dass die Konfiguration der Sensoransicht für eine bestimmte *SensorView*-Nachricht gültig ist.
SensorViewConfiguration-Daten haben zwei Hauptanwendungen:

- Ermöglichung der Umweltsimulation, um einem Sensormodell die notwendigen Eingaben zu liefern.
- Ermöglicht es einem Sensormodell zu prüfen, ob die Eingabe seinen Anforderungen entspricht. Entspricht die Eingabe nicht den Anforderungen, kann das Sensormodell die Simulation beenden.

SensorViewConfiguration-Daten sind für die automatische Konfiguration der *SensorView*-Schnittstelle zwischen einer Umgebungssimulation und einem Sensormodell vorgesehen. Die Daten sind nicht als Mechanismus zur Parametrisierung eines generischen Sensormodells gedacht.
Während der Initialisierungsphase gibt es zwei Quellen für *SensorViewConfiguration*-Daten:

- *SensorViewConfiguration*-Daten können vom Sensormodell an die Umgebungssimulation geliefert werden. In diesem Fall beschreiben die Daten die Eingabekonfiguration, die vom Sensormodell angefordert wird. Wenn das Sensormodell solche Daten nicht bereitstellt, greift die Umgebungssimulation auf die manuelle Konfiguration der Sensoransicht zurück.
- *SensorViewConfiguration*-Daten können von der Umgebungssimulation bereitgestellt werden. Als Reaktion auf die Anforderung des Sensormodells oder auf der Grundlage

der manuellen Konfiguration konfiguriert die Umgebungssimulation den Eingang und liefert eine neue Nachricht, die die aktuelle Konfiguration beschreibt.

Die vom Sensormodell angeforderte Konfiguration kann sich von der durch die Umgebungssimulation bereitgestellten Konfiguration unterscheiden. Dies ist der Fall, wenn die Umgebungssimulation eine angeforderte Konfiguration nicht unterstützt oder wenn die angeforderte Konfiguration mehrdeutig ist. Als Reaktion auf diese Abweichung kann das Sensormodell entweder diese Abweichung akzeptieren und sich daran anpassen oder es kann die Simulation beenden, um anzuzeigen, dass es die Abweichung nicht akzeptieren kann. Die Paketierungsschicht definiert die Besonderheiten dieses Autonegationsmechanismus. Nach der Initialisierungsphase liefert die Umgebungssimulation die tatsächliche Konfiguration der Sensoransicht als Teil jeder *SensorView*-Nachricht.
Außerdem wurden sowohl in *osi3::SensorView* als auch in *osi3::FeatureData* neue Erweiterung zur Beschreibung von Lidar-spezifischen Effekten eingebracht. Zusätzlich gab es noch Erweiterungen für die Abbildung von Zwischenrepräsentationen für Kamerasensorsignalverarbeitung in *CameraSensorView* bzw. *CameraSensorData*.

OSI::SensorView

Top-Level Nachricht welche den Input für ASAM OSI-Sensormodelle liefert.
SensorView-Meldungen werden von *GroundTruth*-Meldungen abgeleitet. Alle Informationen über die Umgebung werden in Bezug auf das virtuelle Sensorkoordinatensystem angegeben, mit zwei Ausnahmen:

- Physikalische technologie-spezifische Daten, die in Bezug auf das physikalische Sensorkoordinatensystem angegeben werden, das in der Montageposition des entsprechenden physikalischen Sensors spezifiziert ist. Ein Beispiel für technologiespezifische Daten ist *image_data*, Teil von *osi3::CameraSensorView*
- Die GroundTruth wird im globalen Koordinatensystem angegeben.

OSI::FeatureData

Top-Level Nachricht welche als Interface die momentanen Sensordaten ohne jegliche Historie zur Verfügung stellt.

FeatureData-Meldungen enthalten erkannte Features im Referenzrahmen eines Sensors. FeatureData-Meldungen werden aus GroundTruth-Meldungen generiert. Sie dienen z. B. als Eingabe für Sensormodelle, die die Objekterkennung simulieren, oder für Modelle zur Sensorfusion.

Der ASAM OSI-Standard definiert alle Signale prinzipiell als optional. Um dennoch definieren zu können, welche Inhalte der Nachrichten für eine spezifische Simulationsaufgabe verpflichtend sind, sollen Regeln, sogenannte *OSI-Rules* im .yaml Format verwendet werden. Hierfür wurden entsprechende Beispieldateien für die ASAM OSI-Nachrichten *SensorView* und *SensorData* erstellt. Hierbei wurde der FMI Standard 2.x verwendet, welcher keine Arrays unterstützt. Deswegen musste eine Behelfslösung erarbeitet werden, um die benötigten Funktionalitäten bereitstellen zu können. Diese Funktionalitäten werden im FMI Standard 3.0 besser unterstützt. Um die Behelfslösung testen und dokumentieren zu können, wurde ein Proof-of-Concept (PoC) der Architektur einer Kamera im SSP Format erstellt, welche in Abbildung 47 zu sehen ist.

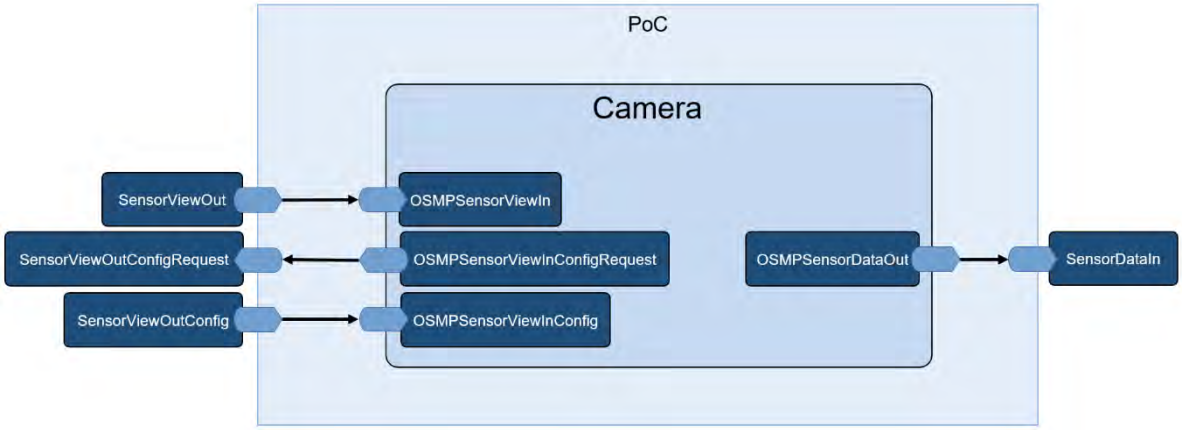


Abbildung 47: SSP Darstellung des Proof-Of-Concepts: "Camera"

Zusätzlich wurden einige projektspezifische Nachrichten erstellt, die nicht in den offiziellen ASAM OSI Standard eingeflossen sind.

sl45::DynamicsRequest

Die SET Level projektspezifische Top-Level Nachricht zur Ansteuerung des Fahrzeugmodells ausgehend vom Motion-Controller bzw. der allgemeinen Agentenmodelle.
Diese Nachricht besteht im Wesentlichen aus der Ziel-Längsbeschleunigung *ax*, aus der sowohl die Zielgrößen für den Antriebsstrang als auch für die Bremse abgeleitet werden. Zur Steuerung des Fahrzeuges in Querrichtung wird die Krümmung verwendet. In einem ersten Entwurf dieser Nachricht wurde anstelle der Krümmung die Gierrate verwendet, jedoch kann dann bei Fahrzeugstillstand, d.h. Geschwindigkeit gleich Null, keine Zielgröße für die Lenkung definiert werden. Die Vorgabe der Krümmung ist geschwindigkeitsunabhängig nutzbar und hat in der finalen Version der Nachricht Verwendung gefunden.

sl45::MotionCommand

Die SET Level projektspezifische Top-Level Nachricht zur Beschreibung des dynamischen Zustandes eines Objektes.
Diese Message beschreibt die voraussichtliche Bewegung eines Fahrzeugs in der nahen Zukunft. Sie umfasst die Trajektorie, der das Fahrzeug folgen soll, sowie den aktuellen dynamischen Zustand, der die aktuellen 2-D Koordinaten, die Orientierung und die dynamischen Eigenschaften des Fahrzeugs wie Geschwindigkeit und Beschleunigung enthält.

2.1.3.3.4 Zusammenhänge zu anderen OpenX Standards

Generell gibt es verschiedene Arten von Verkehrsregeln, die in drei größere Gruppen unterteilt werden:

- Direkt objektbasierte Verkehrsregeln: Geschwindigkeitsbegrenzungsschilder, Vorfahrtsschilder, Ampeln etc.
- Indirekt objektbasierte Verkehrsregeln: Ortseingangs- bzw. Ortsausgangsschilder, Fahrbahnmarkierungen, Zusatzzeichen etc.
- Allgemeine Verkehrsregeln: Vorfahrtsregeln, Rechtsfahrgebot, Abstand etc.

Generell können alle Verkehrsregeln von Land zu Land variieren und werden unterschiedlich definiert. In vielen Ländern existieren Verkehrszeichenkataloge, die eine eindeutige

Zuordnung der Bildtafel erlauben und damit eine konsistente Kennzeichnung und Identifikation ermöglichen. Für allgemeine Verkehrsregeln gibt es dagegen keine direkten Kataloge, die man gegebenenfalls in eine Simulation einbinden kann, sondern muss diese in die jeweiligen Verhaltensmodelle integrieren.
Daraus ergeben sich bestimmte Herausforderungen, wenn ein solches Verhalten zur Laufzeit der Simulation beeinflusst werden soll:

- Unterschiedliches Vorgehen, wenn nur einzelne Objekte ignoriert oder angepasst werden oder bei generellen Verkehrsregeln, die unabhängig von einzelnen Objekten gelten.
- Ein einzelnes Objekt kann auf unterschiedliche Arten identifiziert werden. Die einfachste Möglichkeit bieten die Verkehrszeichen, die an einem bestimmten Ort platziert sind und auch eindeutig zugeordnet werden können. Außerdem können mehrere Schilder desselben Typs über die Nummernzuordnungen gruppiert werden.
- Objekte wie Verkehrsleitlinien, Ortsschilder etc. können entlang einer Strecke gültig sein und sind damit nicht direkt eindeutig zu definieren.
- Allgemeine Verkehrsregeln sind nicht direkt definiert und können sich sehr unterschiedlich auswirken.

Zusätzlich hängt das Verhalten von der Verkehrssituation ab.
Das Ziel der Verwendung von intelligenten Modellen in Simulationen ist, bei komplexen Szenarien verlässlich relevante Szenen zu erzeugen. Deshalb sollen diesen Modellen Befehle übergeben werden, welche hilfreich sind, solche Szenen zu erzeugen. Teilweise wird ein solcher Befehl auch *Modus* genannt.
Eine wichtige Beobachtung ist, dass bisher Simulationen, welche intelligente Modelle verwenden, nur für andere reproduzierbar sind, wenn die Modelle, zusammen mit dem Szenario und gegebenenfalls weiteren Abhängigkeiten, weiteregegeben werden.
Ein Beispiel einer relevanten Szene ist, dass dem Ego-Fahrzeug die Vorfahrt genommen wird. Um dies zu bewerkstelligen, ist es hilfreich, dem vorfahrtnehmenden Agenten mitteilen zu können, dass er das Ego Fahrzeug nicht beachten soll. Hierbei ist wichtig zu erwähnen, dass bei Verwendung unterschiedlicher Modelle exakte Reproduzierbarkeit nicht das Ziel ist. Stattdessen reicht es, dass das generelle Verhalten für relevante Situationen reproduziert werden kann.
Nun stellte sich die Frage, ob diese Befehle für intelligente Modelle zu standardisieren sind. Der dahinterliegende Wunsch einer solchen Standardisierung ist es, modellbasierte Szenarien teilen zu können, ohne die Modelle mitliefern zu müssen. Die Gründe, Modelle auszutauschen, sind vielfältig:

- Modelle werden weiterentwickelt. Hierbei ist sicherzustellen, dass die weiterentwickelten Modelle weiterhin kompatibel mit dem Szenario sind. Standardisierte Anforderungen an Modelle bieten eine Möglichkeit, um dies überprüfen zu können.
- Es werden verschiedene Detaillierungsgrade von Modellen verwendet (sehr detaillierte- im Vergleich zu sehr einfachen Modellen).
- Wiederverwendung von Szenarien bei einem neuen Produkt.

Im Projekt SET Level wurden daher Befehle für intelligente Modelle entwickelt, implementiert und prototypisch getestet. Generell wurden neue Befehle zunächst als *UserDefinedAction* / *CustomCommandAction* (ASAM OpenSCENARIO) und *osi3::CustomAction* (ASAM OSI) implementiert und erprobt. Wenn sich ein Befehl bewährt hat, kann es als explizite Action für ASAM OpenSCENARIO und/oder ASAM OSI vorgeschlagen werden.
In ASAM OpenSCENARIO wird als *type* für *CustomCommandAction* der Wert *SET Level* verwendet, um Kollisionen mit anderen Anwendungen der *CustomCommandAction* besser

vermeiden zu können. Zukünftig können zusätzliche Werte, die *SET Level* als Prefix haben, je nach Anwendungszweck, eingesetzt werden.

Sowohl *CustomCommandAction* (ASAM OpenSCENARIO) als auch *CustomAction* (ASAM OSI) verwenden einen einfachen String als Befehl. Diese sollen möglichst identisch sein. Als Konvention wurde JSON zur Abbildung der Struktur eines Befehls verwendet. Das lässt sich einfach mit Standardmitteln einlesen und unterscheidet sich sichtbar vom XML in ASAM OpenSCENARIO. Bei der Angabe in ASAM OpenSCENARIO muss beachtet werden, dass die JSON-Syntax mit der XML-Syntax kollidieren kann, insbesondere wenn der JSON-String größer-als- oder kleiner-als-Symbole (< und >) enthält. Kollidierende Zeichen müssen bei der Definition innerhalb der XML-Datei mit der passenden Entity Reference ersetzt werden (z. B. < für < und > für >).
Ein Beispiel für solch einen Befehl ist das Befolgen von Verkehrsregeln. Die Action wirkt, bis sie durch eine andere Action derselben Art abgelöst wird.
In ASAM OpenSCENARIO *CustomCommandAction* wird der Befehl an alle Entities übermittelt, die unter Actors in der ManeuverGroup definiert sind.

```
<ManeuverGroup name="MGroup0" maximumExecutionCount="1">
 <Actors selectTriggeringEntities="0">
  <EntityRef entityRef="F1"/>
 </Actors>
 <Maneuver name="F1Maneuver0">
  <Event name="F1Maneuver0Start" priority="overwrite">
   <Action name="IgnoreRoW">
    <UserDefinedAction>
     <CustomCommandAction type="SET Level">
     {
      "trafficRules" : {
       "IgnoreRightOfWay" : true
      }
     }
     </CustomCommandAction>
    </UserDefinedAction>
   </Action>
   <StartTrigger>
    ...
   </StartTrigger>
  </Event>
 </Maneuver>
</ManeuverGroup>
```

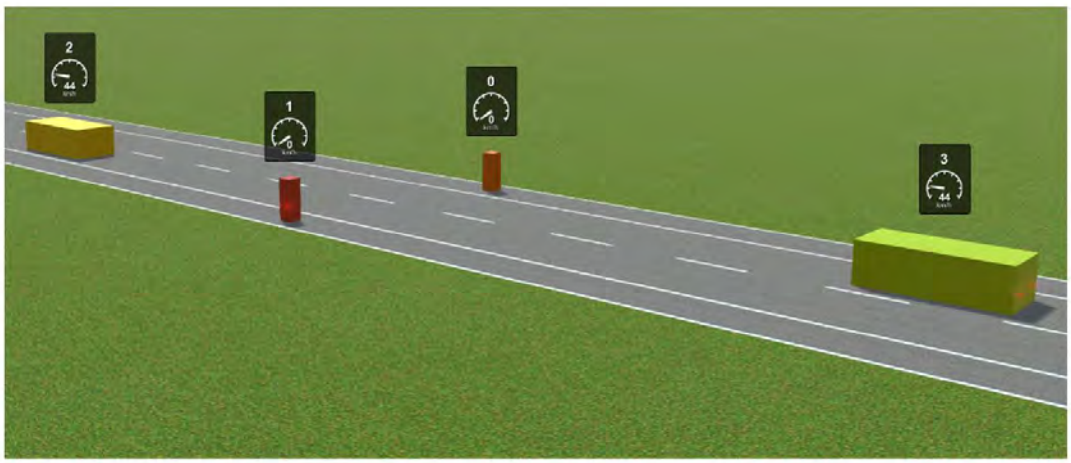
In *osi3::CustomAction* wird die Nachricht dagegen einzeln an entsprechende Verkehrsteilnehmer mit jeweiliger *traffic_participant_id* passend zu z. B. Actor / EntityRef / @entityRef gesendet. Die Darstellung erfolgt als JSON, was in Protobuf Message::DebugString() entspricht. In der Praxis kommt die Nachricht nur direkt im Speicher (als Protobuf-Objekt) oder in binär-deserialisierter Form vor.

```
{
 "version": { "major": 3, "minor": 2, "patch": 2 },
 "timestamp": { "seconds": 0, "nanos": 100 },
 "traffic_participant_id": { "value": 13 },
 "action":
```

```
 {
  "custom_action":
  {
   "action_header": { "action_id": { "value": 0 } },
   "command": "{ \"trafficRules\" : { \"IgnoreRightOfWay\" : true } }"
  }
 }
}
```

Zusätzlich gab es definierte Befehle für die Beeinflussung der Sichtbarkeit aller Verkehrsteilnehmer (*Ignore_AllTrafficParticipants*) sowie Beeinflussung der Sichtbarkeit bestimmter Verkehrsteilnehmer (*Ignore_TrafficParticipants*). Dabei nehmen diese beiden Befehle Einfluss auf gemeinsame Eigenschaften und gehören derselben Kategorie Entity Perception von Actions an. Sie schalten einen internen Zustand um (toggle). Damit enden sie formal sofort nach Auslösung, aber wirken darüber hinaus bis auf Widerruf nach.
Eine Action dieser Kategorie beendet die Wirkung aller vorherigen Actions derselben Kategorie, die an dieselbe Entity übermittelt worden sind.
Beispiel anhand eines Szenarios, wie in Abbildung 48 dargestellt:

1. "Ignore_TrafficParticipants": ["EGO", "F1"]: EGO und F1 werden ignoriert.
2. "Ignore_TrafficParticipants": ["F1"]: F1 wird weiterhin ignoriert, EGO wird wieder beachtet.
3. "Ignore_AllTrafficParticipants": false: F1 wird wieder beachtet.
4. "Ignore_AllTrafficParticipants": true: Alle Verkehrsteilnehmer werden ignoriert.
5. "Ignore_TrafficParticipants": ["F3"]: Nur F3 wird ignoriert.



Actor	# 0	# 1	# 2	# 3
Type	Pedestrian	Pedestrian	Vehicle	Vehicle
Maneuver	Cross street	Cross street	Follow road	Follow road
Additional command	Ignore other traffic participants			

Abbildung 48: Beispiel eines Szenarios mit Fußgängern und Fahrzeugen

Des Weiteren wurde untersucht, ob und in wie weit die ASAM OpenSCENARIO Actions durch die korrelierenden *osi3::TrafficCommand* Actions abgebildet werden können. Dazu wurde zwischen direkter Korrelation (1:1 Unterstützung) oder indirekter Korrelation unterschieden, bei der die Szenario-Engine bestimmte Actions, welche nicht direkt unterstützt

werden, durch geeignete Alternativen oder Kombinationen ersetzt. Eine Übersicht ist in Tabelle 2 dargestellt.

Tabelle 2: Umwandlungsmöglichkeiten der scenario engine

PrivateActions	**Control strategies**
TeleportAction	TeleportAction
LaneChangeAction	LaneChangeAction oder FollowTrajectoryAction oder FollowPathAction
SpeedAction	SpeedAction oder FollowTrajectoryAction (relative speed aktuell nicht darstellbar)
AcquireGlobalPositionAction	AcquireGlobalPositionAction
LaneOffsetAction	LaneOffsetAction oder FollowTrajectoryAction oder FollowPathAction
FollowTrajectoryAction	FollowTrajectoryAction oder FollowPathAction
AssignRouteAction	FollowPathAction
LongitudinalDistanceAction	LongitudinalDistanceAction
LateralDistanceAction	LateralDistanceAction

Dabei wurden zwei Aspekte untersucht, die Kommunikation zwischen ASAM OpenSCENARIO und der Szenario-Engine sowie von der Szenario-Engine zum Agenten. Für den ersten Teil wurden die Metadaten der Agentenmodelle mit Informationen erweitert, welche Actions innerhalb von *osi3::TrafficCommand* unterstützt werden oder nicht. Diese werden in den SRMD Dateien zu jedem Modell gespeichert und können somit automatisiert verarbeitet werden. Dazu enthalten sie sogenannte <ClassificationEntry> Einträge mit definierten *keywords*, welche dann durch einen Wert belegt werden können. Als neue *keywords* wurde die folgenden Einträge hinzugefügt, um die unterstützten Actions darstellen zu können:

- trafficparticipant.actions-supported.OSI-FollowTrajectoryAction
- trafficparticipant.actions-supported.OSI-FollowPathAction
- trafficparticipant.actions-supported.OSI-AcquireGlobalPositionAction
- trafficparticipant.actions-supported.OSI-LaneChangeAction
- trafficparticipant.actions-supported.OSI-SpeedAction
- trafficparticipant.actions-supported.OSI-AbortActionsAction
- trafficparticipant.actions-supported.OSI-EndActionsAction
- trafficparticipant.actions-supported.OSI-CustomAction
- trafficparticipant.actions-supported.OSI-LongitudinalDistanceAction
- trafficparticipant.actions-supported.OSI-LaneOffsetAction
- trafficparticipant.actions-supported.OSI-LateralDistanceAction
- trafficparticipant.actions-supported.OSI-TeleportAction
- trafficparticipant.actions-supported.OSI-TrafficCommandUpdate

Damit ergab sich die Schlussfolgerung, dass die Szenario-Engine ebenfalls Metadaten benötigt, welche direkten und indirekten Korrelationen unterstützt werden. Anhand dieser Metadaten kann bereits im Vorfeld entschieden werden, ob eine Szenario-Engine ein bestimmtes Szenario mit den vorhandenen Agentenmodellen ausführen kann. Mit Hilfe dieser Metadaten lässt sich außerdem eine Methode beschreiben, wie der Auswahlprozess geeigneter

Agentenmodelle für ein vorhandenes Szenario umgesetzt werden kann. Dazu werden aus den angegebenen Modell Repositories mit entsprechenden SRMD Dateien und der Liste der im Szenario benötigten Agentenmodell-Aktionen zwei Listen erstellt, eine mit Modellen, welche die von der Scenario Engine direkt unterstützte Aktionen verwenden kann sowie eine zweite Liste mit Modellen, welche indirekt unterstützte Aktionen verwenden. Aus diesen Listen können dann geeignete Modelle gewählt werden. Sollte weder die eine noch die andere Liste ein geeignetes Modell enthalten, so müssen die Eingangsparameter durch entsprechendes Fachpersonal überprüft werden.

2.1.3.4 Integrationsmechanismen und Methoden

Im Folgenden wird auf die Entwicklung von Methoden, das Tool-Confidence-Level sowie Modellintegration und -kopplung eingegangen.

2.1.3.4.1 Entwicklung von Methoden

Um die Vertrauenswürdigkeit von Modellen und Simulationen nachzuweisen, ist ein strukturierter und systematischer Ansatz erforderlich. Im Rahmen des Projekts SET Level wurde daher der *Credible Simulation Process* und der *Credible Modeling Process* entwickelt. In Abbildung 49 ist verdeutlicht, wie sich die Vertrauenswürdigkeitskriterien zuordnen lassen.

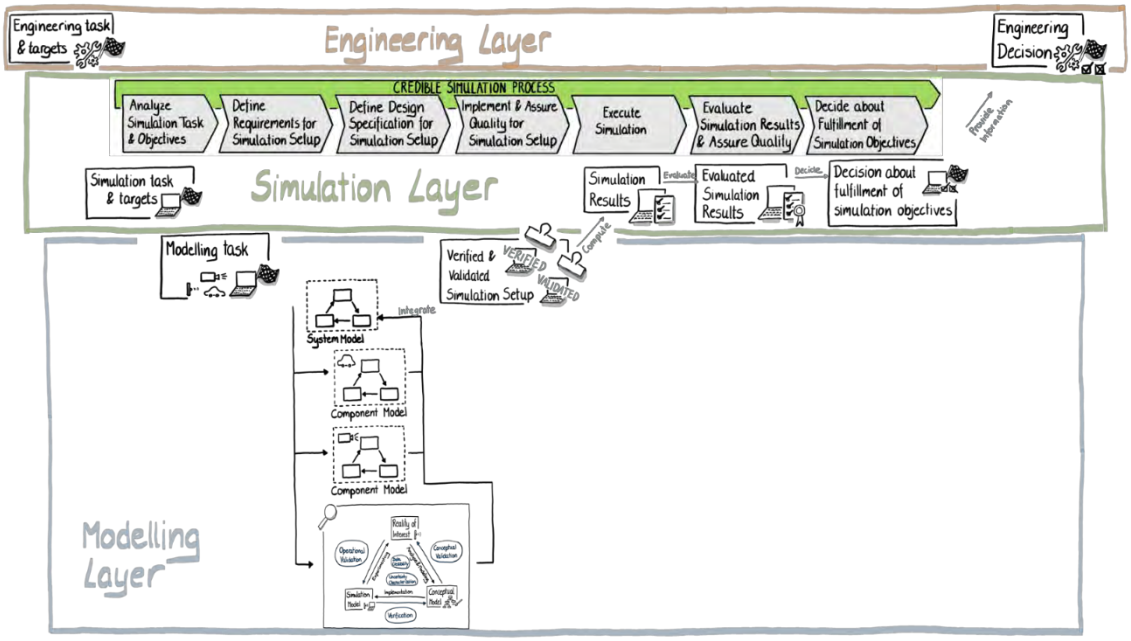


Abbildung 49: Zuordnung von Vertrauenswürdigkeiten zum Credible Simulation Process

Die Abläufe sind in mehreren Ebenen (Layer) unterteilt. Auf diesen Ebenen agieren häufig verschiedene Rollen. Beispielsweise werden die Aktivitäten im Simulation Layer häufig von Testmanagern oder Simulationsingenieuren ausgeführt. Die Aktivitäten im Modelling Layer wiederum werden auch häufig von Modellentwicklern durchgeführt, welche die Modelle der Simulationsebene bereitstellen.
Alle diese Aktivitäten laufen in der Regel iterativ ab. So können zum Beispiel neue oder geänderte Anforderungen innerhalb der Simulations- oder Modellierungsaufgabe zu neuen Anforderungen an die Modelle führen. Dadurch muss gegebenenfalls der Modellierungsprozess in Teilschritten wiederholt oder zumindest geprüft werden, wie in Abbildung 50 beispielhaft gezeigt.

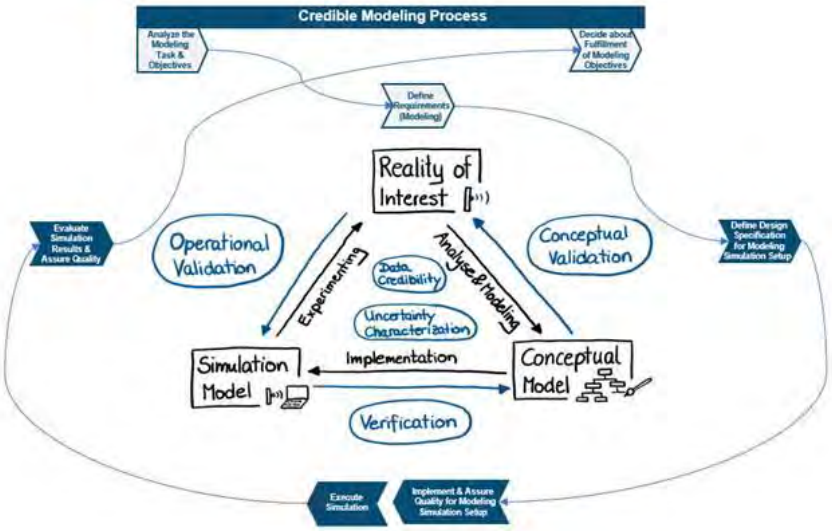


Abbildung 50: Iterativer Modellierungsprozess basierend auf neuen oder geänderten Anforderungen

Modelle, Szenarien und Tests sind dabei möglichst unabhängig zu halten und wiederzuverwenden. Grundsätzlich sollen Modelle und Tests unterschiedlich validiert werden. Daraus resultierend entsteht eine Bibliothek von validierten Modellen für die Verwendung in vielen Tests.
Dabei ist zu beachten, dass jede Simulation und jedes Modell Fehler und Grenzen besitzt. Zu jeder Simulation gehört die Evaluation der Grenzen der eingesetzten Modelle und die damit einhergehende Betrachtung, ob der beabsichtigte Anwendungsfall innerhalb dieser Grenzen liegt. Bei einer gesamtheitlichen Betrachtung kommen zu den Grenzen einzelner Modelle auch Grenzen durch die Verschaltung zu einer Co-Simulation hinzu. Dabei sind zum einen die Schnittstellen und zum anderen auch die gemeinsame Ausführungsgeschwindigkeit zu betrachten. Auch Simulationsplattform und Simulationshardware spielen hier eine Rolle.
Im Projektkontext wurden vier Arten von Grenzen bei der Simulation identifiziert:

- Inhaltliche Grenzen durch Schnittstellen
- Grenzen bei der toolseitigen Befüllung der Schnittstellen
- Grenzen der Ausführungsgeschwindigkeit
- Grenzen der Abbildungstreue eines Modells

Die inhaltlichen Grenzen der Schnittstellen sind modellspezifisch und müssen für jedes individuelle Modell analysiert werden. Exemplarisch sind an dieser Stelle zwei wesentliche Grenzen zu nennen. Bei den objektbasierten Modellen wurde ein Modellierungsansatz entwickelt, der mit sogenannten Refined-Bounding-Boxen arbeitet. Für diesen Ansatz ist notwendig, die Objektkontur oder zumindest eine Konturklasse für jedes Objekt zu kennen. Ein weiteres Beispiel sind Reflektionseigenschaften verschiedener Materialien für die reflektionsbasierten Modelle. Da Materialeigenschaften nicht standardisiert sind, sind die Reflexionsintensitäten, die verschiedene Tools berechnen, nicht vergleichbar. Eine Schnittstelle zur Übertragung von Materialeigenschaften wird in AP 4.4 beschrieben und als Standard vorgeschlagen. Das Nichtvorhandensein eines Standards begrenzt damit derzeit noch die Simulationsgüte.
Selbst wenn alle benötigten Schnittstellen vorhanden sind, ist noch nicht garantiert, dass das Tool oder Modell, welches die Schnittstelle nutzt, dazu tatsächlich befähigt ist. Die Grenzen der Simulation durch ein Modell werden also auch durch die Art der Befüllung der vorhandenen Schnittstellen definiert. Die Ausführungsgeschwindigkeit begrenzt den Einsatzzweck eines Modells bzw. stellt Anforderungen an die Simulationshardware. Bei der Ausführungsgeschwindigkeit muss zwischen der Datenübertragung über die Schnittstelle und der

Rechenzeit des Modells selbst unterschieden werden. Für die Untersuchung der Ausführungszeiten wurde ein Logger in das Modellframework integriert, der die Ausführungszeiten verschiedener Simulationsbestandteile analysiert. Damit können potenzielle Engstellen bei der Ausführungsgeschwindigkeit identifiziert werden.
Intrinsische Grenzen der Abbildungstreue von Modellen entstehen durch die Modellierung selbst und beginnen bereits bei der Definition von Modellanforderungen. Es wurde eine Methode entwickelt, um Modelle systematisch anhand von Effekten und deren Wirkzusammenhängen zu spezifizieren. Dazu werden zunächst die Wirkzusammenhänge systematisch in einer Baumstruktur gesammelt. Dieses Verfahren wird *Perception Sensor Collaborative Effect and Cause Tree* (PerCollECT) genannt. Dazu werden Effekt-Ursache-Beziehungen über Verbindungslinien in einer Baumstruktur dargestellt. Jede Verbindung ist mit einer wissenschaftlichen Quelle belegt. Abweichend von klassischen Baumdiagrammen können bei PerCollECT Ursachen jedoch mehrere verschiedene Effekte hervorrufen. In Abbildung 51 ist exemplarisch ein Ausschnitt aus PerCollECT dargestellt. Durch die schiere Menge an Effekten und Zusammenhängen ist es kaum möglich, als Einzelperson den vollständigen Stand der Technik der Sensoreffekte abzubilden. Deswegen wird der kollaborative Charakter der Methode bereits in ihrem Namen hervorgehoben. Um eine breite Zusammenarbeit an der Effektsammlung auch über die Projektgrenzen hinaus zu ermöglichen, wurde ein Bereich auf der Plattform GitHub geschaffen (siehe https://github.com/PerCollECT). Die Effekte und ihre Zusammenhänge werden in einer JSON-Datei hinterlegt und durch ein Javascript-Frontend visualisiert. Durch die Mechaniken der Git-Versionsverwaltung kann die Datenbank nachvollziehbar öffentlich verändert und erweitert werden.

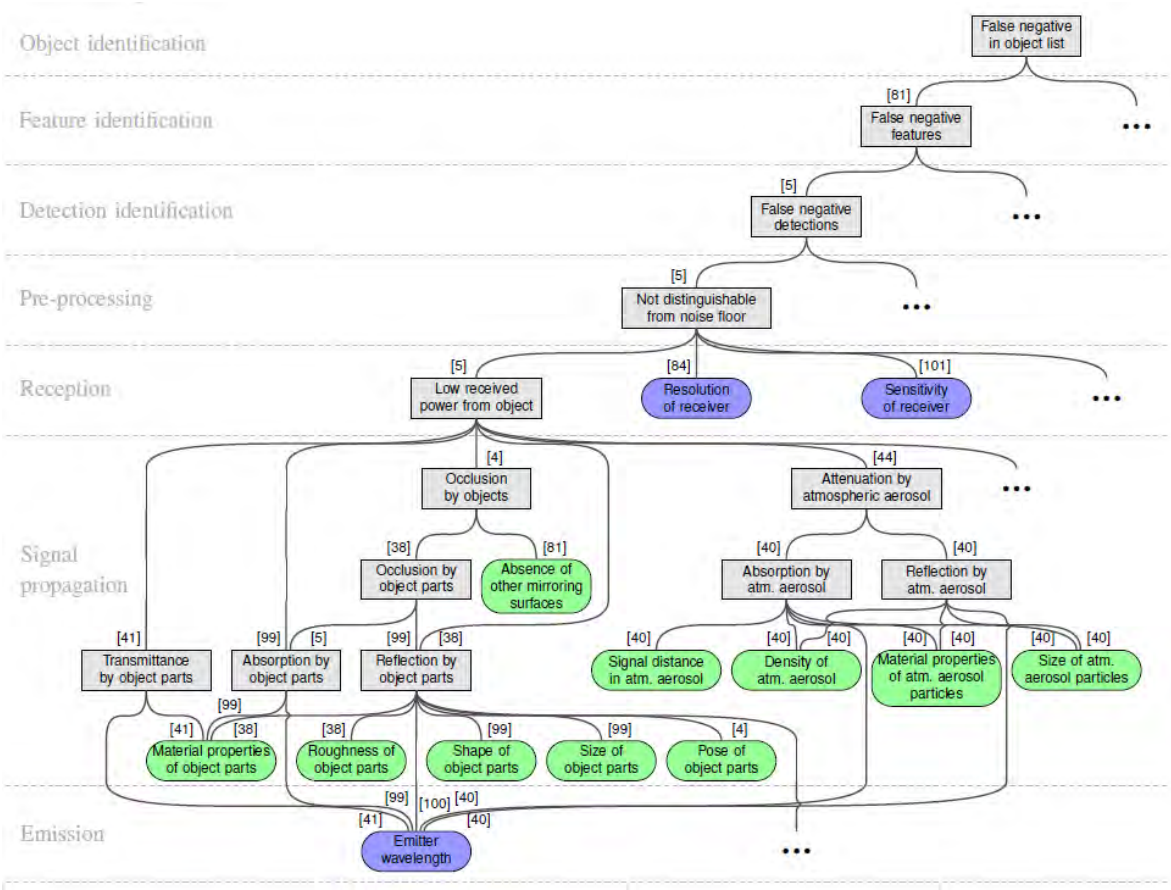


Abbildung 51: Exemplarischer Perception Sensor Collaborative Effect and Cause Tree für aktive Umfeldsensoren nach (Linnhoff, et al., 2021).

Nach der Sammlung der Effekte und Zusammenhänge werden die Wirkketten durch Experten für das jeweilige Simulationsvorhaben in der Methode Cause, Effect, and Phenomenon Relevance Analysis (CEPRA) evaluiert. Die Vorgehensweise wurde dabei an das schon etablierte Verfahren der Failure Mode and Effects Analysis (FMEA) angelehnt. Hierfür werden zunächst die Wirkketten in Bottom-up-Richtung aus PerCollECT für die jeweilige Sensortechnologie automatisiert extrahiert. Jede Wirkkette ergibt dabei eine Zeile der Tabelle, wie Abbildung 52 exemplarisch zeigt. Nachdem die Phänomene, Wirkketten und Ursachen aufgetragen wurden, muss der Testingenieur, welche/r die Relevanz der Wirkketten für das Aufstellen der Anforderungen an das Modell bestimmen will, nach (VDAQMC, 2019) zwei Experten konsultieren:

1. Ein Experte für die zu simulierende Sensorik, die durch das zu spezifizierende Modell abgebildet werden soll, füllt die Spalte für die Auftretenswahrscheinlichkeit (Occurence (O)) aus. Die Werte zwischen 1 und 10 repräsentieren dabei die Feststellungen zwischen „Kann nicht auftreten“ und „Extrem hohe oder nicht bestimmbare Auftretenswahrscheinlichkeit“.
2. Ein Experte für das zu testende System (SuT) füllt die Spalte für den Einfluss (Impact (I)) aus. Die Werte zwischen 1 und 10 repräsentieren dabei die Feststellungen zwischen „sehr gering“ und „beeinträchtigt das sichere Verhalten des Fahrzeugs/anderer Fahrzeuge, oder sogar die Gesundheit von Verkehrsteilnehmern“.

Phenomenon (P)	**Effect chain (EC) of phenomenon**	**Causes of effect chains**	**P&EC occurrence in ODD*** (O, filled by sensor expert)		**P&EC impact on SUT in ODD*** (I, filled by SUT expert)		**Relevance of P&EC**
			$[1, 10]$	Rationale	$[1, 10]$	Rationale	$O + I$
False negative in object list	→ FN features → FN detections → Not dist. from noise floor → Low rec. power from object → Occlusion by objects → Occlusion by object parts → Reflection by object parts	• Materials of reflect. obj. parts • Roughness of reflect. obj. parts • Shapes of reflect. obj. parts • Size of reflect. obj. parts • Emitter wavel.**	*4*	*FN objects caused by occluding reflecting objects occurs rarely in a front radar on a highway, because of multi-path propagation.*	*6*	*FN obj. occurring because of occlusion in a front radar have a moderate impact because mainly only direct neighbor objs. considered.*	***10***
	→ FN features → FN detections → Not dist. from noise floor → Low rec. power from object → Reflection by object parts		*2*	*FN objects caused by compl. away-reflecting obj. cannot be ruled out, but are not expected on highway.*	*9*	*FN objects occurring in a front radar have a very high impact on a highway pilot.*	***11***
	→ FN features → FN detections → Not dist. from noise floor → Low rec. power from object → Attenuation by atm. aerosol → Absorption by atm. aerosol	• Signal dist. in atm. aerosol • ... • Emitter wavel.**	*3*	*FN objects caused by completely absorbing atmospheric aerosol occur only in harsh weather in a front radar on a highway.*	*5*	*FN objects occuring in harsh weather conditions may be covered by safety concept with a moderate impact on the highway pilot.*	***8***
	•••						
•••							

Legend: Normal font: Automatically generated content from PerCollECT after sensor output definition; *Italic: Expert knowledge needed*
*Operational Design Domain (ODD) must be defined beforehand (here: a German highway with all its elements for a highway pilot as SUT).
**These causes are design parameters by the SUT (here: a highway pilot's radar at the front center) and must be defined beforehand.

Abbildung 52: Exemplarische Cause, Effect, and Phenomenon Relevance Analysis (CEPRA) nach (Linnhoff, et al., 2021)

Für Experten gilt, dass die Zahlenwerte für die Nachvollziehbarkeit mit einer kurzen prägnanten Begründung zu versehen sind. Ebenfalls ist es essenziell, dass zunächst die Operational Design Domain (ODD) vollständig beschrieben sein muss, um die entsprechenden Informationen in die jeweiligen Spalten eintragen zu können. Dazu gibt es beim ASAM mit *ASAM OpenODD* bereits Bestrebungen, die ODD-Definition zu standardisieren. Am Ende des Prozesses können für jede Zeile bzw. Wirkkette Relevanzwerte zwischen 2 und 20 durch Addition der durch die jeweiligen Experten bestimmten Werte für *O* und *I* berechnet werden. Mit diesen Werten können Testingenieure dann anschließend entscheiden, welche Wirkketten im Modell abgebildet werden sollen und in welcher Güte diese benötigt werden.
Bei der Definition von Effekten und Wirkketten, die in einem Modell abgebildet werden sollen, werden auch explizit Effekte von der Modellierung ausgeschlossen. Dadurch werden Modellierungsgrenzen klar definiert. Die Grenzen sind und a. durch die Systemgrenzen aus der

ODD beschrieben oder wurden durch die Expertenevaluation für den Simulation-Use-Case als irrelevant deklariert. Die Einhaltung der definierten Anforderungen muss dann durch gezielte Verifikationen des Modells überprüft werden. Bei der Verifikation werden gegebenenfalls. Abweichungen als weitere Simulationsgrenzen identifiziert.

2.1.3.4.2 Tool Confidence Level

Am Beispiel eines gedachten Tools genannt „TESTY V3.0“, das in ähnlicher Form nur als Proof-of-Concept im Rahmen des Projekts SET Level existiert, wurden Aspekte der Tool Qualifizierung durchgespielt. Das Ergebnis mündete in einem Leitfaden zur Toolqualifizierung. Nachfolgend befindet sich eine kurze Zusammenfassung der Ergebnisse.
TESTY ist ein einfaches Testautomatisierungswerkzeug. Es ist ohne Anpassung für ein sehr breites Spektrum von Testanwendungen verwendbar. Der Hauptverwendungszweck von TESTY ist die automatisierte Abarbeitung von Testfällen und deren Testschritten, deren Reihenfolge und Parametrierung in einer XML-basierten Testspecification als Schlüsselwörter mit Parametern vorgegeben sind. Die Testschritte selbst sind in Form von Skripten implementiert und nicht Teil von TESTY. Eine entsprechende Tooldemonstration (siehe https://www.youtube.com/watch?v=nOEA95-YGEE) wurde unter dem Thema „Test Case Specification“ erstellt.

Abbildung 53 zeigt den grundsätzlichen Arbeitsfluss für die Testausführung mit TESTY, den Vorgängerschritt Testvorbereitung und -entwicklung und den nachfolgenden Schritt der Test Auswertung. Anwendungsfall ist z. B. der automatisierte „Nightly Test“ von ECU SW Versionen als SuT. Für die Eingangsdaten und die Implementierung der Test Steps wurde ein V&V Prozess gefordert und der Simulator sollte auch die passende Toolqualifizierung besitzen.

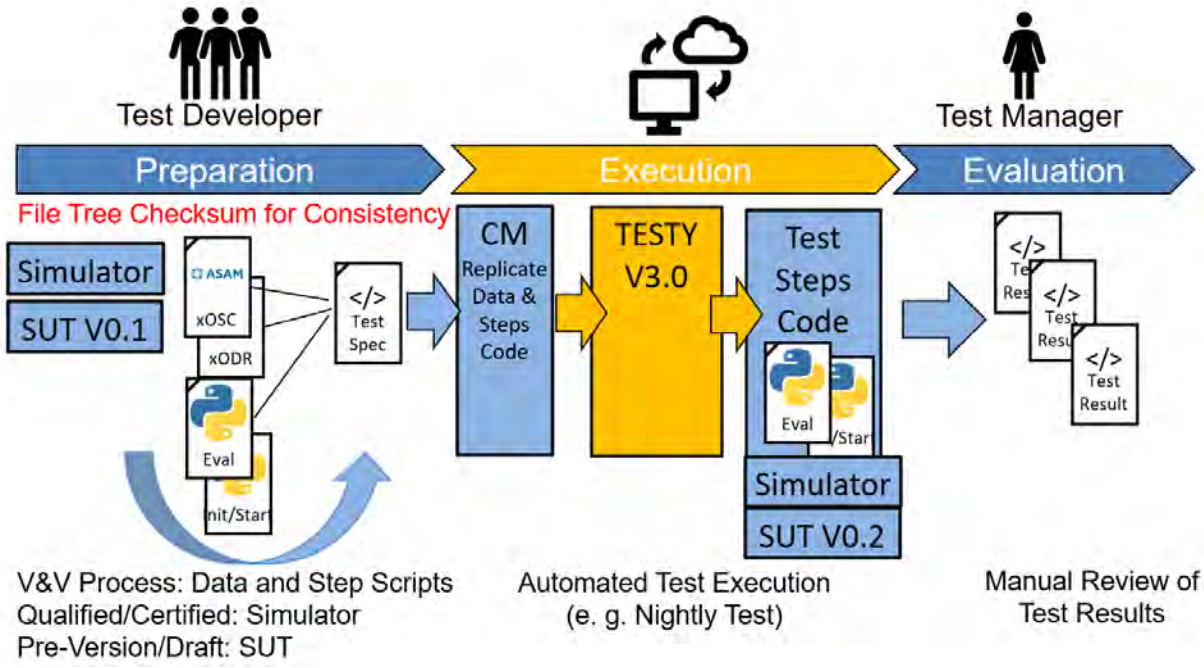


Abbildung 53: Arbeitsfluss für die Testausführung mit TESTY 3.0

Das Vorgehen erfolgte rein exemplarisch. Das TCL von TESTY 3.0 würde demnach bei 2 liegen und eine Tool Qualifizierung ist für den skizzierten Einsatzbereich erforderlich.
Da TESTY nur als Proof-of-Concept im Rahmen des Projekts SET Level existiert, wäre aufgrund der unklaren Verantwortlichkeiten und des fehlenden Entwicklungsprozesses eine

Toolqualifizierung zum Scheitern verurteilt. Tabelle 3 differenziert dies nach den „Fit for Purpose" Kriterien nach (Tüv Süd, 2013)

Tabelle 3: Bewertung der Erfolgschancen für die Toolqualifizierung von TESTY

Kategorie	Inhalt	Bewertung für TESTY V3.0
Validation		
	Fokus auf Hauptanwendungsfälle	Beschreibung existiert
	Assessment durch unabhängige Organisation	nicht vorhanden
Evaluation of the development process applied for the tool		
	Assessment durch unabhängige Organisation	kein definierter Entwicklungsprozess vorhanden
Increased confidence from use		
	Handling der Bug Reports	nicht vorhanden
	Lessons learned (Regression Tests, Verbesserungsprozess, Best Practices)	nicht vorhanden
	Beziehung zu Kunden	nicht vorhanden
Customer information and related processes		
	Analyse der bekannten Fehler (safety relevance)	Analyse der potentiell möglichen Fehler für Klassifizierung begonnen, vorhandene Fehler im Tool nicht bekannt
	Information des Kunden	nicht vorhanden
	Fehlerbehebungsprozess	nicht vorhanden
Safety manual		
	Teil des Zertifikats Beschreibt den Scope des Zertifikats Bietet Mehrwert für den Anwender	Safety manual nicht vorhanden, Ansätze dazu sind beschrieben: Informationen zur Klassifizierung, mögliche Anwendungsfälle (Features), Liste der Risiken und Gegenmaßnahmen
		Ergebnis: Tool Qualifizierung würde nicht erfolgreich sein!

2.1.3.4.3 Modellintegration und -kopplung

Automatisierte und autonome Fahrsysteme werden nicht nur als Möglichkeit betrachtet das Reisen komfortabler, sondern auch sicherer zu machen. Um dies zu realisieren und automatisierte und autonome Fahrsysteme auf den Markt zu bringen, ist es daher sehr wichtig, deren sicheren Betrieb zu gewährleisten. Dies ist keine leichte Aufgabe, da nicht nur die Fahrsysteme selbst hochkomplex sind, sondern auch der Input, den sie erhalten (die Umwelt). Für ihre Validierung und Verifizierung müssen wir sie also nicht nur sicher entwickeln, sondern auch ausgiebig getestet werden. Die Anzahl der Testkilometer, die für eine statistische Verifikation dieser Systeme notwendig sind, beläuft sich (je nach Annahmen/Unfallart) auf mehrere hundert Millionen Kilometer (Wachenfeld, et al., 2015). Zum Vergleich: Alle Straßen in den USA machen nur 6,59 Millionen Kilometer aus (CIA, 2019). Diese Tests müssten mit jedem neu entwickelten oder auch nur leicht modifizierten automatisierten Fahrsystem durchgeführt werden.

Ein möglicher Ausweg aus diesem Dilemma ist der Einsatz von simulationsbasierten Methoden. Um jedoch Aussagen aus der Simulation zu erhalten, die in einer Sicherheitsargumentation verwendet werden können, muss Vertrauen darin bestehen, dass die Simulation tatsächlich realistische Ergebnisse liefert. Dazu gehört nicht nur die Validierung der einzelnen

verwendeten Modelle, sondern auch eine Betrachtung der Methoden mit Hilfe derer diese Modelle miteinander verbunden sind.
Um im Rahmen des Projekts SET Level Gütekriterien für die Kopplung von Simulationsmodellen entwickeln zu können, wurde zunächst eine Übersicht über existierende Kopplungsmechanismen, derzeit angelegte Gütekriterien sowie ein generelles Verständnis relevanter Fehlereffekte benötigt. Dazu gehören beispielsweise Verzögerungen oder numerische Fehler, welche bei der Kopplung unterschiedlicher Modelle bzw. Modellklassen auftreten können.
Ziel war es somit, eine im Projektkonsortium abgestimmte Basis für die weitere Entwicklung von Gütekriterien für die Kopplung von Simulationsmodellen zu schaffen.

Simulation
Simulation ist ein etabliertes Werkzeug zur Unterstützung des Entwicklungs- und Testprozesses komplexer Systeme. Dabei wird das abstrahierte Modell eines Systems (oder Modelle mehrerer Systeme) in einer Laufzeitumgebung ausgeführt, indem die in der Realität erwarteten Werte an den Eingangsschnittstellen des Modells angelegt und die resultierenden Ausgabewerte beobachtet werden. Die Modelle können beispielsweise als Differentialgleichungssysteme oder diskrete Automaten dargestellt werden. Eine Laufzeitumgebung ist dabei ein Softwaresystem, das die Modellausführung ermöglicht und Modellgleichungen löst. Eine Simulation wird komplexer, wenn des Weiteren Eingangswerte anderer Modelle berücksichtigt oder die Ausgangswerte rückgekoppelt bzw. anderen Modellen zur Verfügung gestellt werden sollen.

Simulationsmethoden
Grundsätzlich werden in diesem Dokument verschiedene Simulationsmethoden unterschieden, die nachfolgend aufgelistet sind:

- Zeitdiskrete Simulation (engl. „Discrete-Time Simulation“)
  Der Zustandsübergang und die Änderung der Zustandsgrößen erfolgt nur zu definierten Zeitpunkten. Dazwischen bleiben die Zustandsgrößen konstant.
  - Ereignisdiskrete Simulation (engl. „Discrete-Event Simulation“)
    Der Simulationsfortschritt erfolgt durch die Abarbeitung einer Liste von diskreten Ereignissen, welche z. B. durch einen endlichen Automaten beschrieben sein kann. Es werden somit die Ereignisse simuliert und nicht die zwischen den Ereignissen liegende Zeit.
  - Kontinuierliche Simulation (engl. „Continuous-Time Simulation“)
    Der Simulationsfortschritt und die Änderung der Zustandsvariablen ist kontinuierlich beschrieben und geschieht in Abhängigkeit vom aktuellen Zustand. Bei der Beschreibung des Systemverhaltens finden lineare bzw. nichtlineare Gleichungssysteme oder auch Differenzialgleichungen Verwendung, die von numerischen Solvern gelöst werden.
  - Hybride Simulation (engl. „Hybrid Simulation“)
    Die hybride Simulation beschreibt sowohl zeitlich diskretes als auch kontinuierliches Systemverhalten und wird oft dort eingesetzt, wo kontinuierliche Modelle auf Diskontinuitäten treffen oder auch bei einer diskreten Steuerung von kontinuierlichen Prozessen. Grundsätzlich kann zwischen Hybrid-Discrete-Event Simulation (jede kontinuierliche Komponente wird als zeitdiskrete Komponente verpackt) und Hybrid Continuous-Time (jede zeitdiskrete Komponente wird als kontinuierliche Komponente verpackt) unterschieden werden.
  - Echtzeitsimulation (engl. „Real-time Simulation“)
    Je nach Rechenleistung können Simulationsausführungen durchaus langsamer oder

auch schneller als in Echtzeit durchgeführt werden. Soll jedoch z. B. in einer Hardware-in-the-Loop (HiL) Simulation eine Kopplung mit realer Hardware erfolgen, muss auch die Simulation in Echtzeit erfolgen, so dass eine synchrone Wechselwirkung zwischen der virtuellen und der realen Welt erfolgt (z. B. durch die Übereinstimmung von Modellzeit und realer Zeit an Synchronisationspunkten).

Idealerweise werden Simulationen durchgeführt, indem ein einziges Modell in einer Laufzeitumgebung verwendet wird, da in diesem Fall keine Komplexität erhöhende Kopplung notwendig ist. Dies ist jedoch auf Grund der Vielfalt der zu berücksichtigen Effekte in den im Projekt SET Level betrachteten Anwendungsfällen nicht möglich. Daher müssen vielfältige Modelle, beispielsweise Sensormodelle, 3-D-Weltmodelle oder Fahrdynamikmodelle, verwendet werden, die teilweise unterschiedliche Laufzeitumgebungen erfordern, was die Komplexität der Simulation steigert.

Die koordinierte Ausführung von zwei oder mehreren Modellen, die sich in ihren Laufzeitumgebungen unterscheiden und gegebenenfalls auf verschiedenen Modellierungsparadigmen basieren, wird auch als Co-Simulation bezeichnet. Modelle in einem Co-Simulationssystem können daher sowohl unabhängig voneinander entwickelt als auch implementiert werden. Um die Modelle respektive Laufzeitumgebungen gekoppelt ausführen zu können, sind geeignete Schnittstellen für den Austausch der Daten notwendig. Eine Herausforderung beim Aufbau und der Ausführung einer Co-Simulation besteht nicht nur in der Definition von passenden Datentypen zum Austausch von Informationen, sondern auch in der Definition eines geeigneten, möglicherweise abstrakten Zeitmodells, so dass sich die unterschiedlichen Laufzeitumgebungen bei ihrer Ausführung nicht gegenseitig beeinträchtigen.

Fehlerquellen bei der Durchführung von Simulationen

Während der Durchführung von Simulationen können verschiedenste Abweichungen zum realen System entstehen. Einige dieser Abweichungen sind gewollt und bekannt (z. B. durch die Wahl der Modellierungstiefe), andere Abweichungen hingegen nicht. Neben Modellierungsfehlern, Fehlern im Simulator, Fehler bei der Konfiguration des Simulators und numerischen Fehlern sind Kopplungen von Simulationsmodellen und Simulatoren potentielle Fehlerquellen. Daher ist es grundsätzlich wünschenswert auf Kopplungen zu verzichten. Allerdings werden Kopplungen häufig z. B. auf Grund von Modularitätsanforderungen oder IP Schutz notwendig.

Je nach Art der Kopplung und Art der Simulation können unterschiedliche Effekte auftreten. Geschieht die Kopplung synchron (d. h. ein Datenaustausch zwischen allen an der Simulation beteiligten Komponenten findet nur zu definierten Zeitpunkten statt), so kann es beispielsweise zu einer Verzögerung (engl. delay) im Datenaustausch kommen oder es können algebraische Schleifen auftreten. Aber auch bei einer asynchronen Kopplung kann es z. B. zu Datenverlusten durch Pufferüberläufe kommen. Grundsätzlich wird zwischen lokalen Fehlern (innerhalb eines Modells oder Simulators) und globalen Fehlern (betreffen die Gesamtsimulation) unterschieden. Lokale Fehler können zum globalen Fehler beitragen. Über das Ausmaß dieses Beitrages kann keine allgemeingültige Aussage getroffen werden. Bevor eine detaillierte Betrachtung der potentiell auftretenden Fehlerquellen und Effekte in vorgenommen wird, werden unterschiedliche Ansätze und Mechanismen für die Kopplung von Simulatoren und Simulationsmodellen vorgenommen.

Kopplungsansätze und Kopplungsklassen

In (Geimer, et al., 2006) wird eine Begriffsvereinheitlichung im Themengebiet der Kopplung und Kopplungsansätze für Simulation gegeben, welche an dieser Stelle aufgegriffen wird und als Ausgangsbasis dient. Zur Einordnung der verschiedenen Ansätze wird zwischen einer

geschlossenen und einer verteilten Modellbildung unterschieden. Bei einer geschlossenen Modellbildung erfolgt die Modellierung mit einem Modellierungswerkzeug während bei einer verteilten Modellbildung unterschiedliche Modellierungswerkzeuge zur Realisierung des Modells Anwendung finden. Des Weiteren wird zwischen einer geschlossenen und einer verteilten Simulation unterschieden. Bei der geschlossenen Simulation erfolgt die Ausführung des Simulationsmodells in einer einzigen Laufzeitumgebung wobei bei einer verteilten Simulation mehrere Laufzeitumgebungen genutzt werden. Die Laufzeitumgebungen lösen dabei unter anderem die Modellgleichungen der verwendeten Simulationsmodelle und werden daher im weiteren Verlauf dieses Abschnitts auch Solver genannt.
Modellierungswerkzeuge unterscheiden sich durch die jeweils unterstützen Modellklassen. Eine Modellklasse wird dabei durch das hinterlegte „Model of Computation“ (MoC) (Böde, et al., 2017) definiert. Das MoC definiert die Zeitdarstellung und die Semantik der Kommunikation und Synchronisation zwischen den einzelnen Komponenten in der Implementierung des Modells. Ein MoC ist dabei idealerweise implementierungsunabhängig, kompositionell und analysierbar.
In Abbildung 54 sind beispielhaft verschiedene MoC abgebildet. Bei „Continous Time Models“ wird das Verhalten normalerweise durch Gleichungen über reelle Zahlen ausgedrückt. Dabei werden z. B. Differentialgleichungen verwendet, um interne Feedback-Schleifen auszudrükken. Für die Simulation dieser MoCs werden numerische Solver benötigt. Sprachen und Modellierungswerkezeuge, die zeitkontinuierliche MoCs unterstützen, sind z. B. Matlab/Simulink, Modelica und SystemC-AMS. Bei „Discrete-Time Models“ sind dagegen Ereignisse mit jeweils einem diskreten Zeitmoment verbunden. Zeitdiskrete und insbesondere ereignisdiskrete Modelle werden häufig für die Hardwaresimulation verwendet. Beispiele für diskrete Ereignissprachen und zugehörige Simulatoren sind VHDL, Verilog und SystemC. „Synchronous Models“ sind eine Abstraktion von diskreten MoC. Die Zeit wird hierbei durch eine diskrete Menge von ganzen oder natürlichen Zahlen dargestellt, wobei die Zeiteinheit keine physikali-

sche Zeiteinheit, sondern eine abstrakte Interpretation der Zeit ist. In synchronen Modellen kann z. B. das Auftreten von Ereignissen nicht genau definiert sein, sondern ist durch den Beginn und das Ende von Zeitfenstern (oder Zeitschritten) begrenzt. Sprachen und Modellierungswerkzeuge, die synchrone zeitdiskrete MoCs unterstützen, sind z. B. Matlab/Simulink

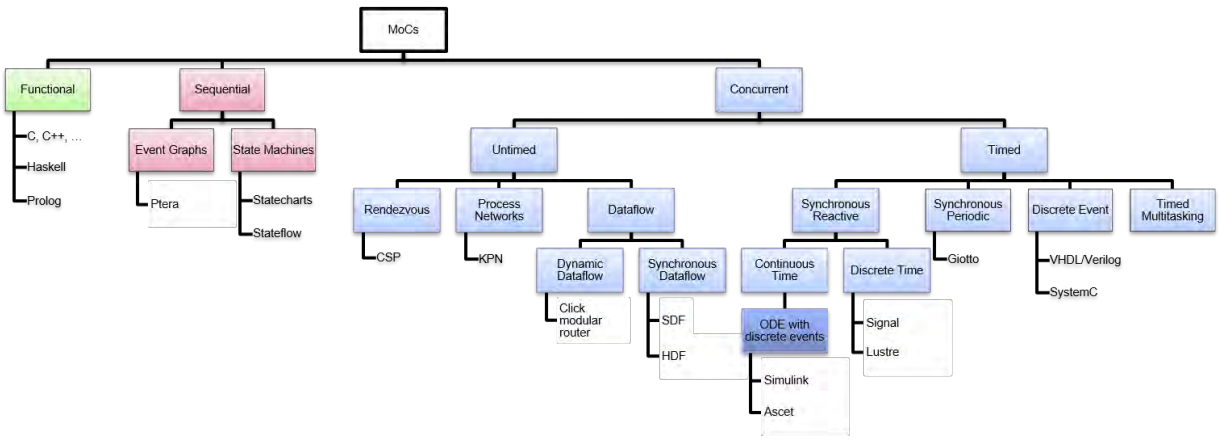


Abbildung 54: Klassifizierung von Models of Computation (MoC) nach (Jantsch, 2004)

und SystemC-AMS. „Untimed Sequential“ MoCs werden beispielsweise durch sequentielle Programmiersprachen wie Java, C oder C++ realisiert, welche es ermöglichen komplexe

Funktionalitäten zu beschreiben. Des Weiteren gibt es mit State Machines (z. B. Stateflow und Statemate) die Möglichkeit maßgeblich kontrollflussdominierte Systeme zu erstellen.

Kopplungsklassen

In Abbildung 55 sind angelehnt an (Geimer, et al., 2006) verschiedene Kopplungsklassen dargestellt. Neben der Ausführung eines Modells einer Modelklasse/MoC mit Hilfe eines Solvers, bei der es keine Kopplung gibt („Klassische Simulation“), gibt es drei unterschiedliche Klassen von Kopplung, die im weiteren Verlauf dieses Abschnitts betrachtet werden, da alle drei Kopplungsklassen im Rahmen des Projekts SET Level Verwendung finden können. Diese sind im Folgenden aufgeführt.

- **Klasse „Modellkopplung“**: Das Gesamtsystem ist durch mehrere Teilmodelle unterschiedlicher MoC modelliert und wird mit Hilfe eines Simulationskerns ausgeführt und durch nur einen Solver gelöst. Dies bedingt, dass entweder die verschiedenen MoC semantikerhaltend ineinander überführt werden können (z. B. durch Export aus dem Modellierungswerkzeug) oder das der Solver in der Lage ist, die verschiedenen MoC zu lösen.
- **Klasse „Simulatorkopplung“**: Das Gesamtsystem ist durch mehrere Modelle eines identischen MoC modelliert, welche auf mehreren Simulationskernen ausgeführt werden. Dies bedingt eine Synchronisation der Simulationskerne, so dass die Ausführungssemantik der MoC nicht verletzt wird.
- **Klasse „Co-Simulation“:** Das Gesamtsystem ist durch mehrere Modelle unterschiedlicher MoC modelliert welche wiederum auf mehreren Simulationskernen ausgeführt werden. Neben der Synchronisation bzw. Orchestrierung der einzelnen Simulationskerne muss bei der Co-Simulation die Kompatibilität der MoC sichergestellt werden.

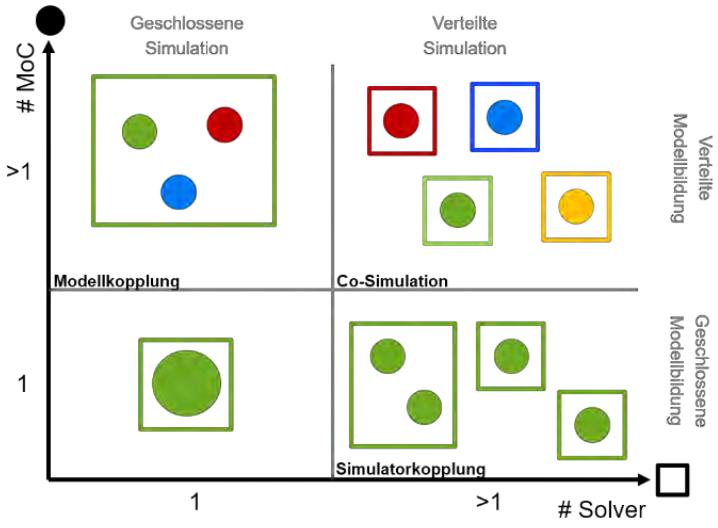


Abbildung 55: Kopplungklassen, angelehnt an (Geimer, et al., 2006)

Modellkopplung

Bei der Modellkopplung werden mehrere Teilmodelle zu einem Gesamtmodell zusammenführt. Die Modelle werden getrennt voneinander unter Verwendung unterschiedlicher Entwicklungswerkzeuge und MoC modelliert und dann mittels Modellexport bzw. Modellimport zu einem Gesamtmodell zusammengeführt. Dabei ist sicherzustellen, dass entweder die MoC semantikerhaltend ineinander überführt werden können, oder aber der zum Einsatz kommende Solver zum Lösen der Modellgleichungen unterschiedliche MoC handhaben kann.

Simulatorkopplung und Co-Simulation

Wenn mehrere Simulationskerne miteinander gekoppelt werden sollen, muss zwischen den Simulationskernen zu definierten Zeitpunkten ein koordinierter Datenaustausch stattfinden. Da die verwendeten Simulationsmodelle z. B. mit Hilfe unterschiedlicher MoC realisiert sind oder aber die Solver räumlich getrennt auf unterschiedlich performanter Hardware ausgeführt werden, sind bei einer Kopplung verschiedene Aspekte zu berücksichtigen, die im Folgenden beschrieben sind.

- **Zeitlicher Aspekt** (im Folgenden auch Kopplungsstrategie): Nur wenn die beteiligten Simulationsmodelle ihre Berechnungsdaten zu den korrekten Zeitpunkten vorliegen haben, kann eine korrekte Gesamtfunktionalität realisiert werden.
- **Räumlicher Aspekt**: Um den ausgeführten Simulationsmodellen ihre Berechnungsdaten zukommen zu lassen, ist ein effizientes Transportmedium notwendig (z. B. shared memory, verteilt über Netzwerk etc.).
- **Inhaltlicher Aspekt** (Interface/Schnittstelle): Damit ein korrekter Datenaustausch durchgeführt werden kann, müssen die an der Simulation beteiligten Komponenten über entsprechende Schnittstellen verfügen.

Je mehr Komponenten an einer Simulation beteiligt sind, desto mehr Aufwand bedeutet eine Kopplung dieser Komponenten.

Für den Datenaustausch zwischen den Simulatoren ist im Normalfall eine Middleware zuständig, die eine Menge von (standardisierten) Interfaces bereitstellt (siehe Abbildung 56). Auf diese Weise kann ein verbundener Simulator indirekt Daten mit allen anderen angebundenen Simulatoren austauschen. Dies setzt voraus, dass die Schnittstellen für alle integrierten Simulatoren standardisiert sind. Dadurch gibt es nur ein Interface, dass für jeden Simulator implementiert werden muss, um mit allen anderen Simulatoren zu kommunizieren (im Gegensatz zur individuellen Kopplung).

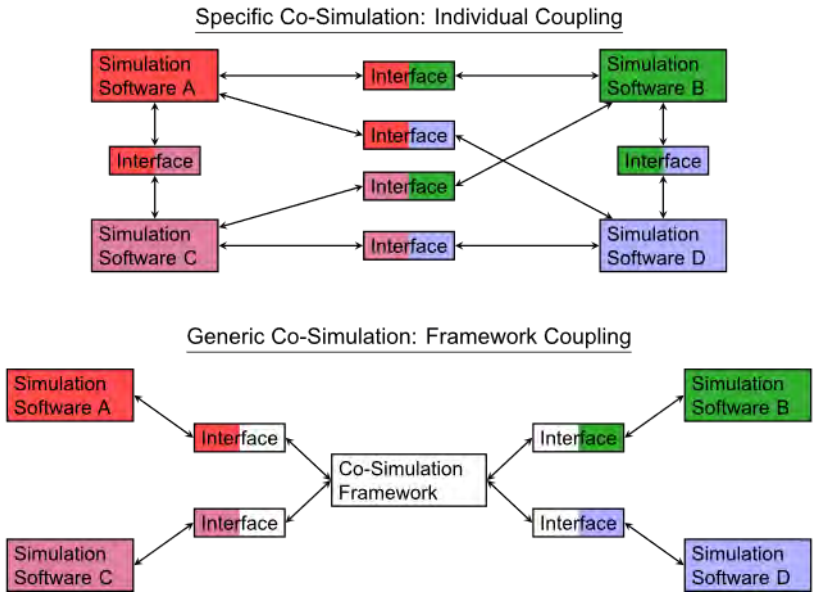


Abbildung 56: Gegenüberstellung einer individuellen Kopplung und einer Frameworkkopplung

Das Framework übernimmt das Starten und Beenden von Prozessen, sowie die Initialisierung und die Ablaufsteuerung. Der dabei benötigte Datenaustausch kann anhand von verschiedenen Strategien durchgeführt werden, die in Abbildung 56 überblickshaft dargestellt sind.

Eine Kopplung kann z. B. mit Hilfe von Master-Slave Algorithmen gesteuert werden. Im Allgemeinen ist der Master dafür zuständig, Kommunikationszeitpunkte zu definieren und die Slaves (Simulatoren) zur Simulation aufzufordern. Er fragt die Ausgangswerte der Slaves ab, verarbeitet diese und stellt das Ergebnis als Eingangsgröße den anderen Slaves bereit.
In einem Master können mehrere Master-Algorithmen vorhanden sein. Diese rufen die Slaves entsprechend ihrer Priorität auf. Ausgehend vom Algorithmus kann die Kommunikationsschrittweite variabel oder fest sein und Zyklen einfach oder mehrfach auf verschiedene Weisen durchlaufen werden. In der Konfigurationsdatei, die der Master liest, ist festgelegt, welche/r dieser Algorithmen genutzt wird/werden.

Kopplungsstrategien bei Simulatorkopplung und Co-Simulation
Grundsätzlich kann zwischen verschiedenen Kopplungsstrategien und a. anhand des Kopplungszeitpunkts (synchron oder asynchron) sowie der Integrationsprozesse (parallel oder sequentiell) unterscheiden werden. Im Folgenden werden beispielhaft einige Kopplungsstrategien vorgestellt (siehe auch Abbildung 57).
Im allgemeinen Schema (a) läuft eine verteilte Simulation, die aus mehreren Simulatoren (im Folgenden auch Subsystemen) besteht, so ab, dass zuerst die Eingabegrößen der Subsysteme während eines Makroschrittes approximiert werden. Ein Makroschritt H markiert einen Koppelzeitpunkt, also die Zeit, zu der die Subsysteme Ein- oder Ausgabewerte austauschen. Ein Makroschritt besteht aus mehreren Mikroschritten. Mikroschritte verschiedener Subsysteme sind unabhängig voneinander. Nach der Integration und Ermittlung des Ausgangswertes werden Koppelgrößen zu Koppelzeiten zwischen den Subsystemen ausgetauscht. Diese Koppelgrößen werden wiederum erneut approximiert und für die Integration des anderen Subsystems genutzt.
Im synchronen Schema (g) sind die Koppelzeitpunkte der Subsysteme zum gleichen Simulationszeitpunkt.
Ein synchrones Schema ist das Jacobi-Schema (b). Hier werden zunächst die Eingangswerte durch Extrapolation approximiert. Die beiden Subsysteme integrieren dann parallel. Nach Berechnung der Ausgänge werden diese zum Koppelzeitpunkt gemäß den Koppelbedingungen an das jeweils andere Subsystem übertragen.
Das Gauß-Seidel-Schema (d) ist synchron sequentiell. Somit sind die Koppelzeitpunkte gleich, die einzelnen Subsysteme integrieren jedoch nicht parallel, sondern nacheinander. Dies kann zum Beispiel vorteilhaft sein, wenn das zweite Subsystem aktuelle Ergebnisse des ersten Subsystems benötigt. Der Eingangswert wird somit von Subsystem B zuerst extrapoliert und dann integriert. Nachdem der Ausgangswert berechnet wurde, wird dieser zum nächsten Koppelzeitpunkt an Subsystem A übertragen. Hier wird nun mit der Interpolation dieses Wertes integriert und der nächste Ausgangswert berechnet, der dann an Subsystem A weitergegeben wird.
Weiterhin können Kopplungsstrategien auch überlappend stattfinden (siehe Schema (e)). Es kann über zwei synchrone Makroschritte gerechnet werden, die versetzt beginnen.
Neben synchronen Schemata, gibt es auch asynchrone Schemata.
Bei einer asynchronen sequentiellen Synchronizität (c) laufen die Integrationen nacheinander ab und die Koppelzeitpunkte finden zu verschiedenen Simulationszeiten der Subsysteme statt. Im Gegensatz zum Gauß-Seidel-Schema werden die Eingangswerte vom Subsystem B interpoliert und dann die Eingangswerte vom Subsystem A extrapoliert. Die Wahl zusätzlicher Mikroschritte entfällt hier.
Neben den oben genannten expliziten Schemata gibt es auch implizite Schemata. Darunter fällt das iterative Schema waveform relaxation (f). Hierbei wird zunächst ein Makroschritt nach dem Jacobi- oder dem Gauß-Seidel-Schema durchgeführt. Nach dem Datenaustausch

zwischen den Subsystemen wird jedoch nicht zum nächsten Makroschritt übergegangen, sondern der aktuelle Makroschritt wiederholt, bei dem die Eingänge beider Subsysteme interpoliert werden. Dies geschieht sooft, bis ein Konvergenzkriterium erreicht wird. Dann wird zum nächsten Makroschritt übergegangen.

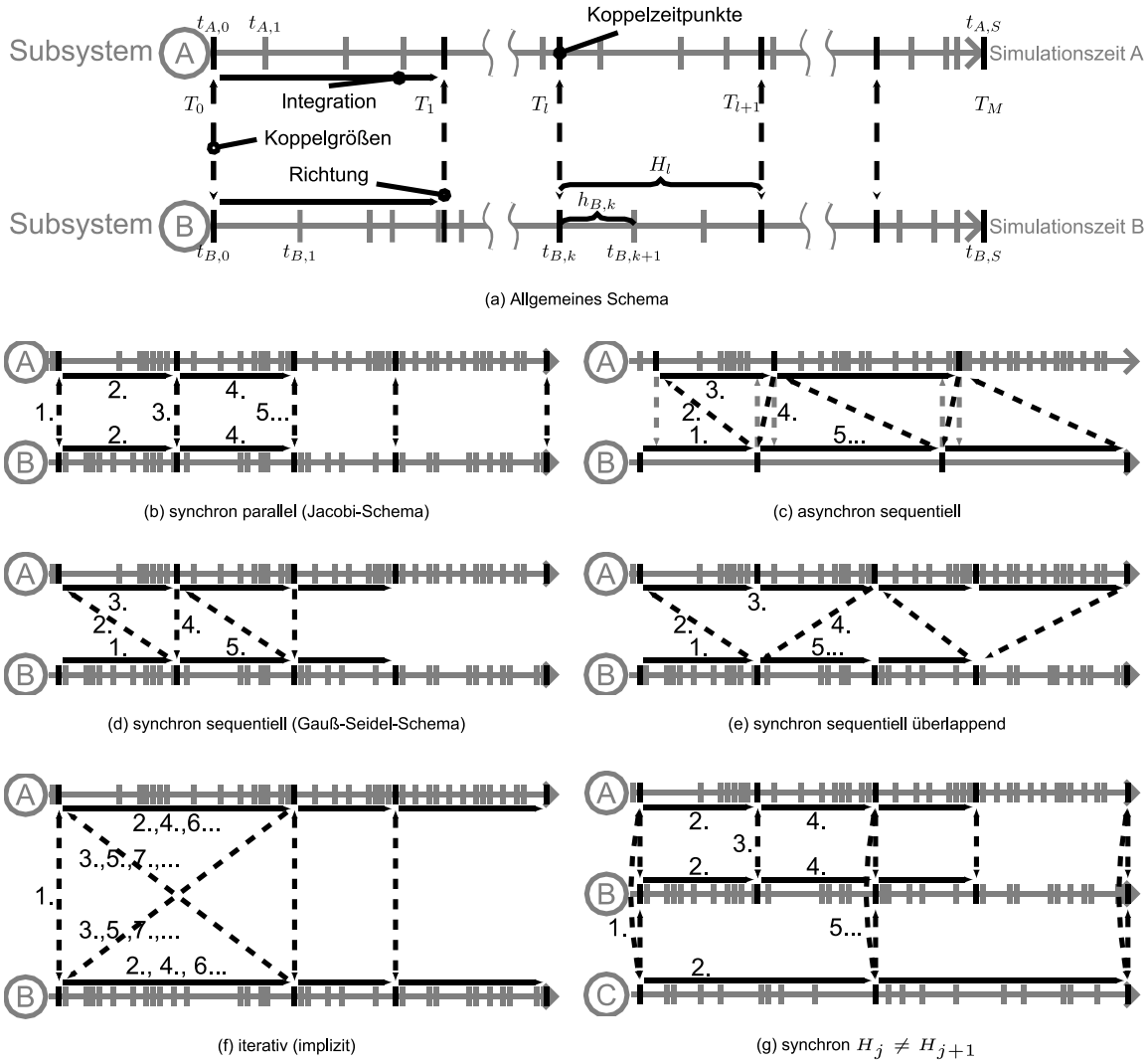


Abbildung 57: Kopplungsstrategien aus (Günther, 2017)

Kopplungsbeispiel aus SET Level

Zur Analyse, Veranschaulichung und Demonstration der Herausforderungen bei der Kopplung von Simulationen wird hier der *Simulation Use Cases* 1 aus dem Projekt SET Level als Praxisbeispiele herangezogen.

***Simulation Use Case* 1: Kritikalitätsanalyse**

Der *Simulation Use Case* 1 (SUC 1) – Kritikalitätsanalyse, beschäftigt sich mit der gezielten Exploration eines Szenarienraums. Im Bereich der Kritikalitätsanalyse im Partnerprojekt V&V Methoden wird basierend auf Expertenwissen eine Liste von Kritikalitätsphänomenen aufgestellt. SET Level steuert an dieser Stelle geeignete Simulationen bei, um Kritikalitätsphänomene zu verfeinern und Wirkzusammenhänge zu plausibilisieren (Neurohr, et al., 2021). In Abbildung 58 ist die Plattformarchitektur des SUC 1 schematisch dargestellt. Das Simulationssystem besteht dabei aus einem Simulationskern, mehreren Agenten für den Umgebungsverkehr, einem EGO Agent sowie zwei Observern zur Messung bzw. Speicherung der Metriken. Bis auf die hochautomatisierte Fahrfunktion (HADF) werden die Simulationsmodelle

und die Observer als FMUs angebunden. Die HADF wird mit Hilfe des Robot Operating Systems (ROS) ausgeführt und über einen FMU-Wrapper an den Simulationskern angebunden. Die Simulationsmodelle und Observer werden im Rahmen des Projekts von verschiedenen Projektpartnern entwickelt. Als Simulationskern wird openPASS verwendet, welches ebenfalls im Rahmen des Projekts stetig weiterentwickelt wird. Als Schnittstelle für den Datenaustausch zwischen den Simulationsmodellen und dem Simulationskern wird das ASAM Open Simulation Interface (OSI) verwendet. Identifizierte notwendige OSI Erweiterungen zur Realisierung der SUCs werden dem ASAM zur Aufnahme in den Standard vorgeschlagen. Die Kontroll- und Evaluationskomponenten, welche sich außerhalb des Simulationssystems befinden und zur Simulationsdurchführung und Simulationsauswertung benötigt werden, werden an dieser Stelle nicht weiter betrachtet.

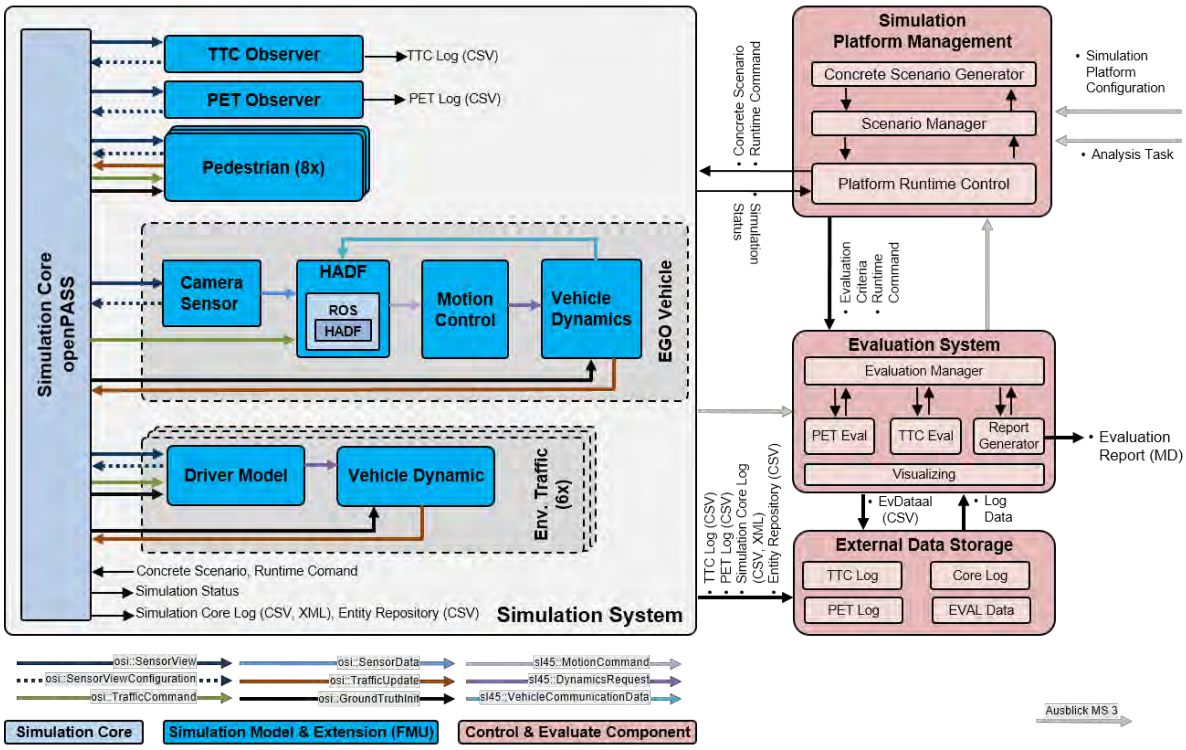


Abbildung 58: Plattformarchitektur des Simulation Use Case 1 (SUC 1) (schematisch dargestellt)

In den folgenden Abschnitten wird erläutert, wie die einzelnen Simulationsmodelle und Observer an den Simulationskern angebunden sind und wie der Datenaustausch realisiert ist.

Anbindung eines Fußgängeragenten an openPASS

In Abbildung 59 ist die Anbindung eines Fußgängersimulationsmodells an den Simulationskern openPASS sowie die Aufrufreihenfolge der beteiligten Komponenten nach erfolgreicher Initialisierung schematisch dargestellt. Das Simulationsmodell des Fußgängers besteht dabei aus einer einzelnen FMU.

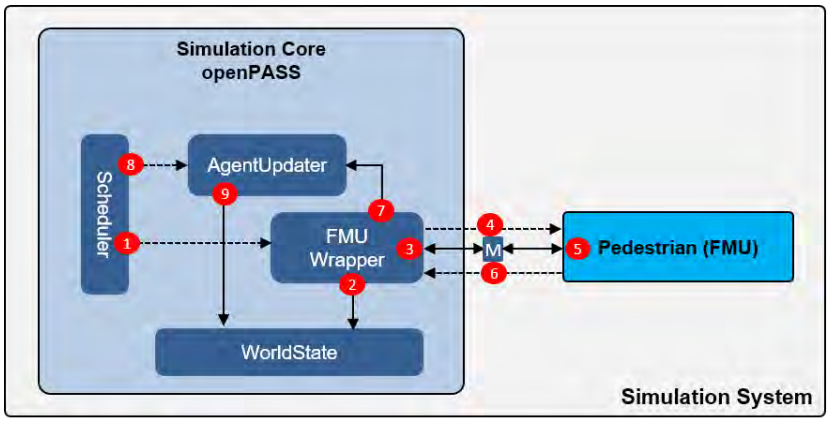


Abbildung 59: Anbindung einer FMU an den Simulationskern openPASS

Innerhalb von openPASS koordiniert ein „Scheduler“ die einzelnen Aktivitäten während der Simulationsausführung. Der aktuelle Zustand der Simulation und aller Agenten wird im „WorldState“ festgehalten. Eine FMU wird in openPASS jeweils über einen „FMU Wrapper“ angebunden und die Aktualisierung des „WorldStates“ erfolgt per „AgentUpdater“. Änderungen im WorldState sind erst nach erfolgreicher Abarbeitung aller Agenten sichtbar.
Nach erfolgreicher Initialisierung aller Komponenten ergibt sich nach dem Simulationsstart folgender Ablauf:

(1) Der Scheduler triggert den „FMU-Wrapper“ per Funktionsaufruf und startet damit die Ausführung.
(2) Der „FMU-Wrapper“ beginnt mit der Ausführung und liest die zur Ausführung der angebundenen FMU benötigten Daten aus dem „WorldState“.
(3) Die Daten aus dem „WorldState“ werden passend für die FMU aufbereitet (z. B. OSI Nachricht) und im „SharedMemory“ abgelegt
(4) Die angebundene FMU (Fußgänger) wird getriggert und dabei die Speicheradresse des „SharedMemory“ mit den Nutzdaten übergeben.
(5) Die angebundene FMU startet mit der Ausführung, liest die Nutzdaten aus dem „SharedMemory“ ein, führt die Berechnungen durch und legt die Ergebnisdaten wieder im „Shared Memory“ ab.
(6) Anschließend übergibt die angebundene FMU die Speicherardresse des „SharedMemory“ mit den Ergebnisdaten an den „FMU-Wrapper“ und beendet die Ausführung.
(7) Der „FMU-Wrapper“ liest die Daten aus dem „SharedMemory“, bereitet diese auf und leitet sie über ein internes openPASS Objekt an den „AgentUpdater“ weiter. Danach beendet der „FMU-Wrapper“ die Ausführung und wartet auf die nächste Triggerung durch den Scheduler.
(8) Der Scheduler triggert als nächstes den „AgentUpdater“.
(9) Der „AgentUpdater“ liest die vom „FMU-Wrapper“ gelieferten Daten aus dem internen Objekt und aktualisiert den „WorldState“.

Anbindung von zwei unabhängigen FMUs

Bei der Anbindung von zwei unabhängigen FMU Simulationsmodellen wie in Abbildung 60 ergibt sich ein ähnliches Bild wie bei der Anbindung eines einzelnen Simulationsmodells. Ein zusätzlicher FMU Wrapper bindet die FMU an und ein zweiter AgentUpdater sorgt für die Aktualisierung des WorldStates. Nachdem in Schritt (9) der AgentUpdater den WorldState aktualisiert hat, triggert der Scheduler den zweiten FMU-Wrapper und ein identischer Ablauf wie bei der Abarbeitung des ersten Simulationsmodells tritt in Gang. Beide FMU-Warpper arbeiten dabei auf einem identischen WorldState, da die durch die AgentUpdater durchgeführten

Änderungen erst nach der Abarbeitung aller angebundenen Simulationsmodelle und somit beim nächsten Simulationsschritt sichtbar werden.

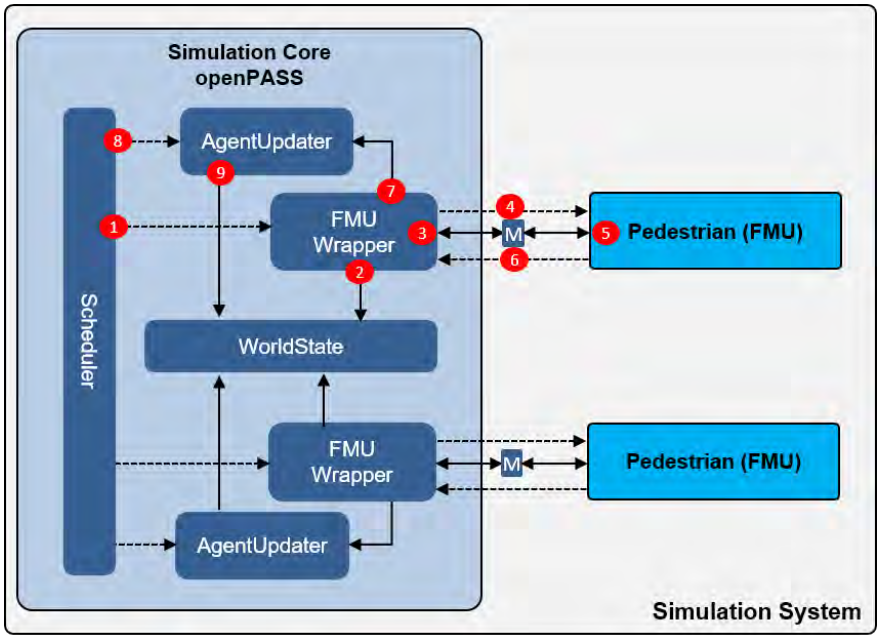


Abbildung 60: Anbindung von zwei unabhängigen Simulationsmodellen

Anbindung der Umgebungsverkehrsmodelle

Die Anbindung der Umgebungsverkehrsmodelle an openPASS wie in Abbildung 61 erfolgt ein wenig anders, da diese Simulationsmodelle durch die Kombination von zwei FMUs realisiert werden welche untereinander Daten austauschen. Die FMU „Vehicle Dynamics“ benötigt von dem „Driver Model“ Informationen welche per OSI Nachricht „sl45::DynamicsRequest“ übertragen wird. Um dieser Anforderung gerecht zu werden, wurde die Architektur und das Konzept zur Anbindung von FMU Simulationsmodellen in openPASS erweitert. Zusätzlich zum „FMU-Wrapper“ und „AgentUpdater“ wurde ein „SSP-Wrapper“ eingeführt. Dieser liest zu Beginn eine SSP Datei mit der Konfiguration der verwendeten Modelle ein und instantiiert für jede FMU einen „FMU-Wrapper“ (siehe Abbildung 61).

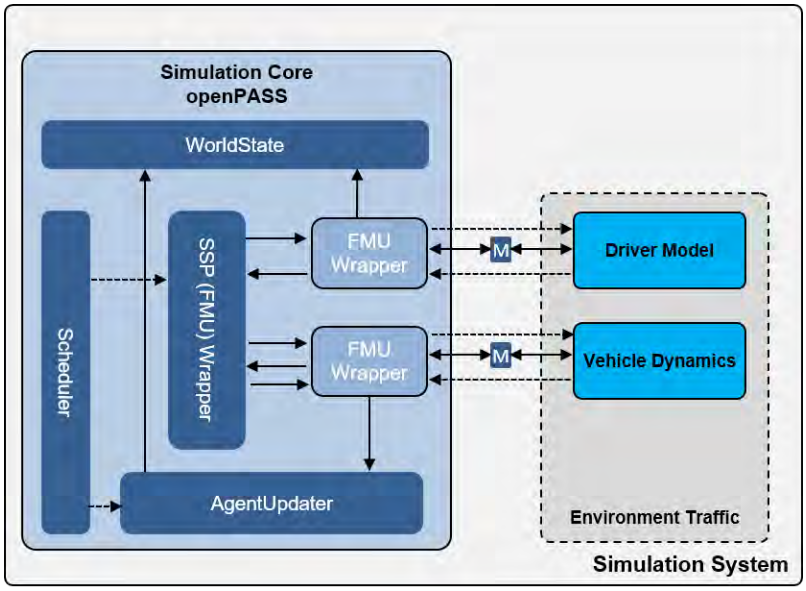


Abbildung 61: Anbindung von Umgebungsverkehrsmodellen

Abbildung 62 beschreibt die Anbindung des Ego-Vehicle.

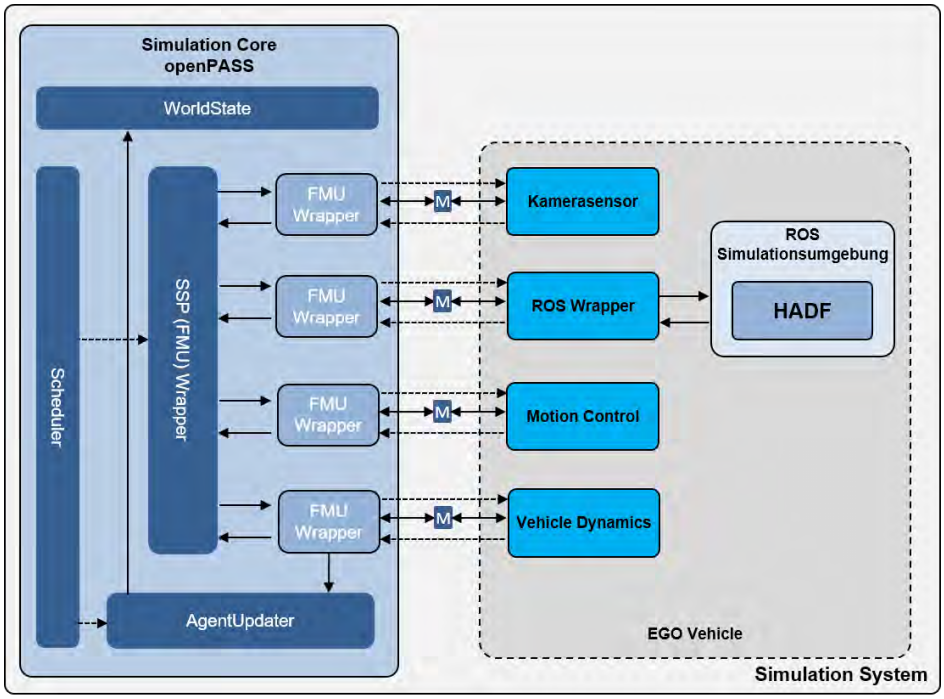


Abbildung 62: Anbindung des Ego-Vehicle

2.1.3.4.4 Zu berücksichtigende Aspekte bei der Kopplung von Simulatoren

In (Gomes, et al., 2017) wird unter anderem auch auf zu berücksichtigende Aspekte bei der Kopplung bei Co-Simulationen und Simulatorkopplung eingegangen. Grob lassen sich diese nach Art der Simulation unterteilen. Dies wird in der folgenden Tabelle 4 dargestellt, die eingetragenen Zahlen werden unter der Tabelle erläutert.

Tabelle 4: Relevante Aspekte bei der Kopplung nach Kopplungsstrategie und Simulationsmethode

Simulationsmethode / Kopplungsstrategie	Discrete Event	Continuous Time	Hybrid
Iterativ	(2), (4), (5b), (18)	(2), (7), (8), (10), (11)	(2), (4), (5b), (8), (10), (11), (13), (14), (15), (16), (17) (19), (20)
Synchron parallel	(2), (4), (5b), (18)	(2), (7), (8), (10), (11)	(2), (4), (5b), (8), (10), (11), (13), (14), (15), (16), (17) (19), (20)
Asynchron sequentiell	(1), (4), (5a), (18)	(1), (4), (7), (8), (11)	(1), (4), (5a), (8), (11), (13), (15), (16), (17), (19), (20)
Synchron sequentiell	(1), (4), (5a), (18)	(1), (4), (7), (8), (11)	(1), (4), (5a), (8), (11), (13), (15), (16), (17), (19), (20)

Synchron sequentiell überlappend	(1), (4), (5a), (18)	(1), (4), (7), (8), (11)	(1), (4), (5a), (8), (11), (13), (15), (16), (17), (19), (20)
Kopplungsstrategie-übergreifend	(3), (6), (9), (12)		

(1) **Pessimistische Kausalität**:
Durch die sequentielle Kopplung entsteht eine globale Ordnung, welche die Abhängigkeiten verschiedener Ereignisse bzw. Simulationseinheiten widerspiegelt. Somit erhält die globale Ordnung die Kausalität. Da Abhängigkeiten jedoch nicht immer erforderlich sind und zum Beispiel nur unter gewissen Voraussetzungen greifen sollen, ist die globale Ordnung in solchen Fällen sehr pessimistisch.

(2) **Optimistische Kausalität:**
Simulationseinheiten gehen optimistisch davon aus, dass das Verhalten anderer Simulationseinheiten sie nicht beeinflusst. Falls dies doch der Fall ist, müssen sie eine Rollback-Fähigkeit besitzen (also vergangene Zustände speichern und sich dorthin zurückversetzen). Dies benötigt Speicherkapazität, erhöht aber die Performanz.

(3) **Determinismus:**
Wenn eine Co-Simulation deterministisch ist, bedeutet dies, dass die Simulation immer exakt gleich abläuft.

(4) **Konfluenz**:
Bei Konfluenz gibt es nur einen möglichen Endzustand, der immer erreicht wird. Konfluenz ist im Allgemeinen schwer zu beweisen, es gibt aber Ansätze die dies sicherstellen können.

(5) **Dynamische Abhängigkeiten**:
Je nach Kopplungsstrategie der Simulationseinheiten, kann es sinnvoller sein keine festen Abhängigkeiten zu nutzen, um die Performanz zu steigern.
   a. Sequentielle Kopplung profitiert somit von dynamischen Abhängigkeiten, die nur wirken, wenn es erforderlich ist.
   b. Bei der parallelen Kopplung führen dynamische Abhängigkeiten eher dazu, dass weniger Speicherkapazität benötigt wird, als zur Verbesserung der Performanz.

(6) **Verteilungskosten**:
Simulationseinheiten einer Co-Simulation werden heutzutage häufig geografisch verteilt, was zu Verteilungskosten und zu Sicherheitsproblemen führen kann.

(7) **Algebraische Schleifen**:
Eine algebraische Schleife, welche zu starken Fehlern führen kann, entsteht, wenn eine Variable indirekt von sich selbst abhängt. Hier wird zwischen zwei Arten unterschieden: solche, die Eingabewerte umfassen, und solche, die zusätzlich Zustandsvariablen umfassen und durch implizite numerische Solver ausgelöst werden können. Im ersten Fall wird die Abhängigkeit unterbrochen, indem statt des Eingabewerts die Extrapolation des Eingabewerts in der Ausgabefunktion genutzt wird. Um den zweiten Fall zu vermeiden, müssen Extrapolationen statt Interpolationen für die Eingabefunktionen genutzt werden.

(8) **Inkonsistente Initialisierung**:
Initiale Zustände der Simulationseinheiten können durch algebraische Bedingungen über die Ausgabefunktion gekoppelt sein. Das bedeutet, dass die initialen Zustände nicht unabhängig voneinander festgelegt werden können. Ein einheitlicher Initialzustand aller Simulationseinheiten wird also bei einer Co-Simulation benötigt.

(9) **Error Control**:
Neben den „normalen“ numerischen Fehlern, die sich aus dem Modell, dem Solver,

den micro-step sizes und der Simulationszeit ergeben, kommen bei verteilten Simulationen noch der numerische Fehler der Extrapolationsfunktion hinzu, der sich durch das Feedback der Simulationseinheiten immer weiter vergrößert. Da es in der Regel nicht möglich ist, den genauen Fehler zu berechnen, versucht man stattdessen durch Error Control herauszufinden, wie der Fehler wächst.

(10) **Numerische Stabilität**:
Man spricht von numerischer Stabilität, wenn eine Makro-Schrittweite ungleich null gefunden werden kann, sodass der Fehler gegen null geht. Iterative Herangehensweisen führen zu einer verschlechterten Performanz, erhöhen jedoch die Stabilität. Numerische Stabilität der einzelnen Simulatoren garantiert nicht, dass auch das Gesamtsystem stabil ist.

(11) **Kompositionale Stetigkeit**: Bei Verwendung von Extrapolation können Unstetigkeiten auftreten. Wenn dies geschieht, wird die Unstetigkeit fälschlicherweise als starke Änderung des stetigen Verhaltens angesehen, was zu Problemen bei den Solvern der Simulationseinheiten führen kann. Dies kann so weit gehen, dass Unstetigkeiten zu anderen Simulationseinheiten propagieren.

(12) **Echtzeiteigenschaften**: Um eine Co-Simulation in Echtzeit zu ermöglichen, muss jede Simulationseinheit und das Netzwerk schnell genug seinsowie, falls ein reales System eingebunden werden soll, muss das System resilient gegen Rauschen von den gemessenen Daten sein. Außerdem muss die Co-Simulation mit verspäteten Dateneingaben umgehen können.

(13) **Semantische Adaption**: Während ein generischer Wrapper, der auf dem MoC der Simulationseinheit basiert, verwendet werden kann, basiert die Realisierung auf der konkreten Co-Simulation. Das heißt, es gibt nicht einen bestimmten, geeigneten Wrapper für alle Co-Simulationen. Um eine Simulationseinheit an einen Wrapper anpassen zu können, muss vorher bekannt sein, wo so eine Anpassung stattfinden kann.

(14) **Schrittweiten**: Sollen ein Discrete-Time (DT) und ein Continuous-Time (CT) Simulator gekoppelt werden, so kann der DT Simulator durch einen Wrapper als CT Simulator dargestellt werden. Der Wrapper muss dann eine Schrittweite festlegen, die auf Wissen über die Simulationseinheit beruht oder der vom Orchestrator festgelegten Kommunikationsschrittweite entspricht, wobei in letzterem Fall fehlende Ereignisse vorkommen können. Aus diesem Grund wird die Schrittweite oft adaptiv geschätzt.

(15) **Event Lokalisation**: Im Vergleich zu Punkt (14) ist es ebenfalls möglich, bei der Kopplung von DT und CT Simulatoren die CT Simulatoren durch einen Wrapper als DT Simulator darzustellen. Die Problematik besteht hierbei in der präzisen Erkennung der Überschreitung eines Schwellwertes für ein kontinuierliches Signal.

(16) **Unstetigkeitsidentifikation**: Zur Laufzeit wird ein Signal oft als eine Menge von zeitgestempelten Werten dargestellt. Die Beobachtung dieser Wertefolge allein macht es nicht möglich, eine steile Änderung in einem kontinuierlichen Signal von einer echten Unstetigkeit, die in einem Ereignissignal auftritt, zu unterscheiden.

(17) **Unstetigkeitsumgang**: Sobald eine Diskontinuität lokalisiert ist, hängt die Art und Weise wie sie behandelt wird, von der Art der Simulationseinheiten und ihren Fähigkeiten ab. Wenn die Simulationseinheit ein Mock-up eines kontinuierlichen Systems ist, dann sollten Diskontinuitäten in den Eingaben traditionell durch Reinitialisierung der Simulationseinheit behandelt werden. Dieser Schritt kann hohe Einbußen bei der Performanz verursachen.

(18) **Legitimität**: Wenn die Legitimität der Simulation verletzt ist, führt dies dazu, dass der Orchestrator eine unendliche Anzahl von Ereignissen mit dem gleichen Zeitstempel zwischen den Simulationseinheiten verschiebt, wobei die Zeit niemals voranschreitet.

(19) **Zeno Verhalten**: Man spricht vom Zeno Verhalten, wenn ein immer kleiner werdendes Zeitintervall zwischen zwei fortlaufenden Ereignissen besteht, sodass die Summe dieser Zeitintervalle endlich ist.

(20) **Zustandsapproximation**: In der Discrete-Event Simulation existieren keine Rundungsfehler und somit auch keine Methoden, mit solchen Fehlern umzugehen. In der Continuous-Time Simulation ist allerdings beides üblich. Dies kann zu Problemen führen, wenn bei einer hybriden Co-Simulation die Discrete-Event Simulation nicht erkennt, wann ein Fehler zu groß ist, um toleriert werden zu können.

2.1.3.4.5 Überblick über Kopplungsmechanismen für Simulationsmodelle und Simulatoren

Es existieren verschiedene Standards, die Teile von möglichen Kopplungsmechanismen adressieren.

Functional Mock-Up Interface (FMI)

In diesem Abschnitt wird das Functional Mock-Up Interface (FMI) in der Version 2.0.2 beschrieben, da sich FMI in der Version 3.0 zum Zeitpunkt der Erstellung dieses Dokuments noch in der Entwicklung befindet und erst eine Alpha-Version vorliegt. Nichtsdestotrotz sollte an dieser Stelle aufgrund der Relevanz in den *Simulation Use Cases* im Projekt SET Level genannt werden, dass die Version 3.0 im Vergleich zur Version 2.0.2 die Übertragung von Binärdaten unterstützt.

Verwendungszweck
In (Modelica Association, 2020) wird FMI 2.0.2 als ein offener, werkzeugunabhängiger Standard beschrieben, der den Modellaustausch und die Co-Simulation von dynamischen Modellen (continuous time und discrete time) unterstützt.

Funktionsweise
FMI definiert ein Interface welches durch eine Functional Mock-Up Unit (FMU) bereitgestellt wird. Durch den Aufruf einer fest definierten Methode kann eine FMU instanziiert und anschließend diese Instanziierung simuliert werden. Eine FMU besteht dabei aus

- einer *.xml Datei, die alle Variablen spezifiziert, auf die von außen zugegriffen werden kann, sowie Informationen über diverse optionale Parameter enthält (z. B. Verfügbarkeit von Richtungsableitungen, Möglichkeit der Benutzung einer variablen Kommunikationsschrittweite).
- Funktionen in C, die entweder als Quellcode, Binärdateien oder deren Kombination vorliegen. Dabei können insbesondere für unterschiedliche Plattformen auch unterschiedliche Binärdateien zur Verfügung gestellt werden. Hierzu gehören auch Funktionen zur Instanziierung der FMU sowie zur Steuerung und Zerstörung der Instanzen.
- zusätzlichen Informationen (z. B. ein Modellsymbol als Bitmap, Dokumentationsdateien).

Bei FMIs für Co-Simulationen wie in Abbildung 63 wird der Datenaustausch zu diskreten Kommunikationspunkten limitiert. In der Zeit zwischen diesen Kommunikationspunkten rechnen die Simulatoren unabhängig voneinander mit ihren eigenen Solvern. Sogenannte Master-Algorithmen kontrollieren den Datenaustausch und die Synchronisation von allen Solvern (Slaves).

Abbildung 63: FMI für Co-Simulation und FMI für Model Exchange nach (10 Years of FMI, 2018)

Der Master-Algorithmus ist nicht Teil des FMI-Standards, es können jedoch in der Anwendung Einschränkungen durch die *.xml Konfigurationsdatei vorliegen (z. B. durch die fehlende Unterstützung variabler Kommunikationsschrittweiten). Neben dem Master-Algorithmus benötigt der Master eine Lösung für die Kommunikation. Dies kann das FMI sein, mit dem ein Master mit den Slaves kommunizieren kann. Die Abfolge der Abrufe der genannten C-Funktionen unterscheidet sich dabei bei FMI for Model Exchange und FMI for Co-Simulation, siehe Abbildung 64, wobei wie dargestellt unter Nutzung von FMI for Co-Simulation kein Datenaustausch während der Simulationsschritte stattfinden kann, sondern ausschließlich zwischen den Simulationsschritten. Dargestellt wird dies durch die Verlagerung des „step In Progress“-Knotens nach außen und das Fehlen der fmi2GetX und fmi2SetIN an selbigem Knoten. Hingegen ist bei FMI for Model Exchange der Datenaustausch derart umgesetzt, dass ein Abruf der Daten durch eine andere FMU die Berechnung der Gleichungen auslöst und die Daten nach der Berechnung direkt übergeben werden.

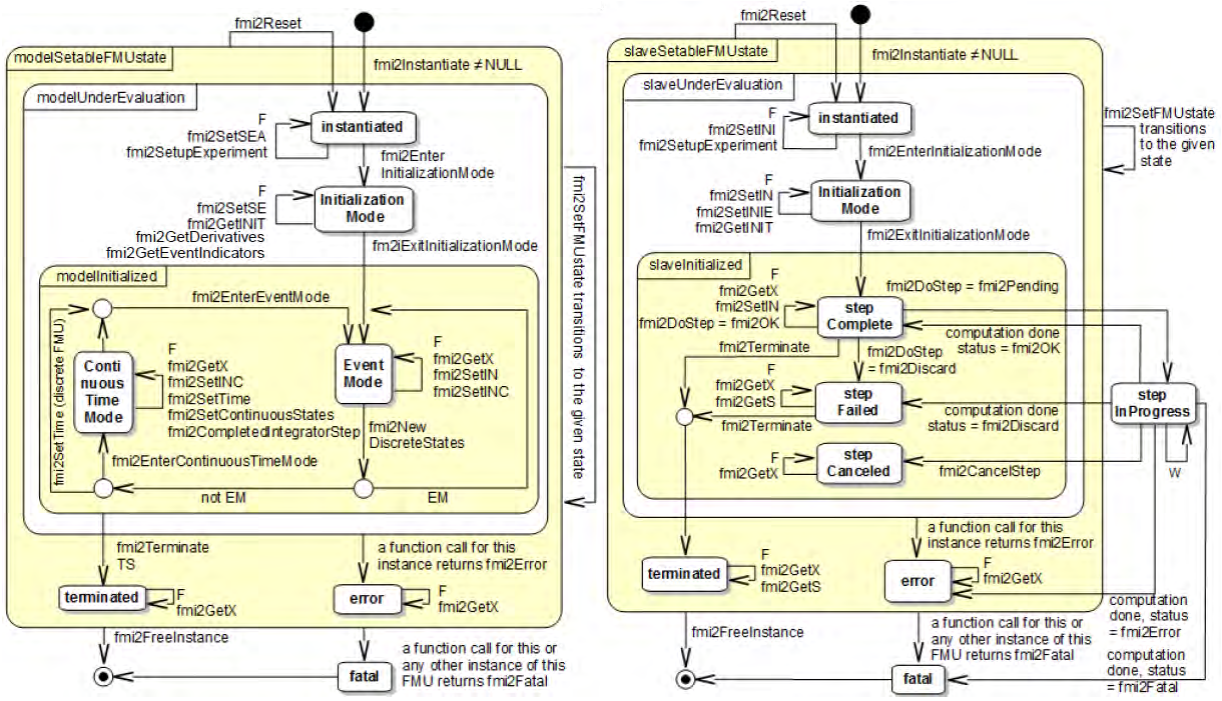


Abbildung 64: FMI State Machines, links FMI for Model Exchange, rechts FMI for Co-Simulation nach (Modelica Association, 2020)

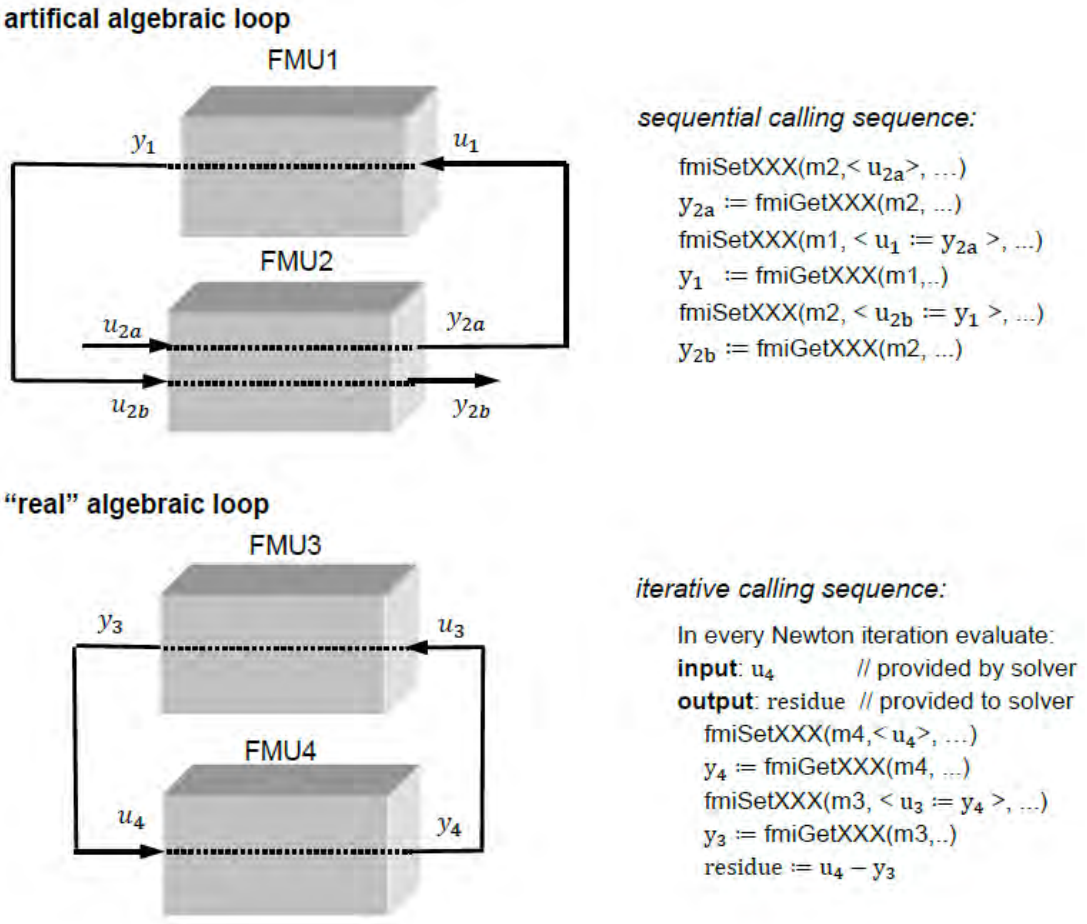


Abbildung 65: Calling sequences for FMUs that are connected in a loop

Weiteres
In (Arnold, et al., 2013) werden mathematische Analysen beschrieben (mit einem Master-Algorithmus aus (Master zur Simulatorkopplung via FMI, 2012)), die zeigen, dass der globale Fehler durch die lokalen Fehler begrenzt wird, falls keine algebraischen Schleifen im System vorhanden sind wie in Abbildung 65 dargestellt.

Distributed Co-Simulation Protocol (DCP)

Das Distributed Co-Simulation Protocol (DCP) (Modelica Association, 2019) ist ein plattform- und kommunikationsmittelunabhängiger Standard für die Integration von Echtzeitsystemen in Simulationsumgebungen.

Verwendungszweck
Die DCP-Entwicklung wurde, wie in Abbildung 66 zu sehen ist, von der Idee getragen, simulationsbasierte Arbeitsabläufe effizienter zu gestalten, den Integrationsaufwand zu reduzieren und damit verbundene Prozesse zu vereinfachen sowie die Integration von Echtzeitsystemen zu verbessern. DCP ermöglicht die Interaktion zwischen Echtzeit und Nicht-Echtzeitsystemen unterschiedlicher Anbieter.

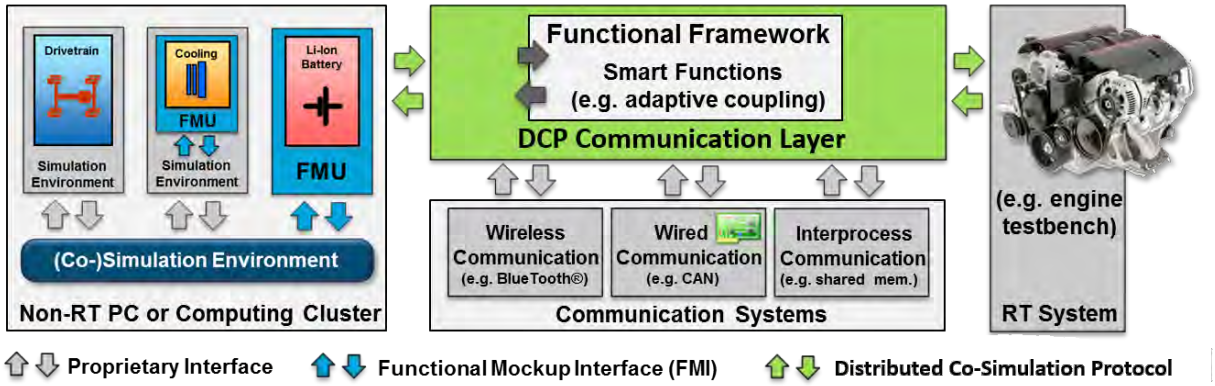


Abbildung 66: Übersicht Verwendungszweck DCP aus (ACOSAR, 2018)

Funktionsweise
DCP verfügt über das Master-Slave-Prinzip. Es spezifiziert ein Datenmodell, eine endliche Zustandsmaschine, einen Satz von Protokolldateneinheiten und ein Kommunikationsprotokoll.
Die diskrete Zustandmaschine sorgt für sichere und zuverlässige Operationen der Systeme. Dabei entspricht ein Simulationszyklus einem Durchgang der Zustandsmaschine. Diese Zustandsmaschine arbeitet nach dem identischen Ablauf wie FMI für Co-Simulation. Die Kopplung und damit die numerischen Eigenschaften der Simulation entsprechen denen von FMI CoSim.
Ein Durchgang ist unterteilt in mehrere Zustände. Zuerst weist sich der Master einem registrierten Slave zu, dann folgen Zustände, die die Initialisierung, einen Kommunikationsschritt und den Datenaustausch beschreiben. Zuletzt wird durch den Slave oder Master in den Zustand des Simulationsstopps gewechselt.
Die Kommunikation unter dem Master und den Slaves geschieht über sogenannte Protocol Data Units (PDU). Außerdem dienen sie der Fehlerbehandlung. PDUs können in verschiedene Kategorien unterteilt werden: Data PDUs übermitteln z. B. Eingangs- und Ausgangsdaten zwischen Master und Slave oder unter Slaves. Notification PDUs werden von Slaves gesendet, um den Master über die Änderung ihres Zustandes zu informieren oder um einen Protokolleintrag zu verschicken. Zu Control PDUs gehören die Request und Response PDUs. Requests PDUs sind Anfragen vom Master an den Slave. Darunter fallen Konfigurations-, Informations- und Zustandsänderungsanfragen. Bestätigt werden diese Anfragen durch Response PDUs vom Slave an den Master.
Darüber hinaus ist DCP unabhängig von dem zugrundeliegenden Transportprotokoll definiert und soll mehrere Transportprotokolle unterstützen können. Bei DCP wird zwischen nativen und nicht-nativen DCP-Spezifikationen unterschieden. Native Spezifikationen sind solche, bei denen das Mapping von PDUs zum Transportprotokoll die Bit-Sequenz bewahrt. Dazu gehören Bluetooth RFCOMM, USB und UDP/IPv4. Ist dies nicht der Fall, wird von Nicht-nativen Transportprotokollen (z. B. CAN) gesprochen. Mit diesem Ansatz kann in Zukunft die Unterstützung für zusätzliche Transportprotokolle hinzugefügt werden. Die DCP-Spezifikation beinhaltet auch eine Standardintegrationsmethode.
Unter anderem kann bei DCP zwischen drei Arbeitsweisen (sog. operating modes) gewählt wählen. Die erste Arbeitsweise ist die harte Echtzeit, was bedeutet, dass alle Fristen eingehalten werden müssen und Simulationszeit synchron zur absoluten Zeit ist. Bei Abweichungen tritt ein Fehler auf. Die zweite Arbeitsweise ist die weiche Echtzeit, was bedeutet, dass die zuvor genannten Abweichungen nicht zwingend zu Fehlern führen und der slave weiterarbeitet. Die dritte Arbeitsweise ist die Nicht-Echtzeit, was bedeutet, dass die Simulationszeit

unabhängig von der absoluten Zeit ist. Die Simulationszeit kann bei dieser Arbeitsweise also schneller oder langsamer als Echtzeit sein.

High Level Architecture (HLA)

Die High Level Architecture (HLA) (IEEE Standards Association, 2010) beschreibt eine generische Architektur für die integrierte und verteilte Simulation mit dem Ziel der Wiederverwendbarkeit bestehender Simulatoren sowie einer möglichst großen Flexibilität hinsichtlich nutzbarer Simulations- und Datenmodelle zu gewährleisten. Ursprünglich wurde die HLA von dem „Defense Modeling and Simulation Office“ (DMSO) des amerikanischen Verteidigungsministeriums entwickelt. Im Jahr 2000 wurde die HLA dann zum internationalen Standard IEEE 1516. Verwaltet werden die einzelnen Simulationen dabei von einer zentralen Komponente, der sogenannten "Run-Time-Infrastructure" (RTI). Diese überwacht den kompletten Simulationsablauf und verwaltet die Verteilung der Daten zwischen den Einzelsimulationen. Die HLA eignet sich sowohl für die Co-Simulation als auch für die Simulatorkopplung. Sie unterscheidet sich von FMI und a. dadurch, dass sie spezifische Mechanismen für den Datenaustausch und das Zeitmanagement bereitstellt. Dadurch wird die verteilte Simulation ermöglicht.

Funktionsweise
Die HLA definiert ein generelles Framework und Regeln, an die sich die an der Simulation beteiligten Simulatoren halten müssen, um die Kompatibilität mit den anderen Komponenten zu gewährleisten. Es wird allerdings keine Implementierung der notwendigen zentralen Steuerungskomponente, der sogenannten Run-Time Infrastuktur (RTI) geliefert. Lediglich notwendige Schnittstellen die eine RTI zur Verfügung stellen muss werden spezifiziert. Alle Informationen werden dezentral gespeichert und nur auf Anforderung geteilt.
Bei HLA spricht man von federation execution und federate. Im Folgenden werden diese als Föderation und Föderat bezeichnet. Eine ausführbare Föderation besteht aus einem Zusammenschluss von Föderaten und einem gemeinsamen Föderationsobjektmodell (FOM). Eine Föderat ist vergleichbar mit einer Simulation und kann einer Föderation beitreten. Die beigetretenen Föderaten sind durch die RTI verbunden.
Es gibt drei verschiedene Föderationsobjektmodelle. Zum ersten gibt es ein FOM, welches die Informationen definiert, die während der Laufzeit innerhalb der Föderation ausgetauscht werden. Das zweite FOM, das Simulationsobjektmodell (SOM) gibt an, welche Art von Informationen jeder einzelne Föderat anbietet oder welche Art von Information ein Föderat von anderen erhalten kann. Das dritte FOM, das Managementobjektmodell ist eine Gruppe von vordefinierten Konstrukten zur Unterstützung bei der Kontrolle und Bewachung der Föderation.
Neben dem Objektmodell Template (OMT) gibt es das Föderat Interface, welches das funktionale Interface zwischen den Föderaten und der RTI beschreibt und deutlich macht, welche Services das RTI bietet. Zudem gibt es je fünf Regeln für Föderaten und Föderation, die die Grundlagen für eine standardmäßige Kommunikation absichern.
Jeder Föderat ist zuständig für sein eigenes Zeitmanagement. HLA unterstützt drei Mechanismen, in der Zeit voranzuschreiten: koordinierte Zeit, wobei das Zeitmanagement jedes Föderaten miteinander koordiniert ist, über Nachrichten, die das Voranschreiten eines Föderaten in der Zeit auslöst, und optimistisch. Bei dem optimistischen Mechanismus mit Rollbacks kann jeder Föderat unabhängig voneinander in der Zeit voranschreiten. Dies wird oft bei Echtzeit-Föderaten genutzt.

Herausforderungen
Die Run-Time Infrastructure (RTI) verbindet die Föderaten miteinander zu einer Föderation. Wenn zwei Föderationen oder zwei Föderaten miteinander verbunden werden sollen, ist dies

bei unterschiedlichen RTI Implementierungen nicht direkt möglich. Unterschiedliche RTI Implementierungen sind in der Funktion zwar sehr ähnlich, können jedoch nicht einfach miteinander zusammenarbeiten, da sie verschiedene, geschützte Protokolle zum Datentransport nutzen (wire protocol). Dieses Protokoll beschreibt, wie Informationen zwischen Föderaten ausgetauscht werden.
Eine Möglichkeit zwei Föderationen oder Föderaten zu verbinden ist, eine Föderation oder einen Föderaten zu der RTI Implementierung der oder des anderen wechseln zu lassen. Diese Umstellung ist jedoch nicht ohne Verluste möglich.
Durch einen Standard für wire protocols wäre die Verbindung verschiedener RTI Implementierungen möglich. An einem entsprechenden Standard wurde in der Vergangenheit gearbeitet, dieser existiert bisher aber nicht. Zudem würde so ein Standard den Nutzungsumfang von HLA einschränken.
Wenn man Föderaten unterschiedlicher RTI Implementierungen verbinden will, gibt es die Möglichkeit der RTI-zu-RTI Brücken. Diese soll Daten von einem Föderaten erhalten und an einen Föderaten mit anderer RTI Implementierung schicken. Dafür muss die Brücke jedoch der ganzen Föderation beitreten, die sie überbrücken will, da sie die Daten ansonsten weder empfangen noch senden kann. Die RTI-zu-RTI Brücke fungiert also als Föderat in beiden Föderationen und verbindet sie so miteinander. Somit hat diese Methode negativen Einfluss auf die Leistung. Außerdem leidet die Funktionalität unter den Brücken und für manche RTI Services ist solch eine Überbrückung nicht möglich. Wenn zum Beispiel ein Service nur den internen Zustand des RTI ändert, wird es anderen Föderaten, und a. der Brücke, nicht mitgeteilt. Somit kann die Brücke nicht dieselbe Änderung in der anderen Föderation auslösen.

Kopplung mit FMI
FMI für Co-Simulation bietet keine direkte Möglichkeit FMUs und HLA Föderationen zu koppeln. Außerdem kann eine solche Simulationsumgebung nur primitive Datentypen statt komplexe unterstützen. FMI schränkt in dem Fall HLA ein.
In (Youssef Bouanan, 2018) werden zwei Möglichkeiten vorgestellt FMI und HLA mit Hilfe von hybriden Föderaten zu koppeln. Diese Herangehensweisen sind jedoch noch nicht vollständig ausgearbeitet.
Bei der Adapter-based Methode besteht der hybride Föderat aus einer FMU und einem Adapter, der die Interaktionen zwischen der RTI und der FMU sowie den Lebenszyklus der FMU verwaltet.
Bei der Mediator-based Methode besteht der hybride Föderat aus drei Elementen: der FMU, dem HLA Föderat und einem Mediator. Letzteres ist dafür zuständig, zwischen der FMU und dem HLA Föderat zu kommunizieren und den gesamten hybriden Föderat zu koordinieren.

Robot Operating System (ROS)

Das Roboter Operation System (ROS) (Stanford Artificial Intelligence Laboratory et al., 2018) ist ein Open-Source-Software-Framework für Anwendungen in der Robotik.

Verwendungszweck
ROS stellt Bibliotheken und Tools zur Verfügung. So auch ein modulares Navigationssystem, welches auf Landzeichen basiert. Es kann für beliebige Robotik Systeme verwendet werden und findet somit auch in der Automobilbranche Beachtung

Funktionsweise
Ein Robotersteuerungssystem besteht aus mehreren Knoten (Nodes), die für unterschiedliche Berechnungen zuständig sind. Die Kommunikation zwischen diesen Knoten findet über

Nachrichten (Messages) statt und kann in synchrone und asynchrone Kommunikation unterteilt werden. Unter anderem für diesen Nachrichtenaustausch ist der Master zuständig.
Bei der anonymen asynchronen Kommunikation (publish-subscribe pattern) sendet ein Knoten eine Nachricht, indem er sie einem bestimmten Thema (Topic) übergibt. Ein Knoten, der dieses Thema abonniert hat, erhält daraufhin diese Nachricht.
Bei der synchronen Kommunikation (request/reply) bietet ein Knoten einen Service an. Der Bezieher (Client) fragt mit einer Nachricht den Service an und wartet auf die Antwort, die ebenfalls in Form einer Nachricht zurückgegeben wird. Im Gegensatz zur synchronen Kommunikation findet der Austausch hier nur einmalig statt.

Herausforderungen
Determinismus ist nicht automatisch in jedem ROS gegeben. Da nicht-deterministische Systeme häufig schlechtere Ergebnisse liefern, ist es wichtig, die Reproduzierbarkeit des Systems sicherzustellen. Dabei ist auf folgende Dinge zu achten.
Beispielsweise kann die falsche Zeiterfassung zu einem Problem werden. Eine Eingabe hat einen Zeitstempel, der angeben soll, wann dieser Wert erfasst wurde. Dieser Zeitpunkt kann mit Hilfe einer Uhr, die in dem Sensor enthalten ist, zusammen mit den Eingabedaten gesendet werden. Bei diesem Vorgehen ist zu beachten, dass die Synchronisierung der Sensor- und PC-Uhr nie exakt ist und zu einem Offset (clock misalignment/ skew) führt, welcher sich über die Zeit vergrößert (clock drift). Eine Möglichkeit, dies zu umgehen, ist es, auf Zeitangaben zu verzichten und den Host ein Signal schicken zu lassen, sobald Daten erfasst werden sollen. Hierbei muss unter anderem auf die Verzögerung in der Signalübertragung geachtet werden.
Asynchrone Kommunikation kann außerdem zum „sampling effect“ führen. Beim „sampling“ werden kontinuierliche Signale in regelmäßigen Intervallen zu diskreten Signalen überführt. Bei entkoppelten Systemen, in denen die Werte nicht direkt übergeben werden, entsteht der „sampling effect“, wenn die Abstände, in denen Daten empfangen werden, sich ändern. Dies wird durch „scheduling jitter“ beeinträchtigt. Durch die sich ändernden Abstände wird manchmal vor Empfang der Daten, manchmal danach gearbeitet, was zu unterschiedlicher Reaktionszeit und unterschiedlichen Ergebnissen führen kann.

System-Funktionen in MATLAB/Simulink (engl. “s-functions”)

Systemfunktionen (kurz „S-Funktionen“, engl. „s-functions“) sind ein leistungsfähiger Mechanismus zur Erweiterung der Fähigkeiten der Simulinkumgebung von MATLAB.
Mit einer S-Funktion kann ganz allgemein ein in Textform vorliegender Algorithmus in ein Simulinkmodell eingebunden werden.

Verwendungszweck
Eine S-Funktion ist, einfach ausgedrückt, eine Codeschnittstelle zur Einbindung von Code in Simulink.
Mit Hilfe der S-Funktionen können so auch in Textform formulierte Simulationsmodelle in eine MATLAB-/Simulink-Umgebung eingebunden werden.
Einige Modellierungstools wie z. B. ADAMS, Simpack oder Dymola nutzen S-Funktions daher auch als Format für einen Simulationsmodellexport für Matlab/Simulink-Umgebungen.

Funktionsweise
Eine S-Funktion kann in MATLAB® m-Code, C, C++ oder Fortran implementiert werden und kann unabhängig davon jeweils als kontinuierliche, diskrete oder hybride S-Funktion definiert sein.

C, C++- und Fortran-S-Funktionen werden zu ausführbaren Dateien kompiliert, die von der MATLAB-Ausführungsengine automatisch geladen und ausgeführt werden können.
S-Funktionen verwenden dabei eine spezielle Aufrufsyntax, die so genannte „S-Funktions-API“, mit der Sie mit der Simulink-Engine interagieren können. Diese Interaktion ist der Interaktion zwischen der Engine und den integrierten Simulink-Blöcken sehr ähnlich.

Herausforderungen
Das S-Funktionen-Format gibt es schon sehr lange in MATLAB/Simulink bzw. in anderen Tools, wenn auch immer wieder mit geänderter Struktur und Syntax.
Der Simulationsmodell-Export in Form einer S-Funktion und deren Einbindung in eine MATLAB/Simulink-Umgebung funktioniert sehr zuverlässig, einfach und schnell und kann auch sehr einfach automatisiert werden.
Dies zum einen auf Grund der umfangreichen Tool-Unterstützung hinsichtlich Generierung, Kompilierung und Einbindung der S-Funktions in MATLAB/Simulink sowie auch zusammen mit Produkten der Firma dSPACE, zum anderen aber vermutlich auch auf Grund der langen Erfahrungen mit diesem Format beim Hersteller The MathWorks und auch anderen Toolherstellern.
Diese Vorteile können daher auch für den Simulationsmodellexport und -import in Form einer S-Funktion direkt genutzt werden.
Ein Nachteil des S-Funktionen-Formats ist jedoch, dass es sich bei diesem Format um ein proprietäres Format der Firma The MathWorks handelt, wenngleich es auch von zahlreichen anderen Toolherstellern unterstützt wird.
Das heißt, dass das S-Funktionen-Format nicht toolunabhängig einsetzbar und streng genommen sogar jeweils abhängig von der jeweils konkret verwendeten MATLAB/Simulink-Version zu deren Erzeugung ist.
Das S-Funktionen-Format ist daher für die Kopplung von Simulationsmodellen nur eingeschränkt nutzbar bzw. auf MATLAB/Simulink-Umgebungen und damit verbundenen Toolketten (wie z. B. die von der Firma dSPACE) beschränkt).

Code und Code Libraries (z. B. in C, C++ oder Fortran)

Code und Code Libraries haben den großen Vorteil, dass sie bis auf den notwendigen Kompiler sowohl toolunabhängig als auch hardwareunabhängig sein können.
Derartige Simulationsmodelle können z. B. aus der Historie vorliegen und z. B. in C oder Fortran programmiert sein.
Oder derartige Simulationsmodelle werden aktuell ganz bewusst so erzeugt und programmiert z. B. in objektorientierten Programmiersprachen wie etwa C++, um die Vorteile der Toolunabhängig als auch Hardwareunabhängig zu nutzen.

Funktionsweise
Simulationsmodelle bestehend aus Code und Code Libraries können z. B. direkt als Code in Simulationsumgebungen eingebunden bzw. gekoppelt und so kompiliert werden wie z. B. in der Simulationsumgebung „CarMaker“ der Firma IPG.
Simulationsmodelle bestehend aus Code bzw. Code Libraries können zur Simulationsmodellkopplung aber auch in ein S-Funktionen-Format gebracht bzw. eingebettet werden oder entsprechend des FMI-Standards in eine FMU verpackt werden.

Herausforderungen
Simulationsmodelle bestehend aus Code und Code Libraries und deren Kopplungen sind sehr zeitlos, sodass sie sehr universell einsetzbar und daher auch sehr einfach in bestehende oder neue Entwicklungsprozesse integrierbar sind.

Von Nachteil bei derartigen Simulationsmodellen und deren Kopplung ist und a. der erhebliche Implementierungsaufwand der nötig sein kann, um derartige Simulationsmodellen mit anderen Simulationsmodellen zu koppeln.
Auch die Notwendigkeit von zum Teil erheblichem Implementierungs-Know-How zur Kopplung derartiger Simulationsmodellen mit anderen Simulationsmodellen und der dadurch einhergehenden, geringen Automatisierbarkeit sind weiterer Nachteile bei der Verwendung derartiger Simulationsmodelle.

2.1.3.4.6 Überblick über Gütekriterien für Simulationsmodelle

Um Gütekriterien für die Kopplung von Simulationen zu beschreiben, müssen auch Gütekriterien der verwendeten Simulationsmodelle berücksichtigt werden. Nur dann kann sichergestellt werden, dass durch die Kopplung der verwendeten Simulationsmodelle und deren Ausführung in einer Simulationsumgebung nicht die Gütekriterien der Gesamtsimulation verletzt werden.
Gütekriterien für Simulationsmodelle können angeben, wie gut die Messungen aus dem Simulationsmodell, mit den Messungen korrespondieren, die aus dem realen repräsentierten Teil des Systems stammen. Dies ist in der Praxis jedoch schwierig, da u. a. die Frage welche Messungen die wichtigen sind, schon ein Problem darstellen kann. Weiterhin müssen geeignete Abstandsmaße für die entsprechenden Größen definiert werden, was in vielen Fällen ebenfalls keine triviale Aufgabenstellung ist.
Natürlich sind Modelle immer Abstraktionen der realen Welt. Ziel bei diesen Abstraktionen ist es, die Teile wegzulassen, die für den gerade zu untersuchenden/betrachtenden Anwendungsfall nicht relevant sind, um damit einen Kompromiss aus dem benötigten Rechenaufwand und der nötigen realitätsnahen Repräsentation des Systems zu finden.
Gütekriterien zur Bewertung von Simulationsmodellen können hierbei unterschiedliche Qualitätsaspekte betrachten. Zunächst muss eine Validierung des Modells die Eignung des Modells für den betrachteten Anwendungsfall sicherstellen und bewerten. Zusätzlich sind auch weitere Attribute (u. a. Datenglaubwürdigkeit, Unsicherheitscharakterisierung, Ergebnisrobustheit) mögliche Gütekriterien.
Eine Möglichkeit, um Simulationsmodelle zu validieren, ist ein Vergleich von Daten, die aus dem Simulationsmodell stammen mit Daten aus der Realität, die sie repräsentieren. Dies geschieht unter Verwendung von Metriken (siehe beispielsweise (Viehof, 2018)).
In (Kuefler, et al., 2017) werden zur Validierung des entwickelten Fahrermodells verschiedene Metriken verwendet. Einerseits wird die gewichtete quadratische Abweichung zwischen der Abweichung der Wahrscheinlichkeitsmasse von Trajektorien, andererseits und a. die Kullback-Leibler-Divergenz (Kullback, et al., 1951) zwischen den empirischen Verteilungen des Modells und Realdaten bezüglich gewählter Eigenschaften (Geschwindigkeit, Beschleunigung ...) verwendet.

2.1.3.4.7 Bewertungskriterien für Co-Simulation

Generelle Problematik

Gütekriterien zur Bewertung der Kopplungsmechanismen haben das Ziel den reinen Einfluss der Kopplung zu identifizieren, zu quantifizieren und zu bewerten. Daher müsste man das gekoppelte System eigentlich mit einem monolithischen System vergleichen, um Aussagen über die Güte der Kopplung zu bekommen. Diese steht jedoch im Allgemeinen nicht zur Verfügung. Ein reiner Vergleich mit der Spezifikation reicht nicht aus, um den Einfluss der Kopplungsmechanismen deutlich zu machen. Hier stellen sich beispielsweise folgende Fragen: Was sind Kopplungsfehler? Was sind Modellierungsfehler? In welchem Umfang wirken sich

die Fehler, die bei der Kopplung entstehen, überhaupt auf das Systemverhalten aus. Daher können tolerierbare Abweichungen nur mit detaillierter Kenntnis über die beteiligten Modelle und deren interne Funktionsweise erfolgen.
Einerseits kann man nun versuchen den Einfluss der Kopplung unabhängig von der implementierten Funktionalität zu bewerten (z. B. an Hand der den Modellen zugrundeliegenden Models of Computation). Möglich wäre aber auch die Berücksichtigung der Spezifikation der zu koppelnden Komponenten (z. B. Berücksichtigung der Fehlertoleranz, benötigten Genauigkeiten der Inputs, Jitter).

Stand der Technik

Nach (Günther, 2017) existieren verschiedene mögliche Bewertungskriterien für Co-Simulationen:

- Genauigkeit bezogen auf das Simulationsergebnis. Um zu bewerten wie sich lokale Fehler auf den globalen Fehler auswirken, müsste man eine monolithische Simulation mit einer gekoppelten Simulation vergleichen (auch genannt Referenzlösung).
- Aufwand. Kopplungsmechanismen erzeugen einen Overhead gegenüber einer monolithischen Simulation. Hierbei kann man
    - Rechenzeit (CPU Zeit, die aktiv gerechnet wird) und
    - Dauer (Wall clock time; Zeit von Start bis Stopp der Simulation)

  mit einer Referenzlösung vergleichen.
- Stabilität / Robustheit gegen numerische Fehler. Trotz numerischer Stabilität einzelner Simulatoren kann eine gekoppelte Simulation instabil sein.

Zusätzlich könnte man Kopplungsmechanismen für Co-Simulationen nach ihrem Aufwand bewerten, um Simulatoren und Simulationsmodelle zu verbinden. Hier stellt sich die Frage welche Informationen über Simulationsmodelle und Simulatoren benötigt werden und wie diese bereitgestellt werden. Diese Art der Aufwandsschätzung ist zumindest teilweise subjektiv. Weiterhin sparen getrennt modellierte Systeme Entwicklungszeit, die jedoch ebenfalls schwer mit einem monolithischen System zu vergleichen sind.
Zusätzlich könnte man Kopplungsmechanismen für Co-Simulationen nach ihrem Aufwand bewerten um Simulatoren und Simulationsmodelle zu verbinden. Hier stellt sich die Frage welche Informationen über Simulationsmodelle und Simulatoren benötigt werden und wie diese bereitgestellt werden. Diese Art der Aufwandsschätzung ist zumindest teilweise subjektiv. Weiterhin sparen getrennt modellierte Systeme Entwicklungszeit, die jedoch ebenfalls schwer mit einem monolithischen System zu vergleichen ist.

Identifikation von weiteren Kriterien zur Bewertung von Kopplungsmechanismen anhand der Simulationsziele

Weitere Kriterien zur Bewertung von Kopplungsmechanismen sollen anhand der Simulationsziele identifiziert werden.
Dazu muss zunächst geklärt werden, was unter einem Simulationsziel verstanden wird. Ein Simulationsziel ist der Grund oder Zweck der Erstellung und Durchführung von Simulationsläufen. Simulationsziele rechtfertigen den gewählten Simulationsumfang. Sie können hierarchisch sein und in weitere Untersimulationsziele unterteilt werden. Sie können in die verschiedenen (Prüf-) Ebenen aufgeteilt und in verschiedenen Detaillierungsgraden definiert werden. Beispiele für Simulationsziele sind in der Kritikalitätsanalyse (SUC 1) kritische Szenarien zu identifizieren oder neue Kritikalitätsmetriken zu erproben.
Diese Simulationsziele können nun herangenommen werden und mit Hilfe der in TP 1 erstellten Anforderungen können daraus die Implikationen für die zu instanziierende

Simulationsplattform erstellt werden und anschließend Eigenschaften abgeleitet werden, die die Kopplungsmechanismen erfüllen müssen oder können.
Beispielsweise existiert für das Simulationsziel „Verifikation einer Fahrfunktion“ die Anforderung, dass es die Möglichkeit geben muss reproduzierbare Simulationsläufe zu erzeugen. Für den Kopplungsmechanismus bedeutet dies also, dass er reproduzierbar ausführbar sein muss.
Eine weitere Anforderung zum gleichen Simulationsziel ist, dass die Möglichkeit gegeben sein muss die Simulationsmodelle auszutauschen. Dies bedeutet für den Kopplungsmechanismus, dass er standardisierte Schnittstellen bieten muss und mit unterschiedlichem Zeitverhalten der Simulationsmodelle umgehen können muss.
Weiterhin soll die Möglichkeit gegeben sein, dass Modellhersteller ihre IP wahren können. Für Kopplungsmechanismen bedeutet dies, dass sie nach ihrer Abhängigkeit von Simulationsmodellinformationen bewerten werden können (bspw. Benötigt die Eigenschaft Rollbackfähig zu sein). Auch hier sind die standardisierten Schnittstellen wieder eine wichtige Eigenschaft.

2.2 Teilprojekt 3: Modellspezifikation, -Entwicklung und -Validierung

2.2.1 Übersicht über Inhalte und Ziele von TP 3

TP 3 liefert einen vollständig definierten Prozess zum Entwickeln von Simulationsmodellen im Kontext der projektweit genutzten Simulationssysteme. Der Entwicklungsprozess, definiert durch die drei Elemente Spezifikation, Implementierung und Absicherung, beinhaltet Anleitungen, Templates und die Vorgabe von etablierten Standards. Zusätzlich wird der Prozess genutzt, um prototypisch im Projekt benötigte Modelle zu entwickeln. Diese Entwicklung der Modelle ist somit gleichzeitig die Überprüfung des Entwicklungsprozesses und somit ein Ergebnis des Arbeitspakets im Sinne eines Proof-of-Concept. Zusätzlich werden die entwickelten Modelle im Gesamtprojekt als Module in der Simulation eingesetzt.

Die im Projekt SET Level entstandenen Prozessbeschreibungen wurden als Ideen vom Projekt SmartSE des prostep ivip Vereins (siehe https://prostep.org) übernommen und ausgearbeitet. Für dieses Dokument wurde eine gesamthafte Darstellung der prozessbezogenen Inhalte gewählt, um ein vollständiges Bild für den Leser zu liefern. Für die Wartung und Weiterentwicklung werden sie zurückübertragen an das SmartSE Projekt.

Es ergaben sich in diesem TP drei Arbeitspakete (AP):

AP 3.1 Modellspezifikation und -aufbau
Methoden und Templates zur Spezifikation und Aufbau der zu entwickelnden Modelle, Erstellung der Modelle, sowie auf den Methoden zur Absicherung der Modelle im Sinne dieser Spezifikation. Es zeigte sich im Projektverlauf, dass neben dem Modellierungsprozess auch ein entsprechender Simulationsprozess zur Verfügung stehen muss. Es wurde deshalb das Credible Simulation Process Framework mit dem Simulation-based Decision Process, Credible Simulation Process, Credible Modeling Process sowie der Detaillierung eines Kooperationsprozesses zwischen zwei Partnern entwickelt.

AP 3.2 Modellauswahl und Integration.
Entwicklung und Erprobung einer Methodik und Vorgehensweise zur Modellauswahl. Umsetzung und Erprobung der in AP 3.1 entwickelten Methodik zur Testautomatisierung der Absicherung an den zu entwickelten Modellen.

AP 3.3 Datenmanagement
Das AP 3.3 „Datenmanagement“ hat die Aufgabe, sicherzustellen, dass alle benötigten Informationen für die Modellauswahl und -integration im Datenmanagementsystem (TP 4) zur Verfügung stehen. Dies beinhaltet insbesondere die Metainformationen der Modelle (z. B. Kurzbeschreibung, Einsatzbereiche, Eigenschaften, Nutzung, PDM, PLM Ansätze), welche Ablage, Versionierung und Auswahl bzw. Suche und Wiederauffindbarkeit ermöglichen.

Aufbau Ergebnisse von TP 3 zum finalen Meilenstein
Es sollten für TP 3 bis zum Projektabschluss iterativ über diverse Meilensteine hinweg die folgenden Ziele durch schrittweise Komplexitätssteigerung erreicht werden:

- Erstellung eines vollständig definierten Prozesses zum Entwickeln von Simulationsmodellen im Kontext des projektweiten Simulationssystems
- Entwicklung eines Modellaustauschprozesses für numerische Simulationsmodelle.
- Entwicklung und Erprobung einer Methodik und Vorgehensweise zur Modellauswahl

- Definition der erforderlichen Metadaten, die durch die Modelle bereitgestellt werden müssen, um den Prozess der Modellauswahl und Integration sicherzustellen
- Entsprechende Definition für Metadaten für Tools, Karten & Szenarien, Parameter
- Methoden zur automatisierten Absicherung von Modellen sind implementiert und an den entwickelten Modellen erprobt worden
- Erstellung der Modelle
    - Umweltmodellierung mit Verkehrsteilnehmern
    - Sensormodelle
    - Fahrfunktion (Umwelterfassung, Entscheidung, Bewegungsplanung)
    - Fahrzeugaktorik
    - Fahrdynamikmodell (mit Einbaulagen der Sensoren)
    - Einfaches Modell des Fahrers, Fokus Level 4 und 5

2.2.1.1 Credible Simulation Process Framework

Die Integration der Simulation in die Entwicklung und Validierung von automatisierten Fahrfunktionen gewinnt zunehmend an Bedeutung. Im Rahmen des Projekts SET Level wurden dazu eine Reihe von abgestimmten Prozessen und Vorgehensweisen entwickelt (Credible Simulation Process Framework). Diese basieren auf den Vorarbeiten des prostep ivip SmartSE Projektes. Besonderes Augenmerk wurde auf folgende Punkte gelegt:

- Nachverfolgbarkeit und Nachvollziehbarkeit
- Anpassungsfähigkeit an unternehmens- und domänenspezifische Anforderungen und Vorgehensweisen
- Kollaboration mit Partnern
- Modularität und Wiederverwendung

Die Beschreibung der Prozesse ist bewusst auf einer generischen Ebene gehalten und soll als Rahmen für eine domänen- oder unternehmensspezifische Spezifikation dienen. Die einheitlichen Grundstrukturen bzw. Frameworks sollen insbesondere die Zusammenarbeit innerhalb des Unternehmens und mit Partnern im Bereich des simulationsbasierten Engineerings unterstützen.
Die Grundstruktur des Credible Simulation Process Frameworks ist in der folgenden Abbildung dargestellt:

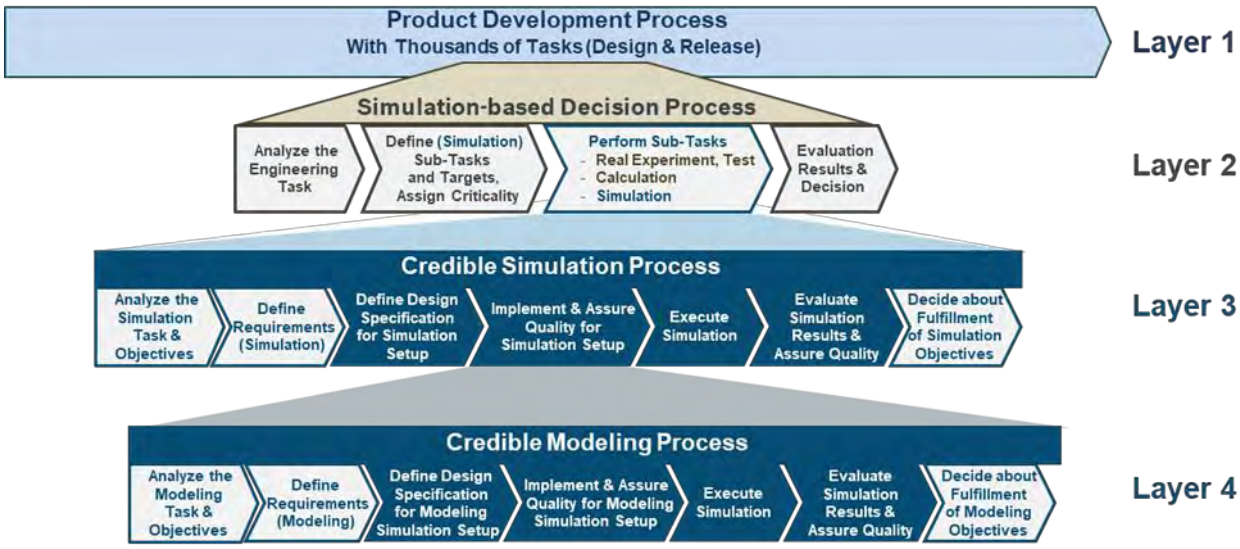


Abbildung 67: Grundstruktur des Credible Simulation Process Frameworks

In einer Produkt-, Plattform- oder Basisentwicklung gibt es viele Entscheidungen für das Design und die Freigabe (Layer 1).
Einer der Entscheidungsprozesse ist der simulationsbasierte Entscheidungsprozess (Layer 2): Das bedeutet, dass die Simulation ein wesentlicher Bestandteil für die Ausführung und Vertrauenswürdigkeit der Teilaufgaben ist.
Im simulationsbasierten Entscheidungsprozess (Simulation-based Decision Process) wird festgelegt, mit welchen Lösungsansätzen die Aufgabe bearbeitet werden soll und was die Ziele und Anforderungen an die Lösungsansätze sind. Diese Informationen und die Simulationsaufgabe selbst sind Input für den Credible Simulation Process (CSP). Der simulationsbasierte Entscheidungsprozess kann auch reale Tests mit einbeziehen.
Die ersten beiden Phasen im Simulationsbasierten Entscheidungsprozess (Layer 2) werden durchlaufen, dann werden die relevanten Informationen an den Credible Simulation Process (Layer 3) weitergegeben und dort verarbeitet. Die Spezifikationen in der zweiten Phase "Define Requirements" des CSP (Layer 3) und der CMP (Layer 4) sollten in Abstimmung mit den Beteiligten des Simulationsbasierten Entscheidungsprozesses definiert werden. Zur Hervorhebung sind die Phasen mit Abstimmung zwischen den Prozessen bzw. Layern weiß dargestellt.
Im Credible Simulation Process (CSP) können Teilaufgaben wie die Modellentwicklung definiert und initiiert werden (Layer 3). Die Informationen werden dann an den Credible Modeling Process (CMP) auf Layer 4 weitergegeben.
In der Anwendung des Credible Simulation Process Frameworks im Kontext des VVMethoden Projektes bedeutet dies: Im VVMethoden Projekt werden die Argumentationsketten und Methoden für Entscheidungen im Bereich Entwicklung und Freigabe von automatisierten Fahrfunktionen entwickelt. Es werden dort die Anforderungen an die spezifischen Simulationsaufgaben spezifiziert und als Simulation Request an den Simulationsprozess übergeben (siehe Abbildung 68).

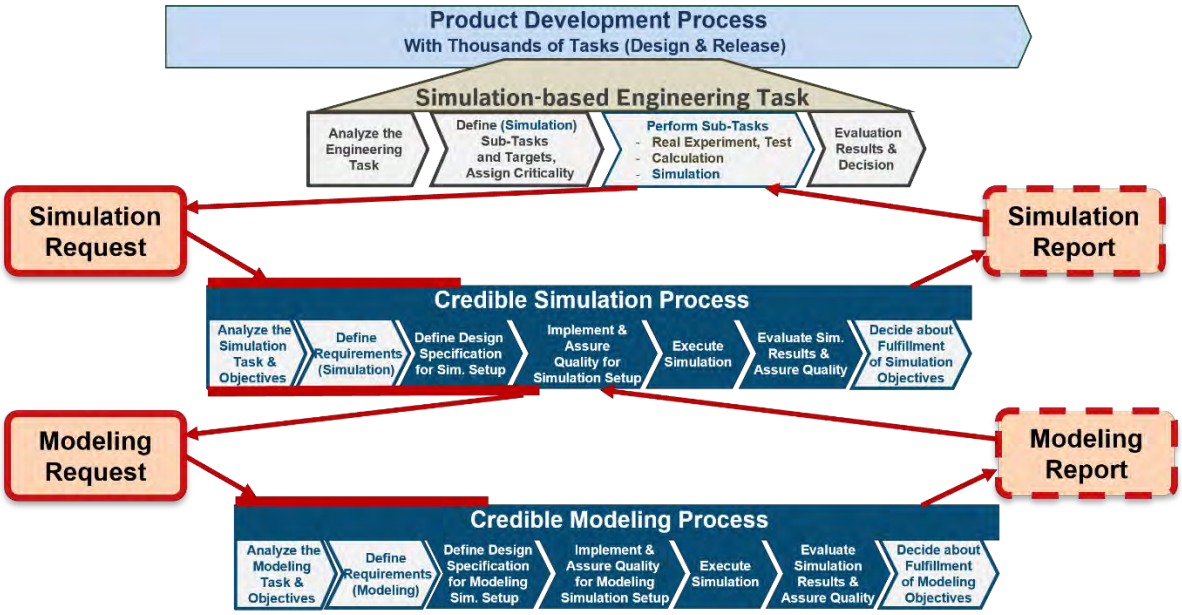


Abbildung 68: Verbindung der Prozesse über definierte Anforderungs-, Ergebnispakete

2.2.1.2 Grundstruktur eines Engineering Prozesses (Phasen)

Im Folgenden wird die allen hier aufgeführten Prozessen zu Grunde liegende Struktur beschrieben (siehe Abbildung 69).

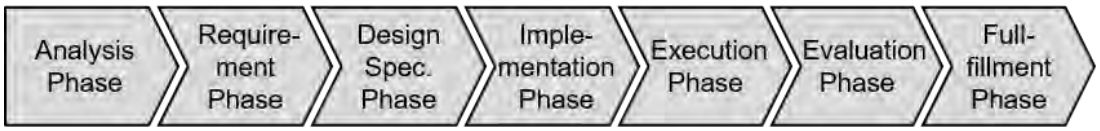


Abbildung 69: Grundstruktur eines Engineering-Prozesses

Analysephase

- Beschreibung der allgemeinen Ingenieuraufgabe
  - Beispiel: Entwicklung einer Funktion zum automatisierten Fahren auf Basis von Kamerasignalen
- Beschreibung der spezifischen Entwicklungsaufgabe und der Ziele
  - Beispiel: Erkennung von kritischen Verdeckungen beim Rechtsabbiegen und beim Kreuzen von Fußgängerüberwegen
- Festlegung der allgemeinen Anforderungen und Ziele für den Lösungsansatz
  - Beispiel: Verwendung der Simulation, Berücksichtigung des Effekts X bei der Verdeckung in der Simulation

Anforderungsphase

- Anforderungsermittlung mit Fokus auf den Lösungsansatz
  - Beispiel: Berücksichtigen Sie nur den Öffnungswinkel der Kamera in Ihrer Simulation, die Simulationsumgebung sollte mindestens zehnmal schneller laufen als die Echtzeit

Entwurfsphase

- Festlegung, wie der Lösungsansatz auf Entwurfsebene realisiert werden soll
  - Beispiel: Im Kameramodell soll der Effekt X mit dem Ansatz Z modelliert werden

Implementierungsphase

- Implementierung des Lösungsansatzes
- Verifikation des Lösungsansatzes
    - Beispiel: Erstellung und Test des Kameramodells, Integration in die Simulationsumgebung

Ausführungsphase

- Ausführung des Lösungsansatzes
    - Beispiel: Durchführung der Simulation

Auswertungsphase

- Auswertung der Ergebnisse
    - Beispiel: Auswertung der Simulationsergebnisse

Entscheidungsphase

- Entscheiden, ob die Anforderungen der Aufgabe erfüllt wurden
    - Beispiel: Basierend auf den Simulationsergebnissen sind die Anforderungen erfüllt / nicht erfüllt

Anwendung des Prozesses

Ein Prozess ist in Phasen unterteilt. Eine Phase kann eigentlich nur dann vollständig bearbeitet werden, wenn alle Inputs aus der vorhergehenden Phase in der erforderlichen Form vorliegen (Wasserfallmodell). In der Realität ist diese vollständige Verfügbarkeit von Informationen zu Beginn nicht gegeben. Es gibt **Iterationen** in den Phasen und zwischen den Phasen, in denen die Informationen im Projektverlauf sukzessive angereichert werden. In größeren Projekten ist es insbesondere für die Zusammenarbeit notwendig, stabile, definierte Arbeitsstände und Zwischenergebnisse an definierten Orten zu haben. Die Struktur des Credible Simulation Frameworks zeigt deshalb **nicht die zeitliche Abfolge, sondern den logischen Zusammenhang** der Tätigkeiten und deren Ergebnisse und ist damit die Basis für die Nachverfolgbarkeit während eines Projektes und nach dessen Abschlusses.

2.2.1.3 Modellaustauschprozess, Zusammenarbeit mit Partnern

In Abbildung 70 ist eine simulationsbasierte Zusammenarbeit von Partnern dargestellt. Partner A möchte z. B. eine Auswertefunktion für eine Kamera, die er von Partner B beziehen möchte, entwickeln. Die Entwicklung erfolgt simulationsbasiert, da z. B. die reale Kamera noch nicht verfügbar ist. Partner A erstellt die für diese Simulationstask notwendige Simulationsumgebung, spezifiziert die Anforderungen an das Kameramodell (welche Effekte sollen abgebildet sein, Schnittstellen …). Diese Anforderungen werden mit Partner B abgestimmt und als Simulation Request übergeben. Partner B erstellt das Modell und übergibt es Partner A.

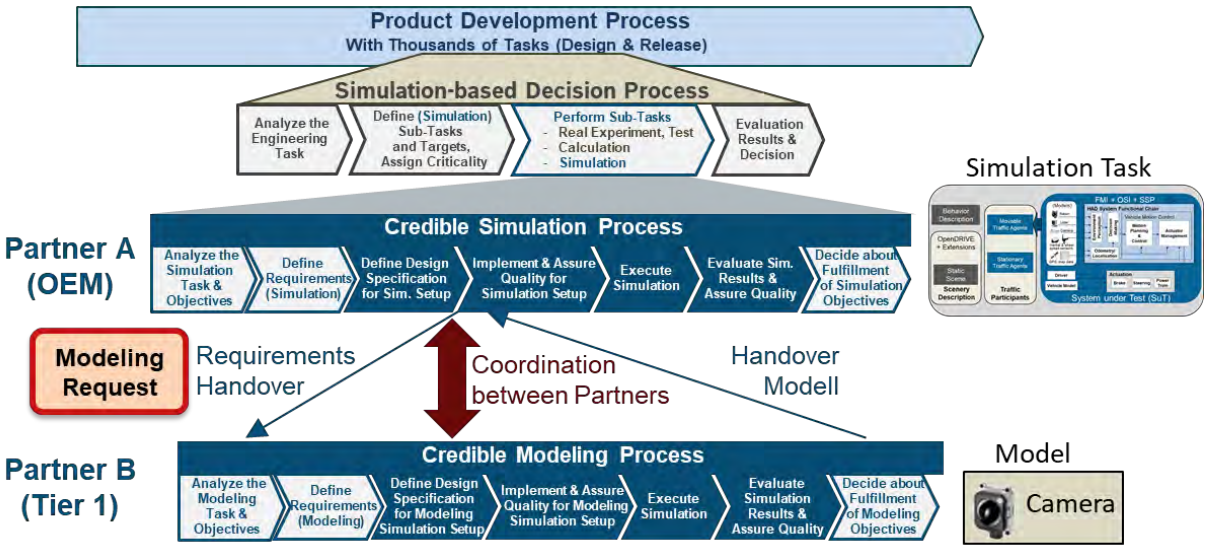


Abbildung 70: Konsistenz von Prozesskette und Informations-/Datenkette

In Kapitel 2.2.3.1.8 ist dieser Zusammenarbeitsprozess detailliert beschrieben. Es wurden zusammen mit TP 5 Industrieworkshops durchgeführt, in welchen die Industrieerfahrungen bei der simulationsbasierten Zusammenarbeit gesammelt, Verbesserungspotentiale identifiziert und entsprechende Verbesserungsvorschläge erarbeitet wurden (siehe https://gitlab.setlevel.de/open/processes_and_traceability/credible_simulation_process_framework Anlage CSP Framework Annex A Work Products).

2.2.1.4 Traceability Roundtrip und Informationskonsistenz

Im Folgenden wird detaillierter auf für die Traceability wichtige Aspekte der Informationskonsistenz in der Zusammenarbeit mit Partnern eingegangen. Wenn ein Partner die Simulation Task allein durchführt, ist in der Regel kein Bruch in der Informationskette gegeben, da ein einheitliches durchgängiges Datenmanagementsystem vorhanden ist. Es muss nun sichergestellt werden, dass, wenn z. B. Partner B das Kameramodell erstellt und dies von Partner A verwendet wird, dies zu keinem Bruch in der Informationskette in der Phase Implementierung führt (durch roten Blitz angedeutet).

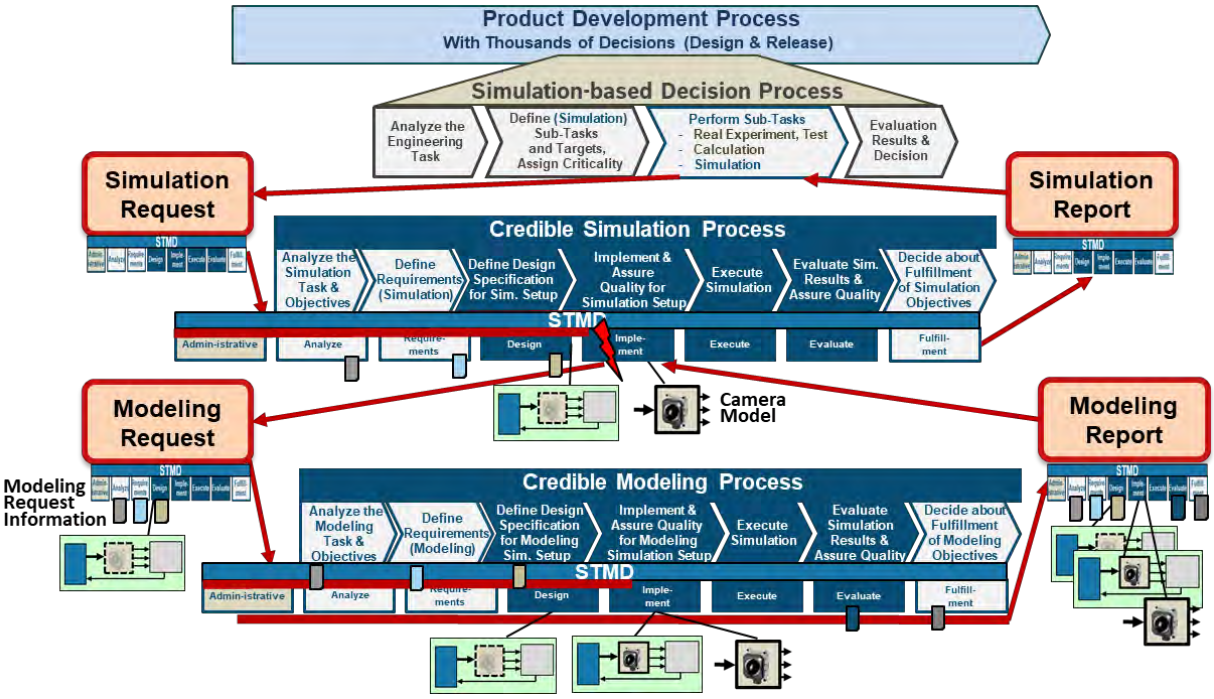


Abbildung 71: Traceability Roundtrip bei der Zusammenarbeit mit Partnern

In der Phase “Implement & Assure Quality for the Simulation Setup” werden die für die Erstellung des Kameramodells notwendigen Informationen gesammelt und als Modeling Request ausgeleitet und dem Credible Modeling Process übergeben. Der Request sollte die gleiche Struktur wie das Datenschema des dazugehörigen Prozesses haben, Dieser Ansatz hat den Vorteil, dass bei der Einbindung eines Requests die Informationen von den Tools direkt in die entsprechenden Stellen des Prozessschemas zugewiesen werden können und spezifische Umformatierungen vermieden werden (siehe Abbildung 71). Eine detaillierte Beschreibung dazu ist unter https://gitlab.setlevel.de/open/processes_and_traceability/credible_simulation_process_framework Anlage Simulation & Modeling Request enthalten.
Derzeit werden üblicherweise Designvorgaben (Schnittstellen, Parameter …) für die Simulationsmodelle rein textuell übergeben und werden dann in der Implementierung entsprechend händisch umgesetzt. Im Projekt SET Level wurde auch der Einsatz der Modelica Standards SSP (System Structuring and Parametrization) erprobt. Der SSP Standard ermöglicht es, dass Modellstrukturen (Verschaltung von Modellen) und deren Parametrierung in einer standardisierten Form abgebildet und so auch Modellstrukturen und deren Parametrierung zwischen unterschiedlichen Simulationswerkzeugen ausgetauscht werden können. Weiterhin können im SSP Standard auch die „Modellhüllen“ mit der Festlegung der Schnittstellen, als Designspezifikation erstellt, übergeben und direkt bei der Implementierung verwendet werden. Damit entfällt bei der Modellimplementierung dieser händische Transformationsschritt von der textuellen Schnittstellenbeschreibung zur Implementierung. Dies bietet nun die Möglichkeit, dass vom Partner A auch schon eine Modell-Verschaltung mit der „Hülle“ des Kameramodells als Testumgebung Partner B übergeben kann (in Abbildung 71 mit dem strichlierten Kameramodell angedeutet). Die im Modeling Request Container enthaltenen Informationen werden dann in die entsprechenden Phasen des Credible Modeling Prozesses übergeben und das Modell entwickelt und getestet. Die für die Traceability notwendigen Informationen werden dann im Modeling Report Container zusammengefasst und Partner A geschickt. Durch diese Vorgehensweise kann nun die Traceability und Informationskonsistenz auch in der Zusammenarbeit von Partnern effizient realisiert werden.

Ein Schwerpunkt im Projekt SET Level war die Entwicklung und Anwendung dieser Prozesse. Für die Unterstützung durch Tools hat, in Entsprechung zu den Prozessschemata des CSP und CMP, das Modelica "System Structure and Parametrization (SSP)"-Projekt zusammen mit dem prostep ivip Projekt Smart SE ein XML-basiertes Datenschema (STMD (Simulation Task Meta Data) oder Glue Particle) entwickelt, das in TP 4 im Demonstrator TRACY umgesetzt wurde. Damit sind diese Prozesse nicht nur als Spezifikation verfügbar, sondern konnten auch mit dieser Implementierung in TRACY erprobt werden. Die Einbeziehung weiterer Organisationen war zum einen wegen des weiteren Feedbacks aus der Industrie als auch wegen der Verbreitung der Ansätze wichtig.

2.2.1.5 Modelle im Kontext Simulation von automatisierten Fahrfunktionen

Im Unterschied zum bisherigen Einsatz von Simulation wird die Entwicklung von automatisierten Fahrfunktionen stärker im Rahmen von Kooperationen stattfinden. Es werden deutlich mehr Modelle ausgetauscht, als auch Modelle von Anbietern angeboten (besonders im Bereich Umweltmodelle, Szenarien). Zur Unterstützung der Suche und Auswahl geeigneter Modelle wurde ein Modellauswahl- und Austauschprozess erstellt (siehe Abschnitt 2.2.3.1.9).
Im Rahmen des Projektes wurden dazu die typischen Use Cases erarbeitet, wie sie bei der Modellauswahl vorkommen. Dies geht von „Sind Modelle prinzipiell für meine Anwendung vorhanden?“ bis zur Unterstützung im konkreten Einsatz. Daraus wurde eine Methodik zur Vorgehensweise bei der Modellauswahl abgeleitet. Dies war dann die Basis zur Festlegung der erforderlichen Metadaten, die durch die Modelle bereitgestellt werden müssen, um den Prozess der Modellauswahl und Integration sicherzustellen. Als Datenformat für die Metadaten wurde das SRMD-Format (Simulation Resource Meta Data) entwickelt. Dieses Datenformat wurde entsprechend auch für die Metadaten von Tools, Karten & Szenarien, Parameter verwendet. Das SRMD-Format ist ein Subset des oben beschriebenen STMD-Formats (Simulation Task Meta Data).
Diese Datenformate wurden von dem in TP 4 entwickelten Tool TRACY erfolgreich pilothaft implementiert und sind die Basis für Traceability, Such- und Auswertefunktionen. Für die Anwendbarkeit speziell in der Zusammenarbeit mit Partnern und Verwendung unterschiedlicher Toolumgebungen ist es wichtig, dass sowohl die Datenformate als auch die Festlegung der Metadaten mit ihrer Semantik eine breite Akzeptanz und Verbreitung finden. Das im Projekt SET Level für die Sensormodelle festgelegte Set an Metadaten fließt inzwischen in die Erweiterung der ISO 11010-x ein.
Zur Unterstützung der Entwicklung und Prüfung der Simulationsmodelle wurde ein Set an Tools wie ein Simulink OSI Wrapper, ein Streckengenerator und Standard Checker entwikkelt.
Schwerpunkt in TP 3 war die Entwicklung der benötigten Simulationsmodelle. Es zeigte sich, dass die Einführung der Simulation Use Cases im Projekt dabei hilfreich war. Aus den Simulation Use Cases konnten die Anforderungen an die Modelle spezifiziert und Rekursionsschleifen zwischen Simulationsaufgabe (Prozess) und Modellerstellung erprobt werden. Es konnte auch erprobt werden, wie und bis zu welchen Grenzen eine Wiederverwendung von Modellen in verschiedenen Simulationsaufgaben sinnvoll ist und wo spezifische Varianten der Modelle erstellt werden sollten. Die Modelle stehen als Open Source Model Library öffentlich zur Verfügung. Zu jedem Modell ist zur Unterstützung der Suche und Nutzung ein Metadatensatz im SRMD-Format erstellt worden.

2.2.2 Ergebnisbeiträge von TP 3

Im Wesentlichen sind in TP 3 drei Ergebnispakete entstanden

2.2.2.1 Process Framework.

Das Credible Simulation Process Framework umfasst alle für die Erstellung und Durchführung einer Simulation und deren Einbindung in Firmenprozesse notwendigen Teilprozesse und Elemente. Spezieller Wert wurde dabei auf die Nachverfolgbarkeit und Nachvollziehbarkeit mit Fokus auf Zusammenarbeit mit Partnern gelegt.

Beiträge aus AP 3.1
https://gitlab.setlevel.de/open/processes_and_traceability/credible_simulation_process_framework

- Credible Simulation Process (CSP)
- Credible Simulation Process mit Beispiel
- Credible Modeling Process (CMP)
- Credible Modeling Process mit Beispiel
- Simulation-based Decision Process
- Simulation und Modeling Request
- Vereinbarungsprozess zwischen Partnern
- Kooperationsprozess (Zusammenwirken des CSP und CMP)
- Credible Simulation Process Framework Annex
  mit einer Auflistung der Arbeitsprodukte aus AP 3.2
- Modellauswahl- und Austauschprozess

2.2.2.2 Metadatenformate und Metadaten

Für die Umsetzung der Prozesse in Workflows und Suche nach Artefakten wie Modellen speziell in heterogenen IT-Infrastrukturen und in Zusammenarbeit mit Partnern werden standardisierte Metadatenformate und Metadaten benötigt. Im Rahmen von SET Level sind Folgende erstellt bzw. erweitert und erprobt worden.
Beiträge aus AP 3.2

- Simulation Resource Metadata (SRMD) Format
- Simulation Task Metadata (STMD) Format
- STMD Format für Simulations- und Modellierungs-Tasks
- STMD Format für Beauftragungen

Aus AP 3.3

- SRMD für Modelle und Templates
- SRMD für Tools und Templates

Aus AP 3.3 und AP 4.1

- SRMD für Karten & Szenarien und Templates

2.2.2.3 Simulationsmodelle und Best Practices

Ein Schwerpunkt in TP 3 war die Erstellung der Simulationsmodelle, die in TP 4 und den Simulation Use Cases benötigt und eingesetzt wurden. Diese Modelle wurden als Open Source Modelle erstellt.
Beiträge aus AP 3.1.2

- Open Source Model Library
  - Modell
    - Doc (CMP user view)
    - SRMD

Aus AP 3.2 und AP 3.3

- Tools für Modelle
  - OSMP
  - Simulink OSI Wrapper
  - Streckengenerator
  - Standard Checker

2.2.3 Detaillierte Ergebnisbeiträge von TP 3

2.2.3.1 Process Framework

Das Credible Simulation Process Framework umfasst alle für die Erstellung und Durchführung einer Simulation und deren Einbindung in Firmenprozesse notwendigen Teilprozesse und Elemente. Spezieller Wert wurde dabei auf die Nachverfolgbarkeit und Nachvollziehbarkeit mit Fokus auf Zusammenarbeit mit Partnern gelegt.

2.2.3.1.1 Credible Simulation Process (CSP)

Die Dokumentation des CSP umfasst die Beschreibung eines generischen Engineering-Prozesses, die Zielgruppe der Nutzer des Credible Simulation Processes, eine detaillierte Beschreibung des CSP, seiner Anwendung und Verbindung zu übergeordneten Prozessen. Der CSP bietet einen Rahmen für die Verwendung von Methoden, legt aber nicht fest, welche Methoden verwendet werden sollten (siehe https://gitlab.setlevel.de/open/processes_and_traceability/credible_simulation_process_framework/-/blob/main/Credible-Simulation-Process-v1-3.pdf, siehe Abbildung 72).

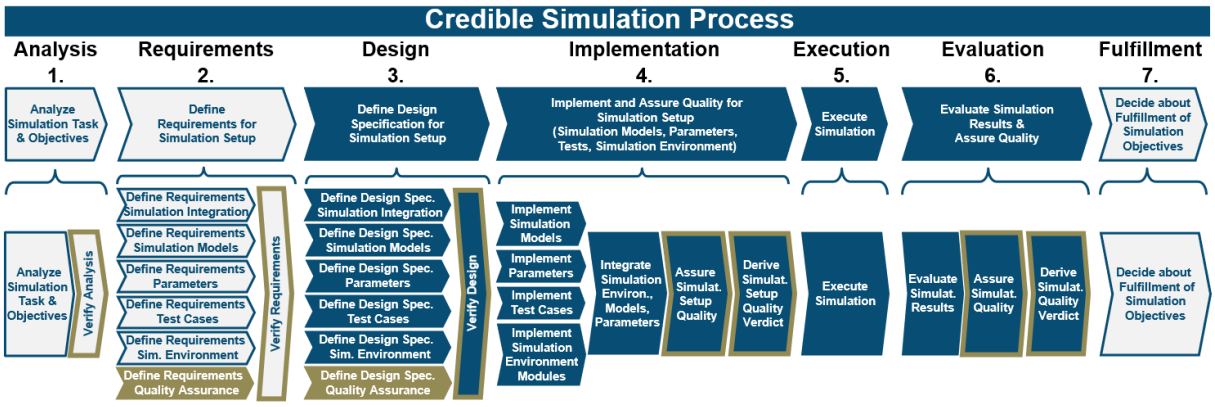


Abbildung 72: Abbildung des Aufbaus des Credible Simulation Processes

2.2.3.1.2 Credible Simulation Process mit Beispiel

Wie der Credible Simulation Process angewandt werden kann, wird anhand eines einfachen Beispiels gezeigt (siehe https://gitlab.setlevel.de/open/processes_and_traceability/credible_simulation_process_framework/-/blob/main/Credible-Simulation-Process-Example-v1-1.pdf).

2.2.3.1.3 Credible Modeling Process (CMP)

Die Dokumentation des CMP umfasst die Beschreibung des Credible Modeling Processes, seine Verbindung zum CSP mit einem Beispiel, eine Erläuterung wie die Konsistenz des CMP mit der Modelldokumentation hergestellt werden kann, sowie Umgang mit dem Intellectual Property-Schutz (siehe https://gitlab.setlevel.de/open/processes_and_traceability/credible_simulation_process_framework/-/tree/main, siehe Abbildung 73).

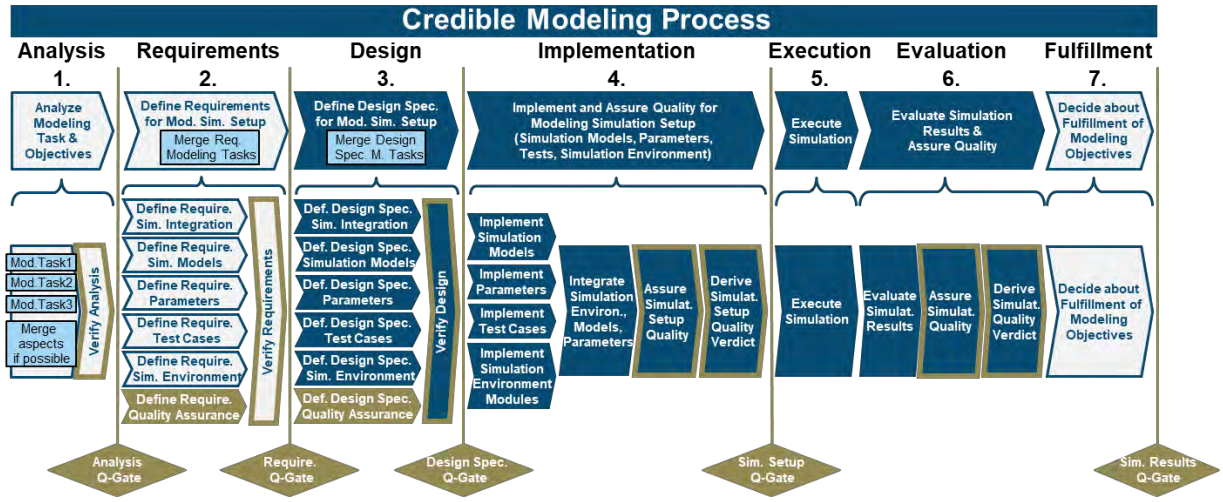


Abbildung 73: Abbildung des Aufbaus des Credible Modeling Processes

2.2.3.1.4 Credible Modeling Process mit Beispiel

Wie der Credible Modeling Process angewandt werden kann, wird anhand eines einfachen Beispiels gezeigt (siehe https://gitlab.setlevel.de/open/processes_and_traceability/credible_simulation_process_framework/-/blob/main/Credible-Modeling-Process-Example-v1-1.pdf).

2.2.3.1.5 Simulation-based Decision Process

Die Dokumentation umfasst die Beschreibung des dem Credible Simulation Processes übergeordneten Prozesses, soweit er für die Definition der Übergänge zum CSP notwendig ist. Dabei werden die Simulationsaufgabe und -ziele für den CSP aus den Anwendungssituationen (z. B. Analyse, Optimierung, Test) abgeleitet. Link zu Bericht Simulation-based Decision Process: https://gitlab.setlevel.de/open/processes_and_traceability/credible_simulation_process_framework/-/blob/main/Simulation-based-Decision-Process-v1-1.pdf, siehe Abbildung 74.

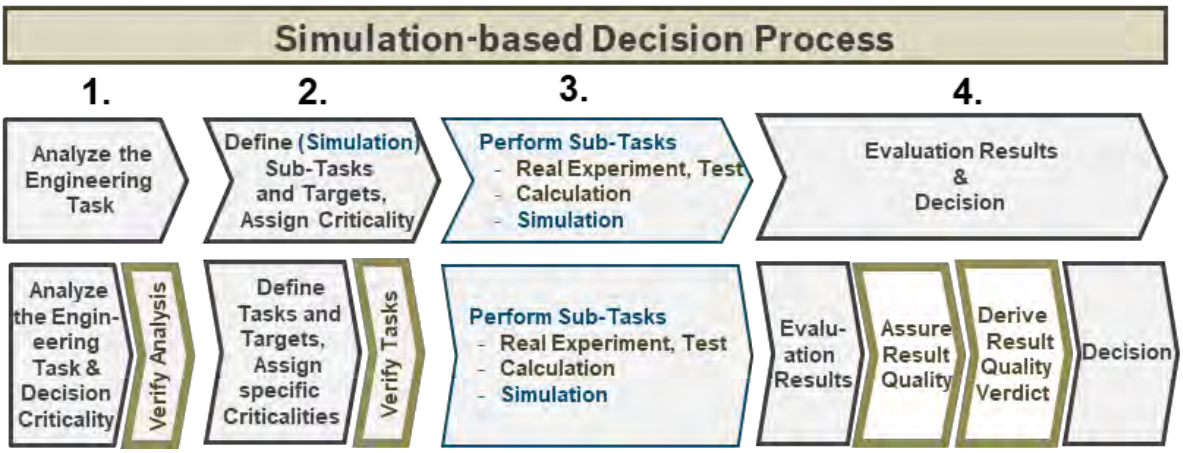


Abbildung 74: Abbildung des Aufbaus des Simulation-based Decision Processes

2.2.3.1.6 Simulation und Modeling Request

Der Simulation and Modeling Request soll einen Rahmen für Kunden und Dienstleister definieren, in dem alle relevanten Aspekte für die Definition und Erfüllung eines "Request" berücksichtigt werden, um u. a. das Requirements Engineering so effizient wie möglich zu gestalten. Link zu Bericht Simulation und Modeling Request: https://gitlab.setlevel.de/open/processes_and_traceability/traceability_data/-/blob/main/metadata_for_simulation_tasks_and_requests/SimRequests_Tasks_Traceability-v1-1.pdf, siehe Abbildung 75.

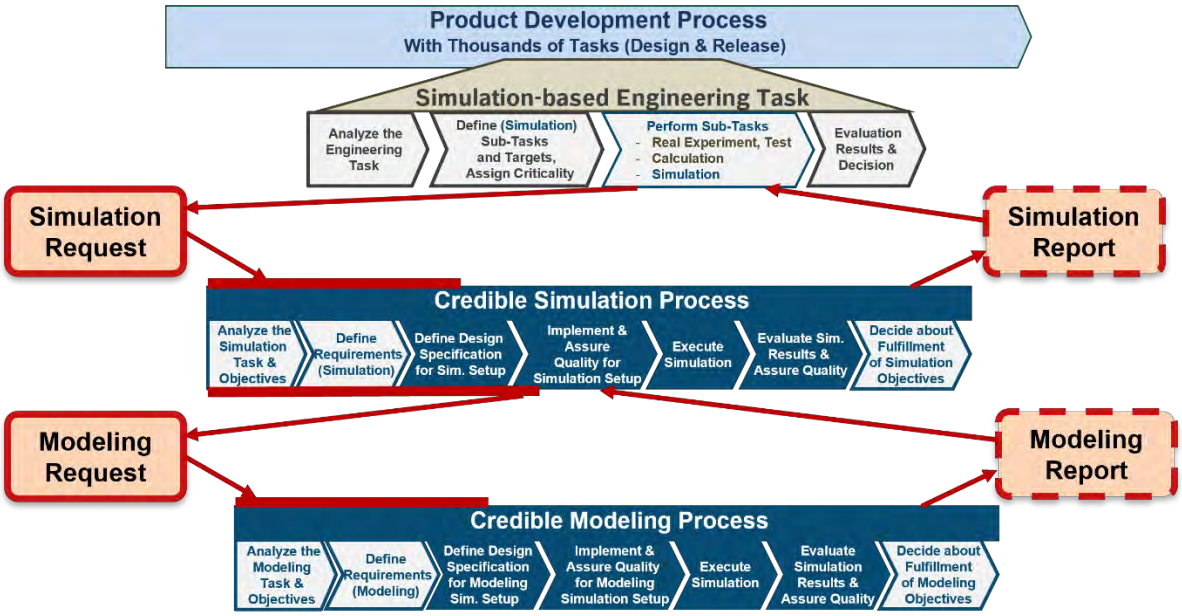


Abbildung 75: Einbindung der Simulations- und Modellierungsbeauftragung in das Prozess-Framework

2.2.3.1.7 Vereinbarungsprozess zwischen Partnern

Im Vereinbarungsprozess zwischen Partnern werden die rechtlichen und inhaltlichen Aspekte, die vor Beginn eines gemeinsamen Simulations- oder Modellierungsprojekts zwischen den Partnern zu definieren sind, beschrieben. Link zu Bericht Vereinbarungsprozess zwischen Partnern: https://gitlab.setlevel.de/open/processes_and_traceability/credible_simulation_process_framework/-/blob/main/Agreement-Process-Simulation-Task-v1-1.pdf, siehe Abbildung 76.

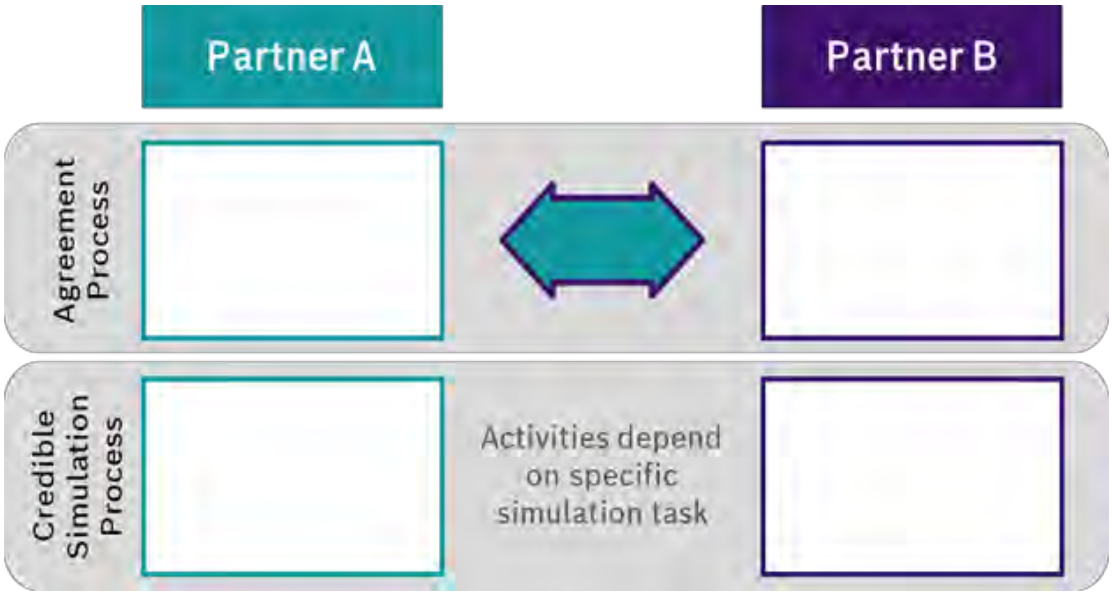


Abbildung 76: Vereinbarungsprozess zwischen Partnern, vorgelagert zum Credible Simulation Process

2.2.3.1.8 Kooperationsprozess (Zusammenwirken des CSP und CMP)

Die Dokumentation umfasst das Zusammenspiel des CSP und CMP mit dem prostep Referenzprozess in Bezug auf Nachvollziehbarkeit, Iterationen, Rollenverteilung in einer kooperativen Zusammenarbeit zwischen Partnern. Link zu Bericht Kooperationsprozess: https://git-lab.setlevel.de/open/processes_and_traceability/credible_simulation_process_framework/-/blob/main/Cooperation-Process-v1-1.pdf, siehe Abbildung 77.

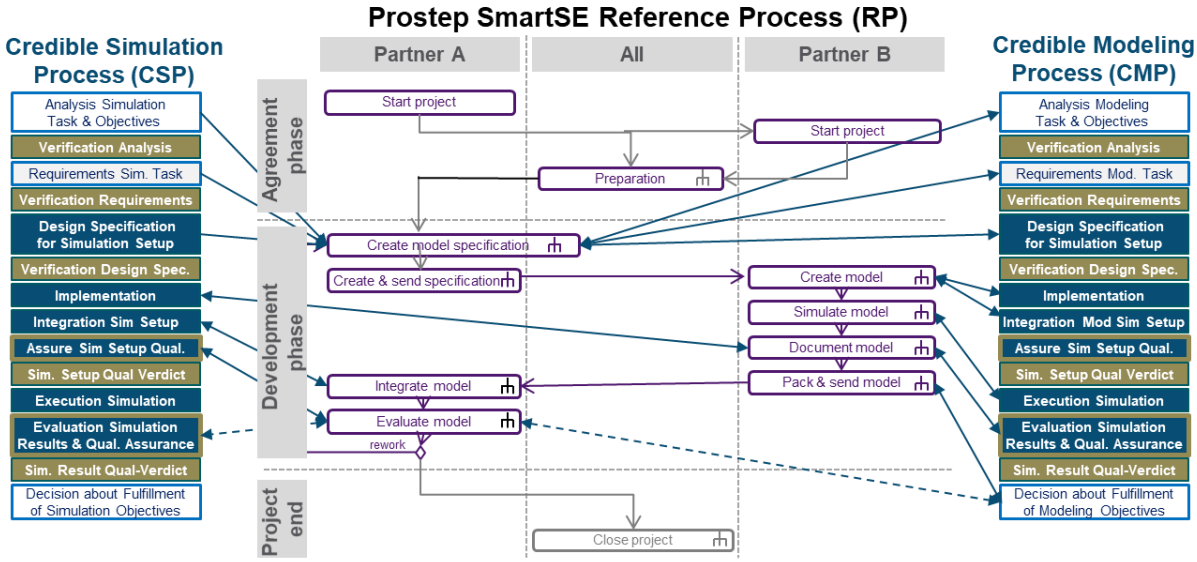


Abbildung 77: Zusammenspiel des CSP & CMP mit dem prostep Referenzprozess

2.2.3.1.9 Modellauswahl- und Austauschprozess

Die Dokumentation umfasst die Prozessbeschreibung für die Auswahl geeigneter Modelle und Aspekte, die beim Austausch zu berücksichtigen sind, z. B. IP-Schutz.
Als sensorspezifische Erweiterung des Modellauswahl- und Austauschprozesses wird eine Sensormodellklassifikation vorgestellt, die verschiedene Klassen von Sensormodellen

beschreibt. Link zu Bericht SET Level Model Echange and Selection Process: https://gitlab.setlevel.de/open/processes_and_traceability/credible_simulation_process_framework/-/blob/main/SETLevelModelExchangeAndModelSelectionProcesses.pdf, siehe Abbildung 78.

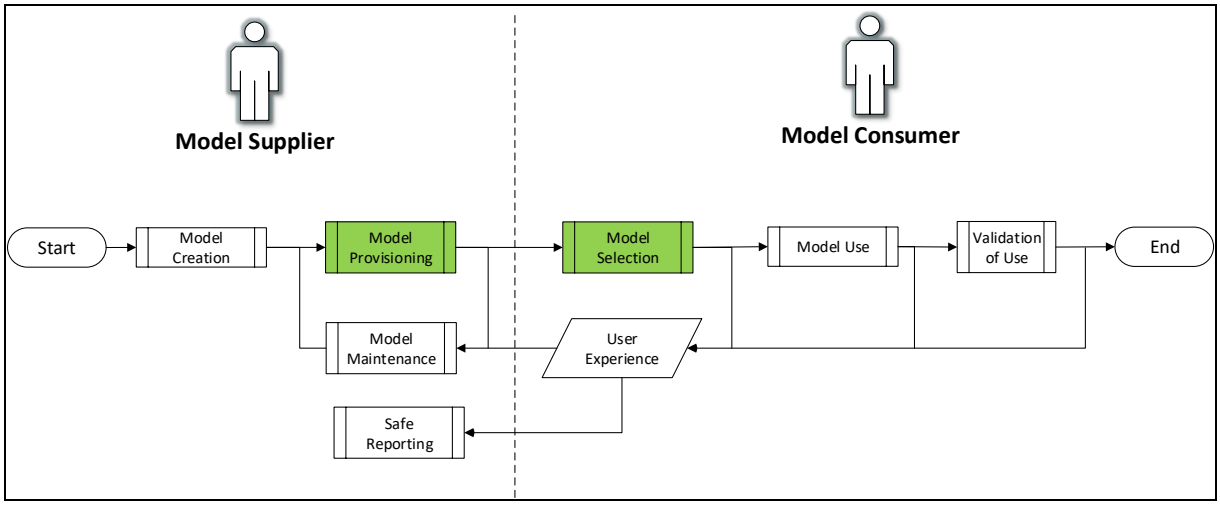


Abbildung 78: Zusammenspiel Modellersteller, Modellkonsument

2.2.3.1.10 Credible Simulation Process Framework Annex mit einer Auflistung der Arbeitsprodukte

Beschreibung der Arbeitsprodukte mit Zusatzinformationen: Abkürzungen, Rollen, Checkliste für CSP, Best Practice des Zusammenarbeits-Prozesses. Link zu Annex: https://gitlab.setlevel.de/open/processes_and_traceability/credible_simulation_process_framework/-/blob/main/CSP_Framework_Annex_A_Work_Products.pdf.

2.2.3.2 Metadatenformate und Metadaten

Für die Umsetzung der Prozesse in Workflows und die Suche nach Artefakten, wie Modelle speziell in heterogenen IT-Infrastrukturen und in Zusammenarbeit mit Partnern, werden standardisierte Metadatenformate und Metadaten benötigt. Im Rahmen von SET Level sind folgende Metadatenformate und Metadaten erstellt, bzw. erweitert und erprobt worden.

2.2.3.2.1 Simulation Resource Metadata (SRMD) Format

Zur Unterstützung der Rückverfolgbarkeit von Simulationen sowie der Prozessautomatisierung, z. B. der Auswahl eines bestimmten Modells unter Verwendung von Informationen auf einer höheren Ebene über dieses Modell, wurde im Projekt SET Level ein Format definiert, der erste Sets von Metadaten in einem maschinenlesbaren Datenformat konsolidiert. Das Datenformat mit der Bezeichnung Simulation Resource Meta Data (SRMD) ist ein XML-Schema, das die Klassifizierungsfunktionen des SSP-Traceability-Ansatzes (siehe https://pms-fit.github.io/SSPTraceability/master/#_simulation_resource_meta_data_file_format) bündelt. Link zu Bericht: https://gitlab.setlevel.de/open/processes_and_traceability/traceability_data/-/blob/main/SETLevel_SRMD_and_classifications_for_metadata.pdf, siehe Abbildung 79.

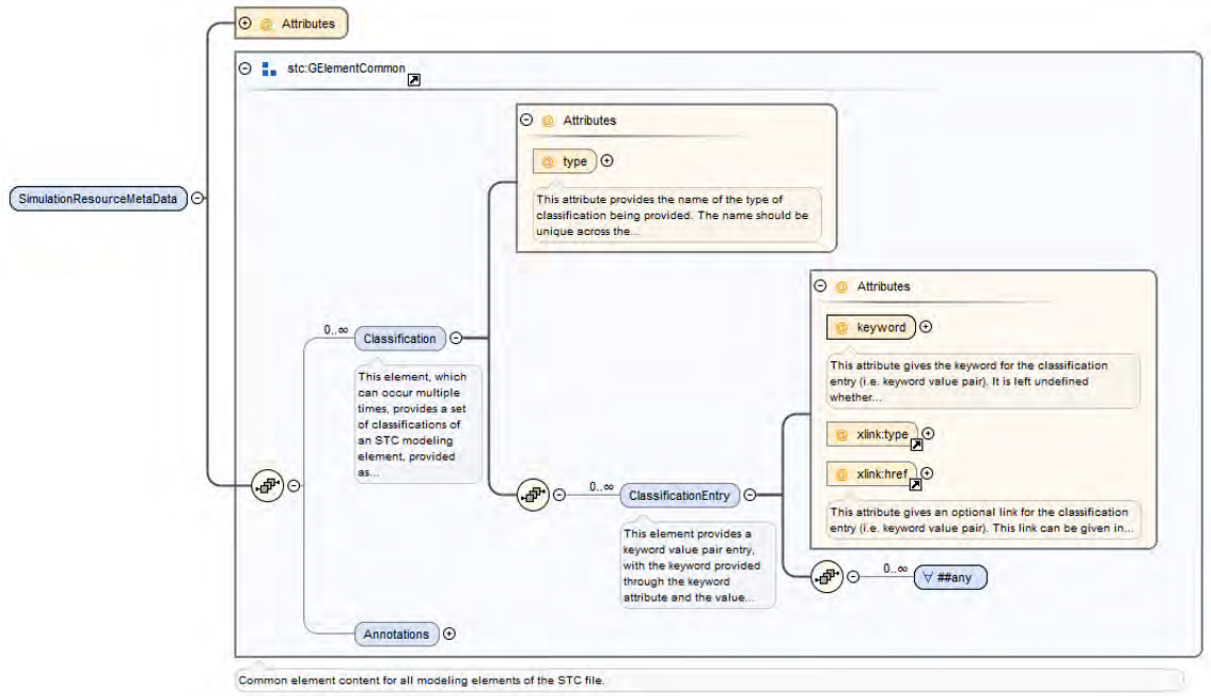


Abbildung 79: Simulation Resource Meta Data (SRMD) Schema

2.2.3.2.2 Simulation Task Metadata (STMD) Format

Das in Abschnitt 2.2.3.2.1 beschriebene SRMD Format stellt eine Teilmenge des Simulation Task Metadata (STMD) Formats dar. Über das STMD Format wird die Maschinenlesbarkeit und Rückverfolgbarkeit von kompletten Simulations- oder Modellierungsaufgaben unterstützt. Im Rahmen von SET Level wurde dieses Format erprobt. Die Ergebnisse flossen in die Weiterentwicklung des STMD Formats ein. Link zur STMD-Formatspezifikation: https://github.com/PMSFIT/SSPTraceability/blob/master/specification/4___STMD.adoc. Das STMD und das Subset SRMD Format werden von der Modelica Association im Rahmen des SSP-Projekts öffentlich frei zur Verfügung gestellt und weiterentwickelt.

2.2.3.2.3 STMD Format für Simulations-, Modellierungs-Tasks und Beauftragungen

Die Anwendung des STMD Formats auf Simulations- und Modellierungstasks ist in der SSP Traceability Specification (siehe https://pmsfit.github.io/SSPTraceability/master/) beschrieben und öffentlich verfügbar. Für Simulations- oder Modellierungsbeauftragungen (Requests) wird ebenfalls dieses STMD Format verwendet.

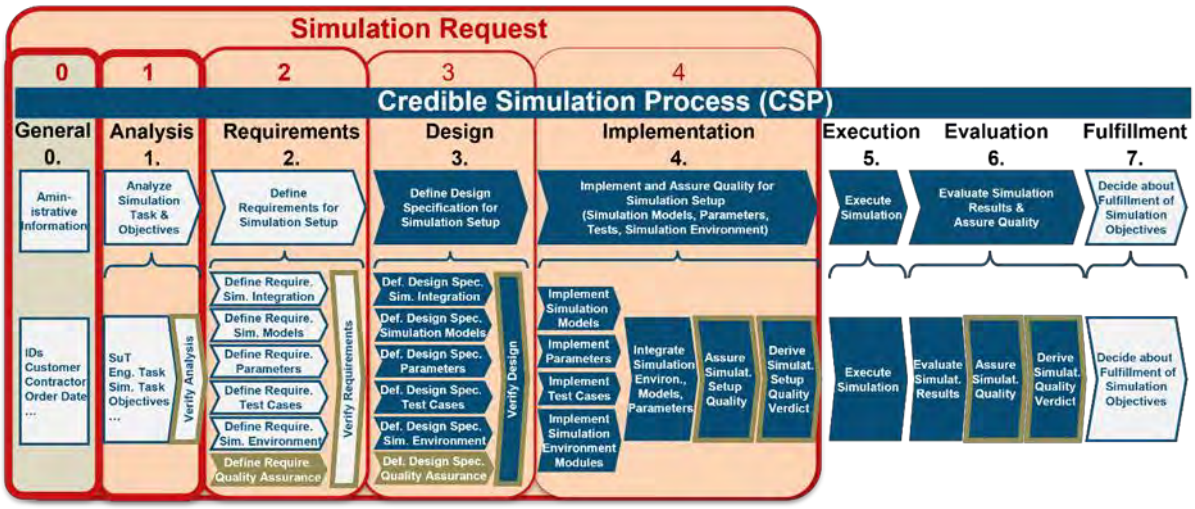


Abbildung 80: Simulation Request als „teilbefüllter“ Credible Simulation Process

In Abbildung 80 ist dargestellt, dass ein Request die gleiche Struktur wie das Datenschema des dazugehörigen Prozesses hat, dieser aber nur teilweise befüllt ist. Dieser Ansatz hat den Vorteil, dass bei der Einbindung eines Requests die Informationen von den Tools direkt in die entsprechenden Stellen des Prozessschemas zugewiesen werden können und spezifische Umformatierungen vermieden werden.

2.2.3.2.4 SRMD für Modelle und Templates

Die Nutzung von SRMD (Simulation Resource Meta Data) für Modelle soll den Modellauswahlprozess unterstützen. Zwei Klassifikationen sind für die Beschreibung von konkreten Modellmetadaten definiert worden.
Zum einen ist eine generische Klassifikation **„de.setlevel.srmd.model-meta-data“** definiert worden. Diese stellt Metadaten zur Durchführung des Credible Modelling Process bereit, also alle übergeordneten Informationen über das Modelloriginal und Zielnutzung aber auch Metadaten über die Maturität des Modells, ob es z. B. qualifiziert ist. Enthalten in dieser generischen Klassifikation sind auch spezifische Metadaten für die im Projekt SET Level erarbeitete Modelltypen: Agentenmodelle, Sensormodelle, Fahrzeug- und Aktorikmodelle.
Zum anderen ist eine ISO spezifische Klassifikation **„de.setlevel.srmd.ISO-11010-X“** definiert worden. Diese Klassifikation gibt die Möglichkeit die in der ISO-11010 definierte Nomenklatur zu Modelltypen in Abhängigkeit ihres Einsatzbereiches als Metadaten zu nutzen. Die genaue Definition dieser Klassifikationen und deren spezifizierten Metadaten sind veröffentlicht worden und unter https://gitlab.setlevel.de/open/processes_and_traceability/traceability_data/-/blob/main/SETLevel_SRMD_and_classifications_for_metadata.pdf zu finden. Diese sind zum prostep ivip Projekt SmartSE zur weiteren Standardisierung transferiert worden. Ein Template wurde erstellt und veröffentlicht unter https://gitlab.setlevel.de/open/processes_and_traceability/traceability_data/-/tree/main/metadata_format_for_models.

2.2.3.2.5 SRMD für Tools und Templates

Die Nutzung von SRMD für Tools soll die Nachverfolgbarkeit über die gesamte Simulations Toolkette unterstützen. Dafür ist die Klassifikation **"de.setlevel.srmd.tool-meta-data"** definiert worden. Diese Klassifikation enthält Metadaten über das Tool, wozu es gemacht worden ist und ISO 26262 Tool bezogene Informationen. Die genaue Definition von diesen Klassifikationen und deren konkreten Metadaten sind veröffentlicht worden und unter https://gitlab.setlevel.de/open/processes_and_traceability/traceability_data/-

/blob/main/SETLevel_SRMD_and_classifications_for_metadata.pdf zu finden. Eine Template hierfür wurde erstellt und veröffentlicht unter https://gitlab.setlevel.de/open/processes_and_traceability/traceability_data/-/blob/main/metadata_for_tools/OverallToolSRMDTemplate.srmd.

2.2.3.2.6 SRMD für Karten & Szenarien und Templates

Die Nutzung von SRMD für Karten und Szenarien soll das Finden von Karten und Szenarien in einer Datenbank unterstützen. Dafür ist die Klassifikation **"de.setlevel.srmd.trafficspaceandscenario-meta-data"** definiert worden. Diese Klassifikation enthält Metadaten über typische Eigenschaften einer Szene, wie z. B. der Anzahl und Typen von Verkehrsteilnehmern aber auch der Art von Fahrmanövern. Die genaue Definition von diesen Klassifikationen und deren konkreten Metadaten sind veröffentlicht worden und unter https://gitlab.setlevel.de/open/processes_and_traceability/traceability_data/-/blob/main/SETLevel_SRMD_and_classifications_for_metadata.pdf zu finden. Templates dazu wurden erstellt und veröffentlicht unter https://gitlab.setlevel.de/open/processes_and_traceability/traceability_data/-/tree/main/metadata_for_trafficspaces_and_scenarios.

2.2.3.3 Simulationsmodelle und Best Practices

Ein Schwerpunkt in TP 3 war die Erstellung der Simulationsmodelle, die in AP4 und den Simulation Use Cases benötigt und eingesetzt wurden. Diese Modelle wurden als Open Source Modelle erstellt.

2.2.3.3.1 Open Source Model Library Overview

Ein Ziel in TP 3 war die Erstellung einer Open source model library. Diese beinhaltet die im Projekt SET Level erarbeiteten Simulationsmodelle und stellt ein kohärentes Set von Modellen bereit, die mit der in TP 2 erarbeitete Simulationsarchitektur angewendet werden können. Die Modellbibliothek kann auch als Startpunkt von weiteren Open source Projekten nach Ende des Projekts SET Level dienen. Die Model Library ist das Hauptergebnis des UAP 3.1.2, sie ist auch öffentlich verfügbar (siehe https://gitlab.setlevel.de/open/models) und beinhaltet:

- Traffic Participants models (https://gitlab.setlevel.de/deliverables/model/model_library_overview/-/blob/main/README.md#traffic-participants-models)
- Perception Sensor models of the EGO vehicle (https://gitlab.setlevel.de/deliverables/model/model_library_overview/-/blob/main/README.md#perception-sensors-models)
- Vehicle dynamics models and actuator models of the EGO vehicle (https://gitlab.setlevel.de/deliverables/model/model_library_overview/-/blob/main/README.md#vehicle-dynamics-models-and-actuator-models)

Erkenntnisse aus den partnerübergreifenden Abstimmungen in den jeweiligen Modellierungsdomänen sind dabei zusammengebracht und konsolidiert worden. Dabei sind Sets von relevanten Anforderungen, Methoden zur Ableitung dieser Anforderungen, Modellierungsansätze, Evaluierungsmethoden, aber auch Lessons learned zur Modellintegration entstanden. Aus diesen Arbeiten haben sich auch mehrere Publikationen ergeben, die weiter unten (siehe Abschnitt 2.2.3.3.5) aufgelistet sind.

Alle Modelle der Library sind in der Form einer FMU nach FMI Standard (siehe https://fmi-standard.org/about/) entwickelt worden. Die meisten davon unterstützen auch ASAM OSI (siehe https://www.asam.net/standards/detail/osi/) zusammen mit FMIs. Jedes Modell wird als eine Simulationsressource Teil eines Datamanagementsystems betrachtet, dafür sind auch High Level Informationen mit dem SET Level Model Metadata (siehe https://gitlab.setlevel.de/open/processes_and_traceability/traceability_data/-/blob/main/metadata_format_for_models/README.md) Ansatz dokumentiert.

2.2.3.3.2 Traffic Participants Models

Erarbeitete Modelle

Traffic Participants Models sind Modelle von Verkehrsteilnehmern, die in einer Szenarienbasierten Simulation zusammen mit dem Ego Fahrzeug interagieren. Drei Traffic Participants sind im Projekt SET Level modelliert worden:

- Ein Fußgängermodell (https://gitlab.setlevel.de/open/models/traffic-participants/osipedestrian)
- Ein Fahrermodell (https://gitlab.setlevel.de/open/models/traffic-participants/driver_model)
- Ein Game-Theory-basiertes Agentenmodell

Diese Modelle sind unter https://gitlab.setlevel.de/open/models/traffic-participants Open source verfügbar. Eine detaillierte Übersicht der Modellierungsaktivitäten ist unter https://gitlab.setlevel.de/open/models/library-overview/-/blob/main/overviews/b1-7d-poster-agent-models.pdf offen verfügbar. Eine Einführung zu den einzelnen Modellen ist unter https://gitlab.setlevel.de/open/models/library-overview/-/blob/main/overviews/b1-7d-slides-agent-models.pdf öffentlich verfügbar.

Herausforderungen

Folgende Herausforderungen sind identifiziert worden:

- Komplexe Verkehrsszenarien und komplexe Interaktionen zwischen Verkehrsteilnehmern fordern flexible und effiziente Traffic Participant Modelle.
- Vorab-Scripting von komplexem Verhalten von Traffic Participant Modellen hat seine technische Grenze erreicht. Diese Modelle brauchen eine eigene „Intelligenz“.
- Die Schnittstelle zwischen der Ausführung von Szenarien und der Ausführung von Verkehrsteilnehmermodellen wird immer wichtiger, um sicherzustellen, dass beide für eine bestimmte beabsichtigte Simulationsaufgabe kompatibel bleiben.
- Hoher anfänglicher Entwicklungsaufwand je nach der für die Szenarien erforderlichen Modelltreue. Die Modelle sollen in der Regel von mehreren Simulationsumgebungen wiederverwendet werden können.

Anforderungen und Spezifikationansätze

Folgende Anforderungen sind identifiziert worden:

Steuerbarkeit und Lane Zuweisung

- Standardisierte Schnittstelle, um Traffic Participant Modelle zu steuern: OpenSCENARIO “actions” sind in neuen OSI TrafficCommands übersetzt worden.
- Einige Traffic Participant Modelle brauchen dedizierte “Lane Types”: Gehwege, Fahrradwege, die auch in Standards einfließen mussten.
- Zur verbesserten und eindeutigeren Spur Zuweisung werden benötigt: logische Spuren, Spuren für Kreuzungen, Spurbeziehungen und Bezug zu Koordinatensystemen.

Generalisierung der Verkehrsteilnehmer Modelstruktur und deren Schnittstellen

- Unterstützung von Modularität und Skalierbarkeit durch Unterscheidung von Verhalten und Dynamik im Modell der Verkehrsteilnehmer. Das Verhaltensmodell enthält die Entscheidungsfindung und andere übergeordnete Aspekte des Verkehrsteilnehmers: Gesten, Einhaltung oder Nichteinhaltung von Verkehrsregeln, Fahrstil, Grad der Steuerbarkeit durch externe Befehle, Fähigkeit, Hindernissen und/oder anderen Teilnehmern auszuweichen und/oder auf sie zu reagieren oder nicht, Fähigkeit, seinen Weg anhand eines Ziels auf einer Karte zu finden, Art der Auswahl von Manövern, ...
- Das Dynamikmodell muss die Realisierung der beabsichtigten Trajektorie und der damit verbundenen Dynamik enthalten: dynamische Steuerung mit offenem oder geschlossenem Regelkreis, Fehlerinjektion ...

Modellierungsansatz

Ein generisches Model Framework ist definiert worden: OSMP for Traffic Participants.

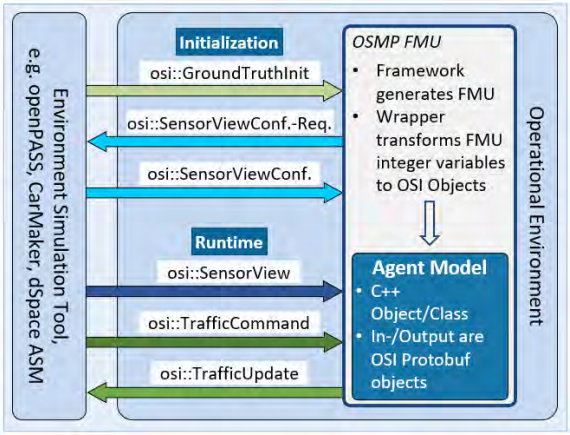


Abbildung 81: Generisches Framework zur Modellstrukturierung und -Integration

Dieses Framework unterstützt die Anforderung an Modellmodularität, und es ist damit möglich, alle drei SET Level Verkehrsteilnehmermodelle jeweils zu integrieren, ohne dafür die äußeren Modellschnittstellen ändern zu müssen.

Zwei Modellierungsansätze sind verfolgt worden:

- Manöverbasierte, in diesem Fall wird ein Manöver komplett oder zum Teil dem Modell vorgegeben, um den Verkehrsteilnehmer zu steuern.

- Spieltheoriebasierte Manöverauswahl, in diesem Fall wird dem Modell kein Manöver vorgegeben, und der Verkehrsteilnehmer darf selbst ein Manöver auswählen.

Eine Beschreibung dieser Ansätze, auf konkrete Modelle appliziert, ist öffentlich verfügbar:

- https://gitlab.setlevel.de/open/models/traffic-participants/osipedestrian/-/blob/master/README.md
- https://gitlab.setlevel.de/open/models/traffic-participants/driver_model/-/blob/main/README.md
- https://gitlab.setlevel.de/open/models/traffic-participants/driver_model_game-theory_based/-/blob/main/README.md

Eine detaillierte Beschreibung der Spieltheoriebasierten Modellierung für die Modellierung von Verkehrsteilnehmer ist in der Publikation Maneuver Based Modeling of Driver Decision Making using Game-Theoretic Planning (siehe https://ieeexplore.ieee.org/document/9658771) veröffentlicht worden.

Evaluationskriterien und -methoden

Um die Qualität der Verkehrsteilnehmer zu bewerten, wurden folgende Einstiegskriterien vorgeschlagen und in den Modellen großenteils umgesetzt:

- Fähigkeit des Modells, ein für ein bestimmtes Szenario angefordertes Manöver zu verstehen und auszuführen: Anfahren, Anhalten, Abbiegen, Geradeausfahren, Überholen, Spurwechsel, Beschleunigen, Abbremsen. Um dies zu testen, werden Einheitstests mit einfachen Referenzszenarien durchgeführt.
- Fähigkeit des Modells, ein Manöver auszuführen und dabei einige vorgegebene dynamische Ziele einzuhalten: eine bestimmte Geschwindigkeit, eine bestimmte Flugbahn.

2.2.3.3.3 Perception Sensor Models

Erstellte Modelle

Berücksichtigt wurden Kamera-, Radar- und Lidarmodelle, wofür drei Typen von Modellen abhängig vom Schnittstellendetaillierungsgrad definiert worden sind:

- Objektbasierte Modelle
- Reflexions- und bildbasierte Modelle
- Geometriebasierte Modelle

Einige dieser Modelle sind unter https://gitlab.setlevel.de/open/models/perception-sensor-models open source verfügbar. Eine Übersicht der Aktivitäten für Sensormodelle ist unter https://gitlab.setlevel.de/open/models/library-overview/-/blob/main/overviews/b1-7b-poster-osi-and-fmu-based-sensor-models.pdf verfügbar. Die drei Typen sind folgendermaßen definiert:

Objektbasierte Modelle

Objekte

- können stationär (Ampeln, Schilder, Markierungen, Hindernisse, ...) und / oder dynamisch (andere Fahrzeuge, Fußgänger …) sein.
- sind durch Umweltmodelle in Simulationstools generiert. Bei dieser Generierung wurden auch Objekteigenschaften bereitgestellt (Position in der Szene, Geschwindigkeit, Orientierung usw.).
- können über den Open Simulation Interface Standard aus einem Sensormodell erzeugt werden.

Diese Modelle werden typischerweise genutzt, um Fahrentscheidungs und -planungsalgorithmen zu definieren oder zu testen.

Reflexionsbasierte und bildbasierte Modelle

Radar-, Lidarreflexionen und Bilder einer Szene

- sind durch das Rendering einer 3D-Szene generiert. Dieses Rendering wird außerhalb des Modells gemacht, im Simulationstool selbst oder in einen 3rd Party Tool.
- können über den Open Simulation Interface Standard an ein Sensormodell übergeben werden.

Diese Modelle werden typischerweise genutzt, um Perzeptionsalgorithmen zu definieren oder zu testen.

Geometriebasierte Modelle

Geometrische Daten,

- sind 3D-Daten einer Szene sowie Materialdaten, die für eine bestimmte Sensortechnologie relevant sind, z. B. OpenMATERIAL (siehe https://github.com/LudwigFriedmann/OpenMaterial/blob/master/README.md).
- sind Modelle der Objekte, z. B. LKW, Fußgänger, aber auch der Infrastruktur, z. B. Straßen, Schilder, liefern ihre 3D- und Materialdaten an den Simulator, der eine geometrische Ansicht der Szene erstellt.
- sind die Eingaben für das Rendering oder Raytracing, bei dieser Modellierungsvariante wird der Rendering- oder Raytracing-Schritt im Modell selbst oder in einem 3rd-Party-Tool durchgeführt.

Geometriebasierte Modelle werden noch nicht durch ein standardisiertes Datenformat unterstützt. Diese Modelle werden in der Regel für die Simulation auf der Ebene des Perceptions-, Umgebungsmodells oder auf der Ebene der Sensorkomponenten verwendet.

Es folgt eine kurze Zusammenstellung der Herausforderungen, die für diese Modelle im AD-Kontext relevant sind.

- Datenrate und Echtzeitfähigkeit

- Erhalt gültiger Renderingergebnisse von Drittanbietern, die im Modell verwendet werden können

Anforderungen und Spezifikationsansätze

Sensor-relevante Effekte

Es wurden typische AD-relevante Effekte für Sensormodelle gesammelt, die in Komponenten- und Systemtests verwendet werden. Einige von ihnen wurden im Projekt SET Level implementiert.

- Dazu wurde eine funktionale Zerlegung eines Wahrnehmungssensors vorgenommen: Emission: aktiver oder passiver Sensor, Primär- und Drittquellen
- Signalausbreitung: die Vorgänge zwischen dem erfassten Objekt und dem Sensor in der Umgebung und ihr Einfluss auf die Erfassung.
- Empfang: die Erfassung des Messsignals durch Optik und Elektronik, aber auch dadurch beeinflusst, dass der Sensor Teil des fahrenden Fahrzeugs ist.
- Vorverarbeitung: die frühe Behandlung der erfassten Signale
- Identifizierung der Erkennung: der Sensor erzeugt auf spezifische Weise verwertbare Informationen auf tiefer Ebene (z. B. Punktwolke)
- Identifizierung von Formen/Merkmalen: die sensorspezifische Art und Weise, wie der Sensor verwertbare Informationen auf einer Zwischenebene erzeugt (z. B. Segmentierung)
- Identifizierung von Objekten: Sensorspezifische Methoden zur Erzeugung von Informationen auf Objektebene mit allen erforderlichen Objekteigenschaften.

Nicht alle Teile dieser Dekomposition sind für jeden Sensormodelltyp geeignet. Der Ansatz und die Effekte werden in Towards Serious Perception Sensor Simulation for Safety Validation of Automated Driving - A Collaborative Method to Specify Sensor Models (siehe https://ieeexplore.ieee.org/document/9564661) detaillierter beschrieben.

Die Effekte werden in einer Baumform mit verbindenden Ursache-Wirkungs-Ketten aggregiert. Diese Methode wird PerCollECT genannt: Perception Sensor Collaborative Effect and Cause Tree und ist als Open-Source unter https://github.com/PerCollECT für alle in Frage kommenden Sensortechnologien verfügbar.

Die Genauigkeit, die für die in den verschiedenen Teilen dieser Dekomposition relevanten Effekte erforderlich ist, variiert je nach Modelltyp und Zielverwendung stark

Parameter

Es wurden die typischen für AD relevanten Sensorparameter gesammelt, die für das Testen von AD Systemen und Komponenten notwendig sind. Einige davon sind in den SET Level Sensormodellen implementiert worden. Diese wurden in „Common“ und „Specific“ Parameter unterteilt. Common Parameter sind relevant für alle Sensortechnologien. Specific Parameter sind aus Sicht von Kamera-, Radar- und Lidarsensoren definiert worden.

Common:

HW Sensor related parameters	**Function related parameters**	**Integration in vehicle**
Angular aperture (horizontal and vertical)	max number of simultaneously detectable objects	Mounting position (X,Y,Z)
Field of view	object types detectable	Mounted Sensor orientation Pitch, Roll, Yaw
maximum and minimum Range	object detection and tracking error behaviors	------
Resolution	------	------
Separation power	------	------

Specific:

Camera HW Sensor related parameters	**Radar HW Sensor related parameter**	**Lidar HW Sensor related parameter**
number of pixels (horizontal and vertical)	Antenna diagram (horizontal and vertical)	Beam pattern
channel format (RGB,...)	Sending power	Beam expansion
bit depth	Modulation	Scan pattern (Angular Resolution Elevation, Azimuth)
------	Bandwidth	Receptive area
------	Frequence	Rotation Rate
------	------	Resolution per beam

Cause, Effect, and Phenomenon Relevance Analysis Method

Im Projekt SET Level und in Kooperation mit dem Projekt VVMethoden ist die CEPRA (Cause, Effect, and Phenomenon Relevance Analysis Method) entwickelt worden, um die Auswahl und die Spezifikation von notwendigen Sensoreffekten für eine gegebene Simulationsaufgabe zu unterstützen.

Die komplette Methode ist in der Publikation Towards Serious Perception Sensor Simulation for Safety Validation of Automated Driving - A Collaborative Method to Specify Sensor Models (siehe https://ieeexplore.ieee.org/document/9564661) veröffentlicht worden. Die Ausführung der Methode benötigt Expertenwissen über die Sensordomäne und nutzt die vorher erläuterte funktionale Dekomposition. Zusammengefasst geht diese Methode nach folgenden Schritten vor:

1. Auswahl der funktionalen Dekompositionsebene und in dieser Ebene, Auswahl des Sensoroutputs welcher das Sensormodell zur Verfügung stellen soll
2. Verstehen der Wirkkette, die der Sensor zur Erzeugung dieses Inputs mit Hilfe von PerCollECT durchläuft. Evaluieren der Auftretenswahrscheinlichkeit dieser

Effektketten, in der Betriebsdomäne, in der das System, das den Sensor verwendet, eingesetzt wird.
3. Die Auswirkung dieser Wirkketten auf das System, welches den Sensor nutzen wird, evaluieren.
4. Auswahl der Wirkketten die wirklich relevant für die anvisierte Anwendung des Sensormodells sind (eine Untermenge von 2.)
5. Verstehen der Ursachen, die an diesen Wirkungsketten beteiligt sind

Aus der Durchführung dieser Methode ergeben sich:

- Ein Set von Sensoreffekten, welche im Modell enthalten sein müssen.
- Die Beziehungen zwischen diesen Effekten, welche im Modell enthalten sein müssen.
- ein Verständnis für die Auswirkungen dieser Effekte, die nicht wie beabsichtigt auf die Simulationsaufgabe wirken, wobei eine höhere Auswirkung bedeutet, dass eine Modellierung mit höherer Wiedergabetreue erforderlich ist.
- Anforderungen an die Umweltsimulation (welche Effekte müssen in den Umweltmodellen auch berücksichtigt werden).

Wirkketten für unterschiedlichen Sensoren werden open source in PerCollECT (siehe https://github.com/PerCollECT) gesammelt.

Modellierungsansatz

Im folgenden erfolgt eine Beschreibung der Ansätze, die zur Realisierung der Modelle entwikkelt wurden.

Modellstruktur für Sensormodelle

Eine generische modulare Struktur für objektbasierte Sensormodelle ist erarbeitet worden. Dabei ist die Modellierung in vier Aspekte unterteilt:

1. eine Modellierung der Erfassungsaspekte gemäß einer Sensor-Hardware-Spezifikation.
2. eine Koordinatentransformation vom Sensor zu den Koordinaten des Egofahrzeugs.
3. eine Clusteringfunktion, um erkannte Objekte zu bestätigen und Positions- und Geschwindigkeitsfehler zu berücksichtigen.
4. eine Verfolgungsfunktion, um weitere Fehler bei den Objekteigenschaften zu berücksichtigen und die vom Sensormodell gelieferte Objektliste zu erstellen.

Dieser modulare Aufbau ermöglicht eine bessere Zuordnung von Effekten und Parametern, eine bessere Unterscheidung zwischen Hardware- und funktionsbezogenem Teil des modellierten Sensors und soll ein besseres Variantenhandling unterstützen. Diese Strukturierung kann mit Modifikationen übernommen werden, um noch detailliertere Sensormodelle zu erstellen. Details über diesen Ansatz und seine Anwendung wurden in „Highly Parameterizable and Generic Perception Sensor Model Architecture - A Modular Approach for Simulation-based Safety Validation of Automated Driving“ (siehe https://tuprints.ulb.tu-darmstadt.de/18672/) veröffentlicht.

Parameter und Parametersets

Ein Modellierungsansatz für die Verwaltung von Parametersets mit Nutzung von OSI und FMI Standard ist entwickelt worden. Damit ist es möglich den Modell Code gleich zu halten und via Kalibrierung das Modell zu konfigurieren. Es unterscheidet zwischen:

1. Basic Parameter, bei der Modellinitialisierung über OSI SensorViewConfiguration initialisiert:
    - Das Modell fragt das Simulationstool nach Parameterwerten für eine Sensorkonfiguration, z. B. Field of View, Mounting position …
    - Das Simulationstool antwortet mit der angefragten Konfiguration.
2. Advanced Parameter, vor definiert in auswählbaren Parametersets und in Sensormodell enthalten.
    - Das relevante Profil wird über FMI bei der Modellinitialisierung ausgewählt.
    - Auswahl über Profilname möglich, wenn Blackboxparameter vor dem Anwender unsichtbar bleiben müssen.

Eine Beschreibung dieser Modellierungsansätze ist in der Publikation „Highly Parameterizable and Generic Perception Sensor Model Architecture - A Modular Approach for Simulation-based Safety Validation of Automated Driving“ (siehe https://tuprints.ulb.tu-darmstadt.de/18672/) veröffentlicht worden.

Modellierungsansätze für Kameramodelle

Weitere detaillierte Übersichten für Kameramodelle sind öffentlich verfügbar:

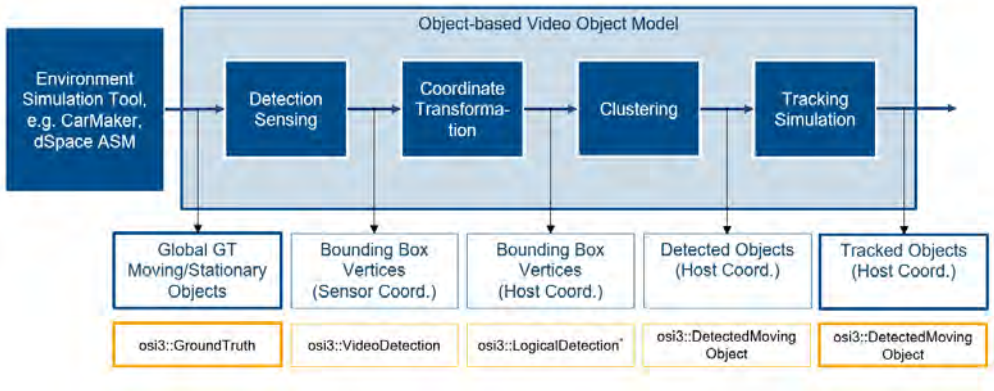


Abbildung 82: Generische innere Struktur eines Objekt-basierte Kameramodells

https://gitlab.setlevel.de/open/models/perception-sensor-models/object_based_camera_object_model/-/blob/main/README.md

https://gitlab.setlevel.de/open/models/perception-sensor-models/image_based_object_detection_model/-/blob/main/README.md

https://gitlab.setlevel.de/open/models/perception-sensor-models/image_based_video_depth_model/-/blob/main/README.md

https://gitlab.setlevel.de/open/models/perception-sensor-models/image_based_video_detected_edges_model/-/blob/main/README.md

Modellierungsansätze für Lidar- und Radarmodelle

Weitere detaillierte Übersichten zu Lidar- und Radarmodellen sind öffentlich verfügbar:

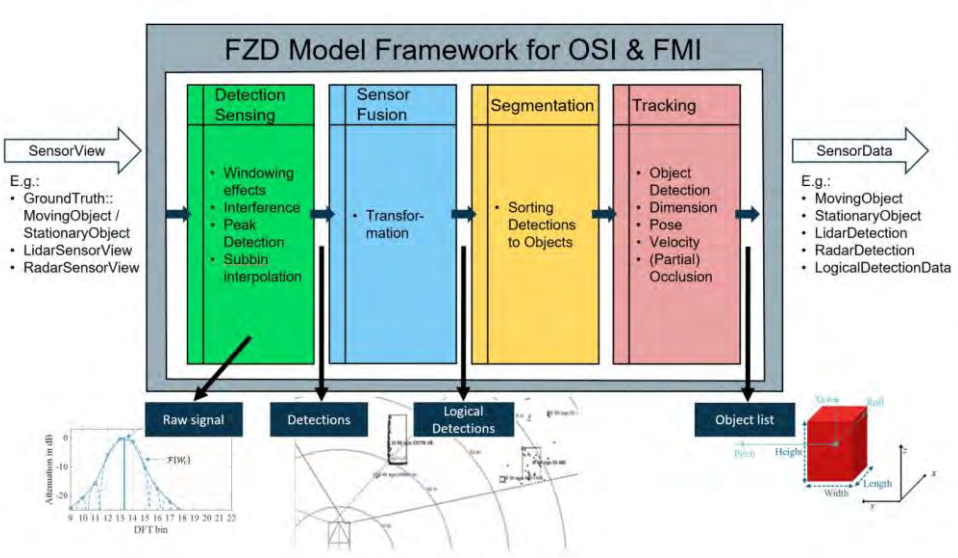


Abbildung 83: Generische innere Struktur eines objektbasierten Radar- oder Lidarmodells

https://gitlab.setlevel.de/open/models/perception-sensor-models/object_based_generic_perception_object_model/-/blob/master/README.md

https://gitlab.setlevel.de/open/models/perception-sensor-models/reflection_based_lidar_object_model/-/blob/master/README.md

https://gitlab.setlevel.de/open/models/perception-sensor-models/reflection_based_radar_object_model/-/blob/main/README.md

Evaluationskriterien und -methoden

Modellverifikation, Funktions-Unit-Tests

Eine Szenarienbasierte Modellverifikation, um die Implementierung von spezifizierten Sensoreffekten zu prüfen ist entwickelt und untersucht worden. Dieser Ansatz ist auf die drei unterschiedlichen Radarmodelltypen angewendet worden: Object based, Reflection based und Geometry based Sensor Models. Es beinhaltet folgende Schritte:

1. Identifizierung der für das Modell spezifizierten Effekte
2. Für diese Effekte eine Definition von minimalistischen Fahrszenarien, die simuliert werden sollen und in denen die Effekte zum Tragen kommen.
3. Für diese Effekte, Metriken und Pass/Fail-Werte definieren.
4. Ausführung den Szenarien und Ableitung der Pass/Fail-Evaluierung.

Mit dieser Methode kann ein Modellprüfer zwar grundlegende Probleme bei der Modellierung der spezifizierten Wirkungen aufdecken, sie hat jedoch ihre Grenzen.

Viele Effekte beeinflussen sich gegenseitig oder treten in Kombination auf, selbst bei einem minimalistischen Szenario, und sie isoliert zu betrachten, ist eine Vereinfachung. Dieser Ansatz kann bis zu einem gewissen Grad dabei helfen, festzustellen, ob ein Modell außerhalb der Spezifikation implementiert wurde, kann aber nicht bewerten, ob die Spezifikation korrekt war, und liefert keine Informationen darüber, wie gut das Modell für einen realen Sensor passend ist.

Sensitivitätsanalyse

Die Auswertung der Ausgabedaten des Sensormodells mit den Ausgabedaten eines realen Sensors gibt Aufschluss über die Genauigkeit des Sensormodells gegenüber dem modellierten Sensor, zeigt aber auch Unterschiede zwischen Sensormodell und realem Sensor auf, die für die zu testende Sensorfunktion, die mit dem Sensormodell getestet wird, nicht immer von gleicher Bedeutung sind. Es wurde ein auf der Sensitivitätsanalyse basierender Ansatz vorgeschlagen, um die Auswirkungen der modellierten Effekte auf die Tests des zu prüfenden Systems weiter zu bewerten. Dieser beinhaltet folgende Schritte:

1. Die für ein Sensormodell relevanten Effekte sind geprüft. Relevante Parameter für diese Effekte und deren Min-Max Wertbereich sind definiert
2. Relevante Samplingszenarien für das Testen des System under Test und Parameter für diese Szenarien sind definiert.
3. Reale und Synthetische Daten für diese Samplingszenarien sind generiert:
    - Relevanten Szenarien werden mit dem realen System gefahren und reale Sensordaten werden gesammelt.
    - Gefahrene Szenarien werden in der Simulation mit Hilfe eines Sensormodells nachsimuliert. Parameterwerte von Schritt 2 werden dabei angewendet und synthetische Daten gesammelt.
    - Reale und synthetische Daten werden durch das System under Test (SuT) verarbeitet und die Ergebnisse anhand identischer Metriken evaluiert.
4. Eine Sensitivitätsanalyse wird durchgeführt, in dem der Einfluss von den definierten Parametern quantifiziert wird.

Dieser Ansatz gibt Aufschluss darüber, welche Modellparameter in welchem Umfang für die in der Simulation beabsichtigten Tests relevant sind, und somit auch darüber, welche Teile des Modells im derzeitigen Zustand möglicherweise nicht mit der beabsichtigten Verwendung des Modells vereinbar sind. Parameter, die als sehr relevant angesehen werden, würden auch eine sorgfältige Modellierung und sorgfältige Validierung erfordern, während weniger relevante Parameter einen einfacheren Ansatz erfordern könnten, um eine vernünftige Laufzeitperformance zu erreichen. Der Ansatz ist mit Einschränkungen verbunden, so hat beispielsweise die Frage, wie gut das angetriebene Szenario in der Simulation wiedergegeben wird, Auswirkungen auf die Ergebnisse dieser Bewertung. Der Ansatz wurde auf ein Radarmodell mit radarsystembezogenen Parametern angewandt, könnte aber auch für andere Modelle verwendet werden und weitere Parameter enthalten, die sich beispielsweise auf ODD oder Umweltsimulation beziehen. Der Ansatz ist in der Publikation “A Sensitivity Analysis Approach for Evaluating a Radar Simulation for Virtual Testing of Autonomous Driving Functions” (siehe https://ieeexplore.ieee.org/document/9162598) veröffentlicht worden.

Weitere SET Level Aktivitäten zur Evaluierung der Qualität synthetischer Daten:

- “A Multi-Layered Approach for Measuring the Simulation-to-Reality Gap of Radar Perception for Autonomous Driving” (https://ieeexplore.ieee.org/document/9564521)
- “Deep Evaluation Metric: Learning to Evaluate Simulated Radar Point Clouds for Virtual Testing of Autonomous Driving” (https://ieeexplore.ieee.org/document/9455235)
- “How to evaluate synthetic radar data? Lessons learned from finding driveable space in virtual environments” (http://tubiblio.ulb.tu-darmstadt.de/120963/)

Lessons learned

- Generische OSMP for sensors (siehe https://gitlab.setlevel.de/open/models/developer_tools/modular-osmp-framework) Modellstruktur ist erfolgreich angewendet worden und in unterschiedlichen Simulationstools integriert.
- Überarbeitung des OSI sensor coordinate systems war notwendig, ist erfolgt und in OSI integriert worden.
- Modellintegration ist mit Hilfe von internen Überwachungen und Datalogging unterstützt worden, folgendes kann untersucht werden:
    - OSI-Schnittstellen, die vom Modell benötigt, aber von der Simulation zur Laufzeit nicht bereitgestellt werden.
    - De-Serialisierung und Serialisierung von OSI-Eingabe- und -Ausgabeschnittstellen für Leistungsbewertungen während der Laufzeit.
    - Funktionsausfälle des Modells oder Arbeiten des Modells in Bereichen außerhalb der Spezifikation während der Laufzeit zur Fehlersuche und Qualitätsbewertung.

2.2.3.3.4 Fahrzeugdynamikmodell und Aktuatormodelle

Erarbeitete Modelle

Für die Modelle der Fahrzeugdynamik und der Aktuatoren wurden zwei Detaillierungsstufen berücksichtigt:

- Detail 1: Fahrzeugdynamikmodell ohne detaillierte interne Architektur,
- Detail 2: Fahrzeugdynamikmodell mit separierten und detaillierten Aktuatormodelle für Bremse-, Powertrain- und Lenkungsteuerung.

Als Fahrzeug sind PKW und LKW mit Anhänger modelliert worden. ISO 11010-1 ist als Referenz für die Modellstruktur und -Klassifizierung genommen worden.

Diese Modelle sind unter https://gitlab.setlevel.de/open/models/vehicle-dynamics-and-actuators-models veröffentlicht worden.

Eine Übersicht der Modellierungsaktivitäten ist unter https://gitlab.setlevel.de/open/models/library-overview/-/blob/main/overviews/b1-7c_poster-vehicle-dynamics-and-actuators-models.pdf veröffentlicht worden.

Eine detaillierte Einführung zu den einzelne Modelle ist unter https://gitlab.setlevel.de/open/models/library-overview/-/blob/main/overviews/b1-7c_slides-vehicle-dynamics-and-actuators-models.pdf veröffentlicht worden.

Herausforderungen

- Aktuatormodelle stammen in der Regel aus verschiedenen Quellen und ihre Integration in ein kohärentes Set, das ein bestimmtes Fahrzeug widerspiegelt, ist eine Herausforderung.
- Diese Modelle sind Gegenstand umfangreicher Varianten, wobei die Modelle entweder bereits für eine Variante parametrisiert oder konfigurierbar sind.
- Je nach Simulationsaufgabe sind große Variationen im Detail der Implementierung erforderlich, um aussagekräftige Ergebnisse zu erzielen.
- Neben der Modellierung des physikalischen Teils der Aktoren ist es eine Herausforderung, den Reglerteil zu integrieren, ohne die Simulationsarchitektur zu sprengen.
- Da mehrere Modelle involviert sind und zusammen ausgeführt werden, wird die Sicherstellung der Korrektheit der dynamischen Zustände und Signale komplex. Numerische Fehler können leicht übersehen werden und zu Oszillationen oder falschen Ergebnissen führen.

Anforderungen und Spezifikationsansätze

Modularität und Skalierbarkeit

Anforderungen für eine generische Modellstruktur sind gesammelt worden. Diese generische Modellstruktur ist für einfache und komplexe Modelle anwendbar mit dem Ziel die Modelschnittstellen gleich zu halten und damit die Simulationsarchitektur für unterschiedlichen Simulationsaufgabe zu beizubehalten.

Diese generische Modellstruktur ist unter https://gitlab.setlevel.de/open/models/library-overview/-/blob/main/overviews/b1-7c_slides-vehicle-dynamics-and-actuators-models.pdf veröffentlicht worden.

Fahrzeugdynamikmodell

Ein 2D kinematisches Single Track Modell, mit idealisierten Aktuatoren und Aktuatorenmanagement ist für Simulationsaufgaben fokussiert auf das Testen und die Analyse von Fahrentscheidungsfunktionen entwickelt worden.

Ein Mehrkörper-Simulationsmodell mit 14 Freiheitsgraden, mit einem kinematisches Dämpfungsmodell, einem Pacejka-Reifenmodell, idealisierten Aktuatoren und Aktuatoren-Management ist für Simulationsaufgaben fokussiert auf das Testen und die Analyse von gesamt AD System mit und ohne Fehlerinjektion entwickelt worden.

Eine Beschreibung dieser Anforderungen ist unter https://gitlab.setlevel.de/open/models/library-overview/-/blob/main/overviews/b1-7c_slides-vehicle-dynamics-and-actuators-models.pdf veröffentlicht worden.

Aktuatormodelle

Für Simulationsaufgaben fokussiert auf das Testen und die Analyse von gesamten AD Systemen mit und ohne Fehlerinjektion sind die Anforderungen für dedizierte Brems-, Powertrain- und Lenkungsmodelle (siehe Abbildung 84) für Physik- und Controllermodelle gesammelt

worden und unter https://gitlab.setlevel.de/open/models/library-overview/-/blob/main/overviews/b1-7c_slides-vehicle-dynamics-and-actuators-models.pdf veröffentlicht worden.

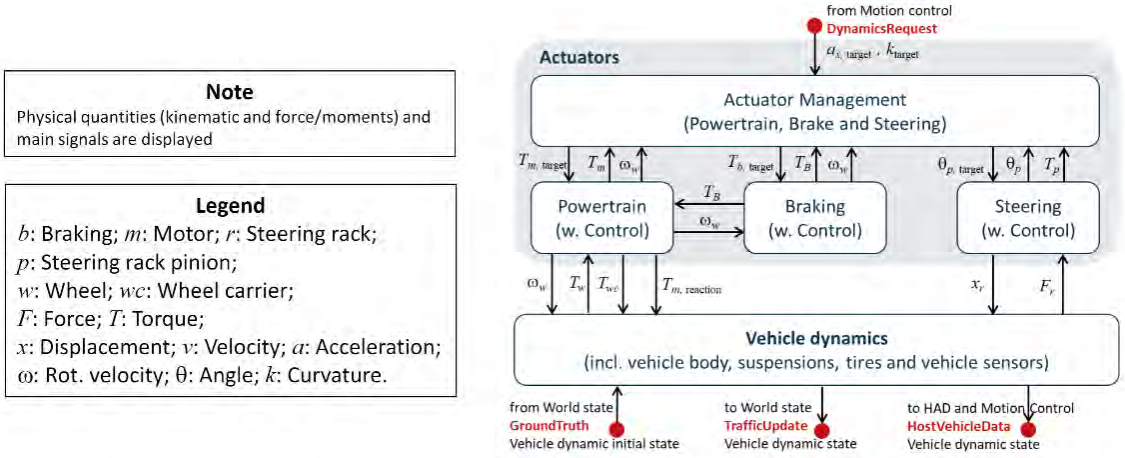


Abbildung 84: Einbindung von Brems-, Powertrain- und Lenkungsmodellen in der Architektur

Modellierungsansatz

Die verfolgten Modellierungsansätze für die einzelnen Modelle wurden öffentlich verfügbar gemacht:

Fahrzeugdynamikmodell:

- https://gitlab.setlevel.de/open/models/vehicle-dynamics-and-actuators-models/vehicle_dynamics_actuators_model_car_detail1/-/blob/main/README.md
- https://gitlab.setlevel.de/open/models/vehicle-dynamics-and-actuators-models/vehicle_dynamics_actuators_model_truck_detail1/-/blob/main/README.md
- https://gitlab.setlevel.de/open/models/vehicle-dynamics-and-actuators-models/vehicle_dynamics_model_car_detail2/-/blob/main/README.md

Lenkungsmodell:

- https://gitlab.setlevel.de/open/models/vehicle-dynamics-and-actuators-models/steering_model_detail1/-/blob/main/README.md
- https://gitlab.setlevel.de/open/models/vehicle-dynamics-and-actuators-models/steering_model_detail2/-/blob/main/README.md

Bremsenmodell:

- https://gitlab.setlevel.de/open/models/vehicle-dynamics-and-actuators-models/braking_model/-/blob/main/README.md

Powertrainmodell:

- https://gitlab.setlevel.de/open/models/vehicle-dynamics-and-actuators-models/powertrain_model_electric/-/blob/main/README.md

- https://gitlab.setlevel.de/open/models/vehicle-dynamics-and-actuators-models/power-train_model_ice/-/blob/main/README.md

Actuator management:

- https://gitlab.setlevel.de/open/models/vehicle-dynamics-and-actuators-models/actuator_management_model/-/blob/main/README.md

Evaluationskriterien und -methoden

Im Rahmen des Projekts SET Level konzentrierte sich die Evaluierungsarbeit auf die Beurteilung, ob ein bestimmtes Modell innerhalb seines Gültigkeitsbereichs verwendet werden kann oder nicht.

Modelle, die außerhalb ihres Gültigkeitsbereichs verwendet werden, können zu ungenauen Ergebnissen führen und es ist schwer zu erkennen, wenn mehrere Modelle zusammen ausgeführt werden. Die Methode fokussiert auf:

- Die Überwachung der Modellzustände eines Modells während seiner Ausführung mit Hilfe so genannter "Gültigkeitsbereichsbeobachter".
- Überprüfung, ob das Modell innerhalb seines spezifizierten Gültigkeitsbereichs bleibt.
- Bereitstellung von Informationen über die gemessene Modellvalidität für die Simulationsumgebung, damit diese mit den Simulationsergebnissen abgeglichen und später verwendet oder die Simulation unterbrochen werden kann.

Ein Beispiel ist unter https://gitlab.setlevel.de/open/models/library-overview/-/blob/main/overviews/b1-7c_slides-vehicle-dynamics-and-actuators-models.pdf erklärt und veröffentlicht worden.

Lessons learned

- Die Bereitstellung des Modells wird komplex, wenn mehrere FMUs zusammen ausgeführt werden, um das Fahrzeugmodell zu erstellen.
- Die Implementierung, das Testen und das Exportieren von Modellen als FMU mit OSI Schnittstellen aus einer Simulink Entwicklungsumgebung hat einen dedizierte OSI wrapper (siehe https://gitlab.setlevel.de/open/models/developer_tools/simulink-osiwrapper) benötigt.
- Modelle unterschiedlicher Partner sind in einem Gesamtfahrzeugmodell mit Hilfe von OSI, FMI and SSP-Standards (siehe https://gitlab.setlevel.de/open/models/library-overview/-/blob/main/overviews/b1-7c_slides-vehicle-dynamics-and-actuators-models.pdf) erfolgreich integriert worden.

2.2.3.3.5 SET Level Publikationen zum Thema Modelle

Modeling and Simulation of Radar Sensor Artifacts for Virtual Testing of Autonomous Driving
Autoren: Martin F. Holder, Clemens Linnhoff, Philipp Rosenberger, Christoph Popp, Hermann Winner
Neunte Tagung Automatisiertes Fahren, München, 21.-22.11.2019
Link: https://mediatum.ub.tum.de/1535116?show_id=1535151&style=full_text

Highly Parameterizable and Generic Perception Sensor Model Architecture - A Modular Approach for Simulation-based Safety Validation of Automated Driving
Autoren: Clemens Linnhoff, Philipp Rosenberger, Martin Friedrich Holder, Nicodemo Cianciaruso, Hermann Winner,
6th International ATZ Conference Automated Driving, Wiesbaden, 13.10.-14.10.2020
Link: https://tuprints.ulb.tu-darmstadt.de/18672/

Sequential lidar sensor system simulation: A modular approach for simulation-based safety validation of automated driving
Autoren: Philipp Rosenberger, Martin Friedrich Holder, Nicodemo Cianciaruso, Philip Aust, Jonas Franz Tamm-Morschel, Clemens Linnhoff, Hermann Winner
Automotive and Engine Technology volume 5, pages 187–197 (2020)
Link: https://link.springer.com/article/10.1007/s41104-020-00066-x

How to evaluate synthetic radar data? Lessons learned from finding driveable space in virtual environments
Autoren: Martin F. Holder, Jan R. Thielmann, Philipp Rosenberger, Clemens Linnhoff, Hermann Winner
13th Workshop Fahrerassistenzsysteme und automatisiertes Fahren, Walting, 03.06.2020
Link: http://tubiblio.ulb.tu-darmstadt.de/120963/

A Sensitivity Analysis Approach for Evaluating a Radar Simulation for Virtual Testing of Autonomous Driving Functions
Autoren: Anthony Ngo, Max Bauer, Michael Resch
Asia-Pacific Conference on Intelligent Robot Systems (ACIRS), (Singapore, Singapore), pp. 122–128, 2020
Link: https://ieeexplore.ieee.org/document/9162598

Deep Evaluation Metric: Learning to Evaluate Simulated Radar Point Clouds for Virtual Testing of Autonomous Driving
Autoren: Anthony Ngo, Max Bauer, Michael Resch
IEEE Radar Conference (RadarConf21), (Atlanta, GA, USA), pp. 1–6, 2021. 07.-14.05.2021
Link: https://ieeexplore.ieee.org/document/9455235

A Multi-Layered Approach for Measuring the Simulation-to-Reality Gap of Radar Perception for Autonomous Driving
Autoren: Anthony Ngo, Max Bauer, Michael Resch
IEEE International Intelligent Transportation Systems Conference (ITSC), (Indianapolis, IN, USA), p. 4008-4014, 2021
Link: https://ieeexplore.ieee.org/document/9564521

Modeling Multi-Driver Interaction in Intersection Scenarios Based on a Hybrid Game Approach
Autoren: Markus Lemmer, Stefan Schwab und Soren Hohmann
CCTA 2021, San Diego, USA, 08.-11.08.2021

Towards Serious Perception Sensor Simulation for Safety Validation of Automated Driving - A Collaborative Method to Specify Sensor Models
Autoren: Clemens Linnhoff, Philipp Rosenberger, Simon Schmidt, Lukas Elster, Rainer Stark,

Hermann Winner
2021 IEEE International Intelligent Transportation Systems Conference (ITSC), Indianapolis, 19.-22.09.2021
Link: https://ieeexplore.ieee.org/document/9564661

Refining Object-Based Lidar Sensor Modeling — Challenging Ray Tracing as the Magic Bullet
Autoren: Clemens Linnhoff, Philipp Rosenberger, Hermann Winner, IEEE Sensors Journal, Volume: 21, Issue: 21, 01.11.2021
Link: https://ieeexplore.ieee.org/document/9548071

Generation of Complex Road Networks Using a Simplified Logical Description for the Validation of Automated Vehicles,
Autoren: Daniel Becker, Fabian Ruß, Christian Geller, Lutz Eckstein
2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), DOI: 10.1109/ITSC45102.2020.9294664
Link: https://ieeexplore.ieee.org/abstract/document/9294664

Agentenmodell für die Closed-Loop-Simulation von Verkehrszenarien,
Autoren: Daniel Becker, Jens Klimke, Lutz Eckstein,
ATZelektronik 05.05.2021, 16. Jahrgang, S.42-46
Link: https://www.springerprofessional.de/agentenmodell-fuer-die-closed-loop-simulation-von-verkehrsszenar/19141908

Vehicle simulation model chain for virtual testing of automated driving functions and systems,
Autoren: R. Bartolozzi, V. Landersheim, G. Stoll, H. Holzmann, R. Möller und H. Atzrodt
2022 33rd IEEE Intelligent Vehicles Symposium (IV), Aachen, Germany June 5-9, 2022, DOI: 10.1109/IV51971.2022.9827074
Link: https://ieeexplore.ieee.org/document/9827074

Fault injection in actuator models for testing of automated driving functions
Autoren: H. Holzmann, V. Landersheim, U. Piram, R. Bartolozzi, G. Stoll und H. Atzrodt
Vehicles 2023, 1
Link: https://doi.org/10.3390/vehicles5010006

System Design of an Agent Model for the Closed-Loop Simulation of Relevant Scenarios in the Development of ADS
Autoren: Jens Klimke (E.Go Moove GmbH), Daniel Becker, Lutz Eckstein (beide ika)
07.10.2020, 29th Aachen Colloquium 2020 Aachen

Agentenmodell für die Closed-Loop Simulation von Verkehrsszenarie
Autoren: Daniel Becker (ika), Jens Klimke (Cariad SE), Lutz Eckstein (ika)
05.2021 ATZelektronik

2.2.3.4 Tools für Modelle

Neben der Open Source Model Library sind einige Tools für die Modellentwicklung und -Integration entwickelt worden:

- OSMP
- Simulink OSI Wrapper

- Streckengenerator
- Standard Checkers

2.2.3.4.1 OSMP

Mit OSMP ist eine OSI und FMI kompatible „Hülle“ zur vereinfachten Modellintegration in Simulationstools erstellt worden. Diese ist für Agentenmodelle und Sensormodelle konkret genutzt worden. Durch OSMP ist es möglich, interne Modellsignale nach OSI und FMI Standard anderen Modellen zur Verfügung zu stellen. Es ist auch möglich, Signale von anderen Modellen nach OSI und FMI Standard einzulesen.
Dediziert für Sensormodelle ist eine Hülle (siehe https://gitlab.setlevel.de/open/models/developer_tools/modular-osmp-framework) auf Basis dieser OSMP entwickelt worden. Es ist damit möglich, auch ein Sensormodell so zu strukturieren, dass Teile des Modells vereinfacht austauschbar sind, ohne dafür die OSI und FMI Kompatibilität zu verlieren.

2.2.3.4.2 Simulink OSI Wrapper

Das Tool Simulink ist für die Entwicklung von Fahrzeugdynamikmodellen und Aktuatormodelle weit verbreitet. Deshalb ist im Projekt SET Level ein Simulink OSI Wrapper (siehe https://gitlab.setlevel.de/open/models/developer_tools/simulink-osiwrapper) erstellt worden. Mit diesem Wrapper ist es möglich, einzelne Modelle aus Simulink als FMU zu exportieren und diese um den OSI Standard zu ergänzen. Damit wird die Integration im Simulationstool weiterhin standardisiert gehalten.

2.2.3.4.3 Streckengenerator

Eine Motivation für ein Umweltmodell, das synthetische Strecken generiert war, dass es zu prüfen gilt, ob auch der Straßenentwurf Einfluss auf die Leistungsfähigkeit einer automatisierten Fahrfunktion hat. Außerdem ist es so möglich, eine Vielzahl verschiedener Szenarien in kürzester Zeit zu generieren und automatisiert durchzuführen.
Im weiteren Verlauf ist der Abschnitt wie folgt strukturiert: Zunächst wird ein neues Konzept für die Beschreibung des Straßenverlaufs eingeführt. Dann wird beschrieben, wie diese logische Streckenbeschreibung in einem Open source Tool umgesetzt ist, um eine Vielzahl von Simulationskarten im ASAM OpenDRIVE Standard zu generieren. Abschließend wird auf die Parametrierung und Anwendung des Modells eingegangen.

Konzept
Das meistgenutzte Format für die Darstellung von Straßennetzwerken in der Simulation ist der Standard ASAM OpenDRIVE, welcher die Möglichkeit bietet Straßenverläufe mit mathematisch beschriebenen Geometrien (bspw. Kreisbögen, Spiralen, Polynomen) auszudrücken. Das hat den Vorteil gegenüber diskreten Punkten, dass eine kontinuierliche Beschreibung vorliegt, die dadurch bei gleicher Datenmenge genauer ist.
Um eine leicht umzusetzende Variation von Straßennetzen umzusetzen, wurde ein Streckenbeschreibungskonzept entwickelt das modular und weniger redundant als OpenDRIVE ist. Die Umsetzung des Eingangsformats wurde in XML umgesetzt und dessen hierarchische Struktur ist in Abbildung 85 dargestellt. Auf oberster Kinderebene werden Segmente und deren modulare Verknüpfung definiert. Segmente können entweder an deren Enden aneinander geflanscht werden, oder es wird automatisch durch eine mathematische Optimierung eine Verbindungsstraße generiert. Segmente können Kreuzungen, Verbindungsstraßen, oder Kreisverkehre sein. Dieses Konzept ermöglicht es den Nutzenden, jedes Segment einzeln

ohne Relation zu anderen Elementen, zu erstellen und anschließend zu einem komplexen Netz zusammenzusetzen (siehe Abbildung 85).

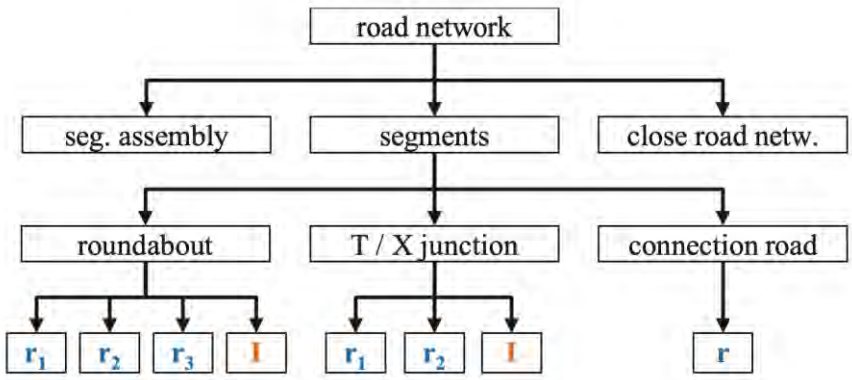


Abbildung 85: Hierarchischer Aufbau des Eingabeformats für die Streckengenerierung

In Abbildung 86 ist exemplarisch zu sehen, wie eine T-Kreuzung auf Referenzlinienebene angelegt wird. Ohne Angabe der Lage im Raum werden die beiden Straßen der Kreuzung durch Linien, Spiralen und Kreisbögen beschrieben (Abbildung Abbildung 86a). Anschließend wird unter Angabe eines Kreuzungswinkels α und einem Punkt entlang der Referenzlinie der Hauptstraße die T-Kreuzung erzeugt (siehe Abbildung 86). In weiteren Schritten werden automatisch Fahrstreifen auf den Zufahrtstraßen generiert, sowie innerhalb der Kreuzung alle nötigen Verbindungsfahrstreifen. Hierfür müssen die ursprünglichen Referenzlinien „aufgeschnitten“ werden. X-Kreuzungen werden analog erzeugt, wobei hier auf beiden kreuzenden Straßen entlang der Referenzlinie Kreuzungspunkte definiert werden.

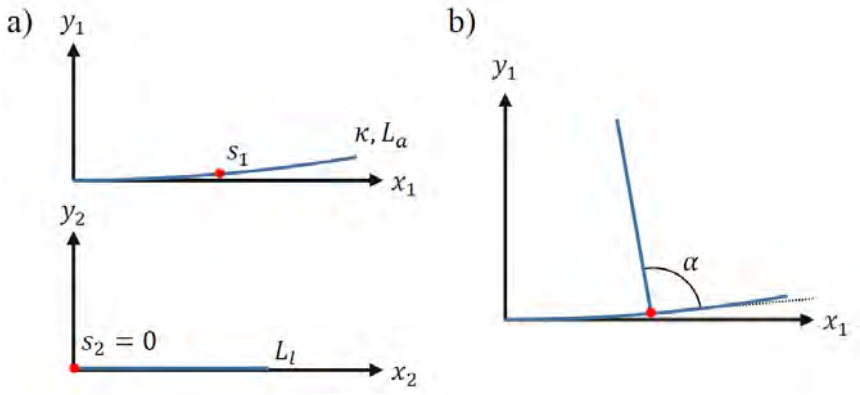


Abbildung 86: Darstellung des Konzepts für die Erstellung einer Kreuzung aus zwei separat definierten Referenzlinien

Damit aus einzelnen Segmenten ein Straßennetz werden kann, gibt es zwei Möglichkeiten zum Verbinden von Segmenten. Zum einen können Elemente direkt aneinandergesetzt werden. Hier gibt es die Randbedingung, dass kein Krümmungssprung auftreten darf. Diese Einschränkung sorgt einerseits dafür, dass stets ein glatter Verlauf entlang der Referenzlinie gegeben ist und vereinfacht das Verketten enorm, weil nur die jeweiligen Enden beider Segmente referenziert werden müssen und so keine Freiheitsgrade mehr vorhanden sind (Vgl. Krümmungsband im Straßenbau). Der Verkettungsprozess ist beispielhaft in Abbildung 87 gezeigt. Zunächst wird ein Referenzelement gewählt (Abbildung 87a) und im globalen Koordinatensystem platziert und ausgerichtet. Im weiteren Verlauf kann der im vorigen Absatz beschriebene Prozess durchgeführt werden (Abbildung 87b) und es entsteht ein Straßennetz.

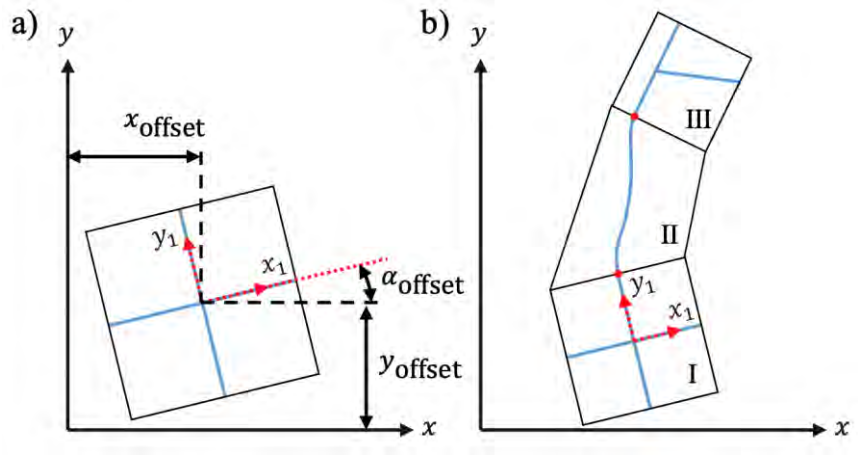


Abbildung 87: Beispielhafte Aneinanderreihung einzelner Segmente

Auf der anderen Seite bietet das Konzept die Möglichkeit, zwischen zwei offenen Enden automatisch eine Verbindungsstraße zu generieren. Dies geschieht mit Hilfe einer numerischen Optimierungsfunktion, die eine Kombination aus Geraden, Spiralen und Kreisbögen errechnet, welche exakt in die definierte Lücke passt.

Implementierung

Das Konzept wurde im Projektverlauf als C++ Library implementiert, um zu zeigen, dass damit standardisierte OpenDRIVE Karten erzeugt werden können, mit denen man Simulationen zur Absicherung von automatisierten Fahrzeugen durchführen kann. Der Code ist auf GitHub öffentlich verfügbar (siehe https://github.com/ika-rwth-aachen/RoadGeneration) und stellt eine prototypische Implementierung dar.
Im Wesentlichen werden die im vorangehenden Abschnitt beschriebenen Schritte zur Überführung der vereinfachten Annahmen des Konzepts in eine OpenDRIVE Karte iterativ abgearbeitet. Wenn die Nutzenden keine spezifischen Angaben zu Fahrstreifenbreiten, oder Kreuzungsabmessungen im Eingangsformat machen, werden Standardwerte gewählt, die den Vorgaben des Straßenbaus entsprechen. So ist es einerseits möglich, ausgewählte Segmente sehr genau zu spezifizieren und andererseits möglich, mit wenig Aufwand Segmente zu spezifizieren, die dann automatisch mit Werten befüllt werden. Innerhalb des Kreuzungsbereichs müssen neue Verbindungsstraßen generiert werden, wofür die ursprünglichen Referenzlinien aufgeschnitten werden. Hierbei wird entweder ein standardisierter Kreuzungsbereich gewählt oder der im Eingangsformat optional definierte Wert für die Aufspanngröße der Kreuzung. Außerdem können extra Abbiegefahrstreifen automatisch erzeugt werden.
In weiteren Schritten werden in der Implementierung die in Abbildung 87 gezeigten Verkettungen realisiert. Eine vollständige Dokumentation der C++ Bibliothek ist auf GitHub zu finden. Abbildung 88 zeigt ausgewählte Beispiele für generierte Kreuzungen.

Variations-Erweiterung

Die ursprüngliche Motivation für das entwickelte Streckenbeschreibungskonzept war die Möglichkeit der einfachen Variation von Straßennetzen (siehe Abbildung 88). Daher wurde in einem weiteren Schritt ein Pythonmodul entwickelt, welches eben dies ermöglicht. Dazu wurde die Eingangssprache um einen „<vars>“ Tag auf oberster Kind-Ebene erweitert, der es ermöglicht beliebig viele Variablen zu definieren, die dann im weiteren Verlauf der XML Datei eingesetzt werden können. Folgende Variablentypen können aktuell verwendet werden:

- Normalverteilung unter Angabe von Mittelwert und Standardabweichung
- Gleichverteilung unter Angabe von Minimal- und Maximalwert

- Lineare Zusammenhänge in Abhängigkeit anderer Variablen, z. B.: c=40*a+b

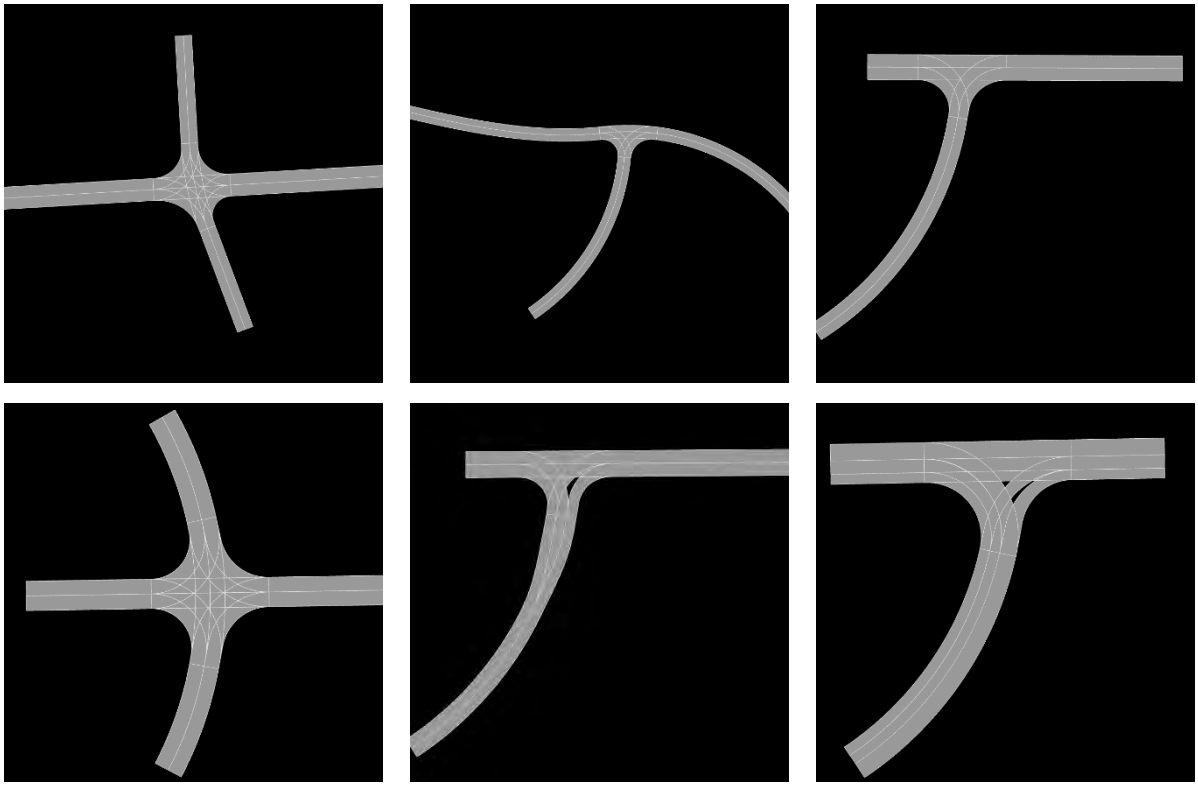


Abbildung 88: Beispiele für generierte Kreuzungen

Bei der Umsetzung wird zunächst die Eingangsdatei eingelesen, um die Variablen zu erzeugen und entsprechend Ihres Typs zu befüllen. Hierbei werden so viele Samples gebildet, wie es Variationen des Straßennetzes geben soll. Anschließend werden temporäre Eingangsdateien mit den eingesetzten konkreten Variablensamples, für die im vorherigen Abschnitt beschriebene C++ Bibliothek generiert. Letztere wird in Python eingebunden und so oft aufgerufen, bis die gewünschte Anzahl von Straßennetzen vorliegt.
Auch das Variationstool ist Teil des oben erwähnten GitHub Repository.

Parametrierung
Nach der Implementierung des Konzepts inklusive der Möglichkeit zur Variation stellte sich die Frage, welche sinnvollen Parameter und Parameterverteilungen genutzt werden können, um eine automatisierte Fahrfunktion zu testen. Hierfür eignen sich zum einen Vorgaben aus den Richtlinien für die Anlage von Straßen, aus denen eine Vielzahl von zu verbauenden Geometrieparametern extrahiert werden können. Auf der anderen Seite entstand die Idee, bestehende Straßennetze systematisch zu analysieren und dadurch reale Parameterverteilungen zu erhalten, die dann für synthetische Kartenerzeugung genutzt werden können. Abbildung 89 zeigt schematisch die konzipierte Verarbeitungskette, welche im folgenden Abschnitt kompakt erläutert wird.

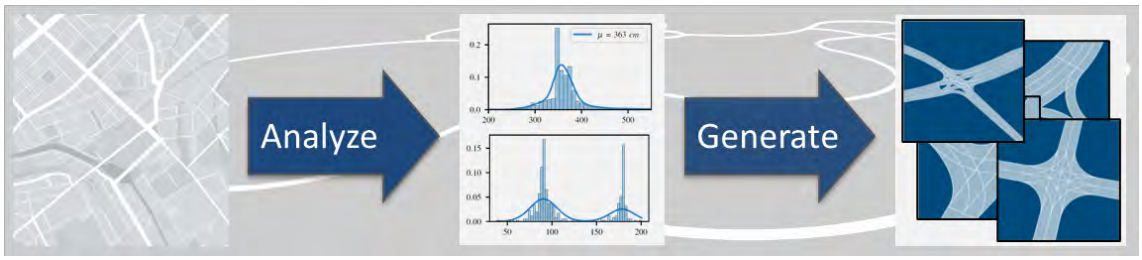


Abbildung 89: Darstellung der Datenverarbeitung für die Streckenerstellung

Bei der Umsetzung der Analyse werden hochgenaue Karten von HERE[7] benutzt, aus welchen die Geometrien von realen Kreuzungen extrahiert werden können. HERE-Karten sind in mehrere sogenannte Kacheln räumlich unterteilt und außerdem in einer Ebenenstruktur abgelegt. Die Datenverarbeitung ist in zwei separate Module unterteilt: Kreuzungsextraktion und Kreuzungsanalyse. Das erste Modul erstellt eine Liste aller Kreuzungen innerhalb bestimmter Kacheln. Darüber hinaus werden weitere Parameter wie der Abstand zwischen Kreuzungen berechnet. Das zweite Modul führt dann mehrere verarbeitete Kacheln zusammen, kategorisiert die Kreuzungen und generiert neue Parameter, um die Kreuzungen genauer zu beschreiben, z. B. die maximale Krümmung der Abbiegespuren. Der Einfachheit halber werden in dieser Arbeit nur Kreuzungen mit drei und vier Armen betrachtet. Das heißt, Autobahnausfahrten oder Kreisverkehre werden in diesem Stadium nicht betrachtet. Folglich werden die übrigen Kreuzungen in vier Typen klassifiziert (T-, Y-, X- und unsymmetrische X-Kreuzungen). Ziel ist es, festzustellen, ob zwei gegenüberliegende Arme einer Kreuzung geometrisch eine durchgehende Straße bilden, was durch einen Winkelbereich definiert wird, der auf 150 bis 210 Grad zwischen zwei Armen festgelegt wird.
Dann folgt der Prozess der Generalisierung. Ziel ist es, die Verbindungen so anzuordnen, dass sie für alle Kreuzungen eines bestimmten Typs immer identisch sind. Bei einer T-Kreuzung beispielsweise erhält der Arm den Index 1 und die Glieder, die die Hauptstraße bilden, werden mit dem Index 0 beziehungsweise 2 bezeichnet. Auf diese Weise ist die statistische Analyse reproduzierbar. Nachdem die Topologie der Kreuzungen extrahiert wurde, werden die Fahrspuren der Verbindungen und die Fahrspuren innerhalb der Kreuzung analysiert. Neben Parametern wie Fahrspurbreite und Anzahl der Fahrspuren werden auch die maximale und mittlere Krümmung der Abbiegespuren berechnet, da dieses Maß in HERE Karten nicht gespeichert ist. Die Krümmung wird mit Hilfe von diskreten Ableitungen für die Mittellinie jeder Fahrspur berechnet werden, die als Polylinie gespeichert ist.
Abbildung 90 zeigt beispielhaft den Parameter „Kreuzungswinkel“ für T- und X-Kreuzungen. Es ist gut zu sehen, dass bei T-Kreuzungen offenbar Normalverteilungen um 90 und 180 Grad vorliegen, was den Erwartungen entspricht. Die Ergebnisse für X-Kreuzungen sehen ebenfalls plausibel aus mit einem Mittelwert von ca. 90 Grad.

[7] https://www.here.com/platform/HD-live-map

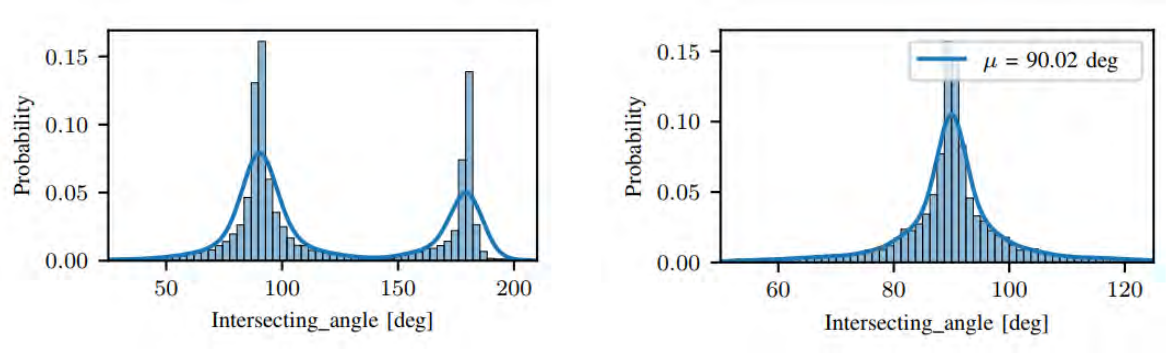


Abbildung 90: Kreuzungswinkel für T- (links) und X-Kreuzungen (rechts) in Berlin

Die Methodik ist beispielhaft in Berlin auf 49 Kacheln mit einer Fläche von 183,75 km^2 angewendet worden. Daraus ergeben sich 1379 X-Kreuzungen und 5131 T-Kreuzungen, die extrahiert werden konnten. Die restlichen beiden Typen treten nur vereinzelt auf und sind für die folgende Auswertung vernachlässigt worden.

Anwendung

Die erarbeitete Werkzeugkette und Parametrisierung wurde im Rahmen von SET Level prototypisch erprobt, um zu zeigen, welchen Einfluss der Straßenentwurf auf die Leistung einer automatisierten Fahrfunktion hat.

In Abbildung 91 sind die Verarbeitungs- und Simulationsschritte einer vom ika entwickelten prototypischen Testumgebung zu sehen. Nachdem ein Satz von logischen Streckentemplates erzeugt wurde, wurden diese variiert und in die für die Simulationskomponenten nötigen Kartenformate umgewandelt. In diesem Fall erwartet die prototypische Fahrfunktion des ika eine Lanelet2-Karte und die Simulationsumgebung CarMaker benötigt das Format „rd5“ wofür IPG ein Konvertierungstool zur Verfügung stellt. Mit diesen nun vorliegenden Eingangsdaten können CarMaker und die ROS-basierte ika-Plattform für die AD Funktion initialisiert werden. CarMaker dient hier als Quelle für Umgebungsfahrzeuge und das Bereitstellen der Fahrphysik. Die ika Fahrfunktion verfügt über eine Bewegungsplanung und übergibt diese an das CarMaker Fahrzeugmodell. Abschließend werden die durchgeführten Simulationen hinsichtlich acht festgelegter Key-Perfomance-Indikatoren ausgewertet und die Fahrfunktion bewertet.

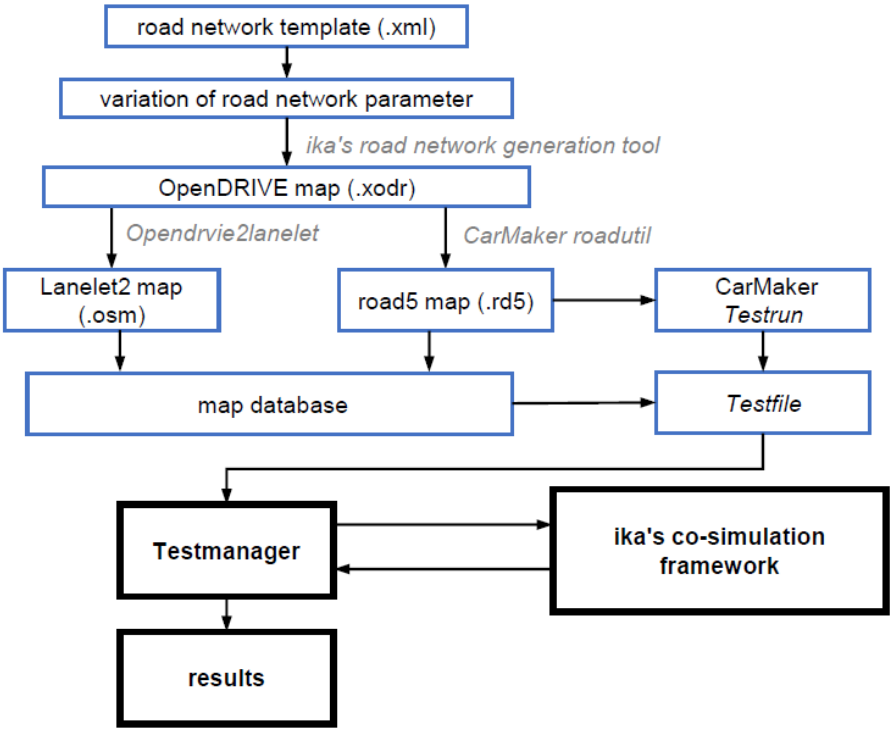


Abbildung 91: Schematischer Aufbau des Testframeworks für die Variation von Straßennetzen in der Simulation

Ziel war es, vorrangig eine durchgängige Werkzeugkette aufzubauen, die grundsätzlich in der Lage ist eine Vielzahl von Simulationen durchzuführen und diese zu bewerten. Die Auswertungen des oben beschriebenen Aufbaus haben gezeigt, dass bestimmte Konfigurationen von Parametern zu schlechteren oder besseren Ergebnissen führen.

2.3 Teilprojekt 4: Instanziierung von Werkzeugketten für Entwicklung und Testen

2.3.1 Übersicht über Inhalte und Ziele von TP 4

Für TP 4 stand die vollständige Instanziierung einer simulationsbasierten Werkzeugkette für den Einsatz der Methoden und Werkzeuge des Projektes für das Entwickeln und Testen von höheren Automatisierungsleveln in urbanen Räumen im Mittelpunkt. Dabei wurde von einem Use-Case "komplexe Kreuzung" ausgegangen.
Es ergaben sich in diesem Teilprojekt (TP) vier Arbeitspakete (AP):

AP 4.1 Ableitung der Schnittstellenanforderungen
Ableitung der Schnittstellenanforderungen (Input aus TP 1 bis TP 3 und Output an TP 5) sowie Unterstützung dieser TPs durch IT-Tools. Die Entwicklung eines Traceability-Tools zum Aufbau eines Demonstrators basierend auf den Anforderungen des SET Level Projektes.

AP 4.2 Aufbau einer exemplarischen Simulationsplattform auf Gesamtfahrzeugebene
Aufbau einer exemplarischen Simulationsplattform auf Gesamtfahrzeugebene zur Simulation logischer Szenarien (z. B. auf der Basis von openPASS): entwicklungsbegleitende Simulation zur Unterstützung der Kritikalitätsanalyse.

AP 4.3 Aufbau einer exemplarischen Simulationsplattform auf Subsystemebene der automatisierten Fahrfunktion
Aufbau einer exemplarischen Simulationsplattform für höhere Automatisierungslevel zum Testen kritischer Szenarien (z. B. auf der Basis einer Co-Simulationsplattform) u. a. zur Unterstützung von virtuellen Testmethoden für die Homologation/Zulassung automatisierter Fahrfunktionen.

AP 4.4 Aufbau einer exemplarischen Simulationsplattform auf Komponentenebene
Aufbau einer exemplarischen Simulationsplattform auf Komponentenebene (z. B. Sensoren und Fahrermodelle) u. a. zur Unterstützung der virtuellen Testmethoden auf Komponentenebene für die Homologation/Zulassung automatisierter Fahrfunktionen.

Aufbau Ergebnisse zum finalen Meilenstein
Es sollten für TP 4 bis zum Projektabschluss iterativ über diverse Meilensteine hinweg die folgenden Ziele durch schrittweise Komplexitätssteigerung erreicht werden:

- Anforderungen der Industrieunternehmen an die Simulationsinfrastuktur sollten nachgeschärft werden
- Anforderungen anderer öffentlich geförderter Projekte sollten nachgeschärft, bewertet und priorisiert werden
- Eine verbesserte Version der Simulationsinfrastruktur unter Berücksichtigung priorisierter Anforderungen sollte vorliegen
- Updates der Referenztools beim FZI und DLR sollten erfolgen
- Priorisierte Simulationstasks und Szenarien sollten beschrieben und in der Simulationsinfrastruktur abgebildet werden
- Konsequenzen der Integrationsarchitekturen auf die Referenztools sollten abgestimmt und priorisiert werden
- Weitere Modelle bzw. Updates bereits vorhandener Modelle sollten definiert werden und in der Simulationsinfrastuktur verfügbar sein
- Simulationen sollten ausgeführt werden
- Ergebnisse sollten an die Anforderer und an die interessierte Öffentlichkeit kommuniziert und Lessons learned beschrieben werden.

Anforderungen der Industrieunternehmen an die Simulationsinfrastruktur wurden nachgeschärft: Zu diesem Punkt wurde im Projekt SET Level von der **Simulationsplattform** gesprochen. Eine Referenzarchitektur für Simulationsplattformen liegt vor und ist für die Simulation Use Cases (SUCs) unterschiedlich ausgeprägt. Für den finalen Meilenstein ist es gelungen,

eine stufenweise Verfeinerung übergreifender Konzepte, die sich am ASAM SimGuide orientierten, auf konkrete SUC-Architekturen auszuarbeiten (siehe Abbildung 92: Verfeinerungsstufen der Architektur). Eine übergreifende Liste von Anforderungen an die Simulation liegt in AP 1.1 vor. Diese schließen Anforderungen an die Simulationsplattform respektive Simulationsinfrastruktur mit ein. Darüber hinaus wurden im Hinblick auf den finalen Meilenstein Industrieanforderungen an die Simulation in AP 2.3 gesammelt und gesamthaft beschrieben. Der Credible Simulation Process (CSP) aus AP 3.1 wurde zur Dokumentation der Anforderungen an das Tooling auch für den finalen Meilenstein in den einzelnen SUCs genutzt. In AP 5.2 liefen Unterstützungsmaßnahmen für die Implementierung der SET Level Simulation Use Cases in verschiedenen Unternehmen. Hier wurde der methodische Ansatz insbesondere durch den Simulation Request bzw. synonym dazu durch die Simulation Order vervollständigt.

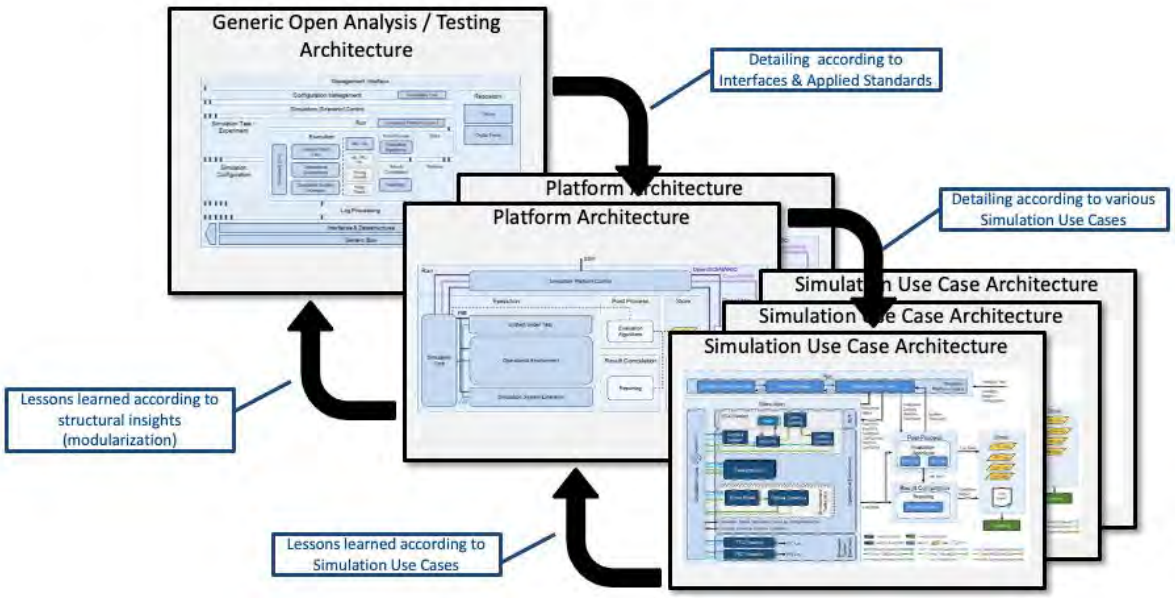


Abbildung 92: Verfeinerungsstufen der Architektur

Anforderungen anderer öffentlich geförderter Projekte wurden nachgeschärft, bewertet und priorisiert: Anforderungen aus **VVMethoden** wurden in der Liste von Anforderungen aus AP 1.1 berücksichtigt. Anforderungen aus VVMethoden, auf Open Source Modelle aus SET Level zuzugreifen, wurden durch die Vergabe entsprechender Lizenzen erfüllt. Anforderungen aus der **KI-Projektfamilie**, auf die Verkehrsräume (Traffic Spaces) von AP 1.2 zugreifen zu können wurden erfüllt. Für beide Aspekte wurden die Ablagestrukturen im SET Level Repository finalisiert, es wurden die Metadaten und die Lizenzen für Referenzszenarien, Traffic Spaces und die Modelle erstellt und die Freigabe als auch der Zugriff der SET Level Artefakte über die SET Level Website ist erfolgt.
Eine verbesserte Version der Simulationsinfrastruktur unter Berücksichtigung priorisierter Anforderungen liegt vor (Endversion): openPASS ist bei Projektpartnern installiert. Die Toolketten von IPG und dSPACE berücksichtigen Anforderungen aus SET Level. TRACY liegt in einer Version vor, in der das Thema Traceability gezeigt wurde. Eine Szenario Engine ist als separate Tool-Komponente in verschiedenen Tools ein sinnvoller modularer Tool-Baustein. Updates der Referenztools bei OFFIS, FZI und DLR sind erfolgt: Unter Referenztools werden im Projekt SET Level die Installationen der Simulation Use Cases in den unterschiedlichen Tools verstanden. OFFIS wurde im Januar 2022 zu einem DLR-Institut und setzt das in Simulation Use Case 1 weiter entwickelte Tooling auf der Basis von openPASS ein. OFFIS hat mit

einer Scenario Engine einen weiteren Toolbaustein beigesteuert. Vom FZI wurde die in den Simulation Use Cases verwendete AD-Funktion (HADf) bereitgestellt, die in den Simulationstools durch die Entwicklung eines OSI-Gateways integriert wurde. Außerdem stellte das FZI einen Renderer für das Radarmodell bereit. Vom DLR wurde eine Forschungsimplementierung umgesetzt. Der zuerst erstellte Simulationskern mit der zentralen Komponente CARLA als Open Source Simulations-Werkzeug kam sowohl im Rahmen dieses Projektes bei der Umsetzung des SUC 2-MS2 Szenarios als auch im Rahmen des Projektes VVMethoden erstmals zum Einsatz. Die aktuelle Implementierung realisiert einen vollständigen technischen Durchstich von der webbasierten GUI aus MS1 über Plattformsteuerungsmodule bis hin zum Simulationskern.
Priorisierte Simulationstasks und Szenarien sind beschrieben und in der Simulationsinfrastruktur abgebildet: dies wurde in den Simulation Use Cases (SUCs) durch den Aufbau beispielhafter Simulationsplattformen ausgeprägt und ist in separaten Kapiteln des gemeinsamen Schlussberichtes separat und ausführlich beschrieben.
Konsequenzen der Integrationsarchitekturen auf die Referenztools wurden abgestimmt und priorisiert: die weitere Arbeit zum Thema Architektur hat über die Spezifikation von Verfeinerungsstufen hinweg (siehe Abbildung 92) gezeigt, wie systematisch und durch einen Top-down Ansatz die Durchgängigkeit abstrakter Anforderungen zur konkreten Ausprägung von Architekturen gestaltet worden ist. Eine Reihe von Veröffentlichungen belegt diesen von SET Level verfolgten Ansatz und hat zu substantiellen Erkenntnisgewinnen im Hinblick auf Simulationsarchitekturen beigetragen. Die Liste der Veröffentlichungen ist am Ende dieses Berichtes zu finden und ist auf der SET Level Website einsehbar.
Weitere Modelle bzw. Updates bereits vorhandener Modelle sind definiert und in der Simulationsinfrastruktur verfügbar: eine Übersicht über die in den verschiedenen Simulation Use Cases genutzten Modelle wurde in den Vorträgen zu den Simulation Use Cases auf dem Abschluss-Event vorgestellt. Diese sind mit entsprechenden Lizenzen versehen auf der SET Level Website veröffentlicht.
Simulationen werden ausgeführt, Ergebnisse wurden an die Anforderer und an die interessierte Öffentlichkeit kommuniziert und Lessons learned sind beschrieben: dies wurde in den Simulation Use Cases implementiert und über ausführliche Berichte in den Quartalsmeetings an die Anforderer kommuniziert. Dabei wurden jeweils auch Lessons learned berichtet. Ausführlich ist dies im gemeinsamen Schlussbericht an anderer Stelle beschrieben. Die interessierte Öffentlichkeit konnte sich durch die Teilnahme am Halbzeitevent am 29.4.2021 und am Abschluss-Event am 11./12.10.2022 über die dauerhaft verfügbaren Vorträge und Folien informieren. Modelle und Veröffentlichungen sind über die SET Level Website verfügbar (siehe https://setlevel.de).

2.3.2 Ergebnisbeiträge von TP 4

Die Ergebnisse von **AP 4.1 Ableitung der Schnittstellenanforderungen** finden sich in folgenden Ergebnisbeiträgen
- Prozesse und Kollaboration
- Simulationsziele
- Architektur
- Metadaten
- Tool-Validierung und Performance
- Traceablity und Traceability-Tool „TRACY“

Die Ergebnisse von **AP 4.2 Aufbau einer exemplarischen Simulationsplattform auf Gesamtfahrzeugebene** finden sich in folgenden Ergebnisbeiträgen

- Closed-loop Verkehrssimulation für Analyseaufgaben
- Befähigung des Werkzeugs openPASS
- Aufbau und Integration Simulationsplattform
- Durchführung und Auswertung Kritikalitätsanalyse
- Begleitende Aktivitäten

Dabei sind vielfältige Zuarbeiten anderer Arbeitspakete mit eingeflossen, insbesondere von AP 1.2 *Referenzszenarien und Verkehrsräume* und von AP 3.1 *Open Source Modelle (z. B. Agentenmodelle)*. Außerdem wurde *durch AP 4.2* auch die Evaluierung von *AP 3.1 Credible Simulation Process* ermöglicht.
Die in AP 4.2 geplanten Aufwände für eine Referenzimplementierung werden unter den detaillierten Ergebnissen aus TP 4 als Forschungsimplementierung unter der Überschrift *Werkzeug Forschungsimplementierung* beschrieben.

Die Ergebnisse von **AP 4.3 Aufbau einer exemplarischen Simulationsplattform auf Subsystemebene der automatisierten Fahrfunktion** finden sich in folgenden Ergebnisbeiträgen:
- Closed-loop Simulation für Validierung und Integrationstests
- Werkzeug Forschungsimplementierung
- Umsetzung in den Werkzeugen von dSPACE
- Umsetzung in den Werkzeugen von IPG
- Bewertung der Ergebnisse

Dabei sind vielfältige Zuarbeiten anderer Arbeitspakete mit eingeflossen, insbesondere von AP 1.2 *Referenzszenarien und Verkehrsräume* und von AP 3.1 *Open Source Modelle.* mit eingeflossen, Außerdem wurde *durch AP 4.3* auch die Evaluierung von *AP 3.1 Credible Simulation Process* ermöglicht.

Die Ergebnisse von **AP 4.4 Aufbau einer exemplarischen Simulationsplattform auf Komponentenebene** finden sich in folgenden Ergebnisbeiträgen
- Validierung einer isolierten Komponente ohne Wechselwirkung
- Allgemeines zur Umsetzung der Kamera-Simulation
- Umsetzung Kamera-Simulation in diversen Werkzeugen
- Allgemeines zur Umsetzung der LIDAR-Simulation
- Umsetzung LIDAR-Simulation in diversen Werkzeugen
- Umsetzung Radar-Simulation in Werkzeugen von ETAS und Open Source
- Bewertung der Ergebnisse

Auch hier sind vielfältige Zuarbeiten anderer Arbeitspakete mit eingeflossen, insbesondere von AP 1.2 *Referenzszenarien und Verkehrsräume* und von AP 3.1 *Open Source Modelle*.

2.3.3 Detaillierte Ergebnisbeiträge

2.3.3.1 Ableitung der Schnittstellenanforderungen

2.3.3.1.1 Prozesse und Kollaboration

Prozesse und Workflows für die Kollaboration

Modelle von Sensoren, Aktoren und insbesondere der Fahrfunktion sind im Kontext von SET Level Verhaltensmodelle und als solche in Software implementiert. Auch wenn diese Softwaremodule als FMU umgesetzt werden, ihre Schnittstellen dem OSI-Standard folgen und der Zusammenbau mittels SSP definiert wird, gibt es doch immer wieder den Bedarf, diese Schnittstellen zu ändern. Formale Änderungen der Syntax (wie viele Parameter

welchen Typs hat ein bestimmter Aufruf?) sind dabei noch harmlos in dem Sinn, dass sich durch geeignete Werkzeuge überprüfen lässt, ob die Änderung an wirklich allen relevanten Stellen berücksichtigt wurde. Viel diffiziler sind Änderungen an der Semantik, also der Bedeutung von Parametern oder Methoden im Kontext der Simulation.
So wurde bereits sehr früh im Projektverlauf deutlich, dass ein definierter und gesteuerter Prozess zur Änderung von Schnittstellen für die erfolgreiche Zusammenarbeit im Projekt unverzichtbar ist.
Deshalb entwickelten AP 2.1 und AP 4.1 gemeinsam einen *Schnittstellenänderungsprozess*, der seitens der Infrastruktur ein Issue-Management-System (hier: GitLab) und ein Code-Versionierungssystem (hier: Git) nutzt. In betreffenden Repositories wurden Change Control Boards (CCBs) etabliert, um Priorisierung, Implementations- und Integrationsplanung der Änderungen abzustimmen. Insbesondere die extern wirksamen Beiträge zur Weiterentwicklung von OSI/OSMP erforderten dabei ein besonders sorgfältig abgestimmtes Vorgehen.
Der Schnittstellen-Änderungsprozess zur nachhaltigen Sicherung der Passung zwischen Modellkomponenten wurde dabei anhand eines Beispiels gemeinsam entwickelt und wie folgt definiert:

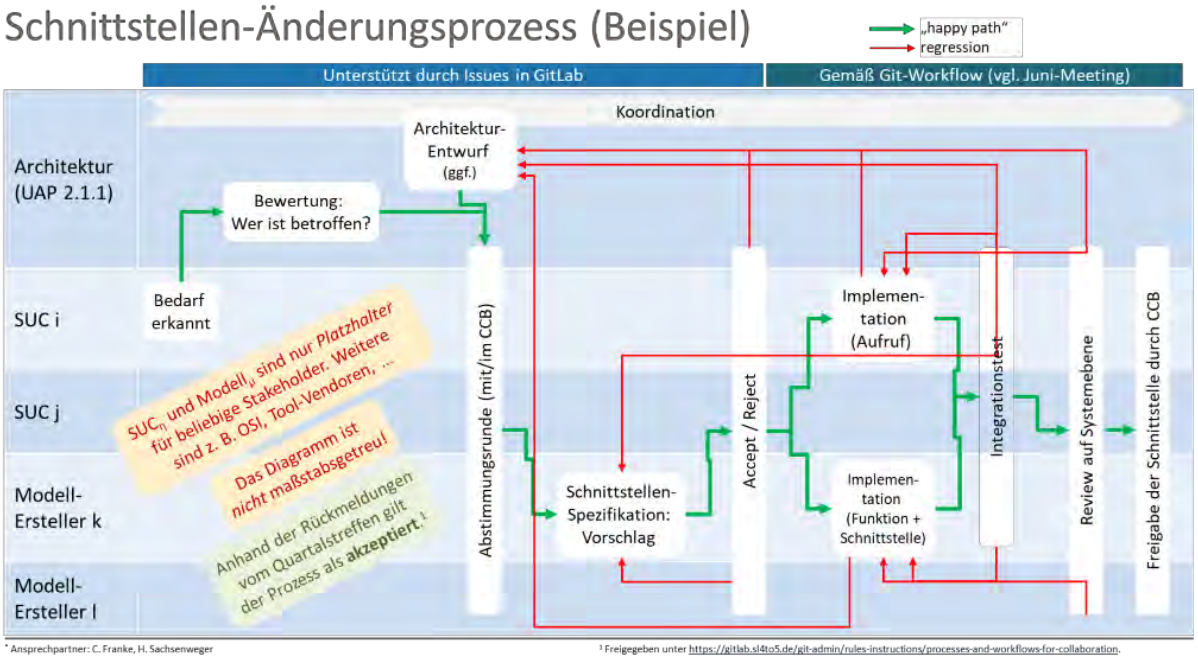


Abbildung 93: Schnittstellen-Änderungsprozess

Spezifikation und Ramp-Up der projektweiten GitLab-Installation

Ursprünglich war davon ausgegangen worden, dass ein Repository für Szenarien benötigt wird. Sehr schnell reifte jedoch die Erkenntnis, dass eine leistungsfähige Versionsverwaltung für *jedwede Art* von Artefakten benötigt wird, die im Projekt entstehen. Selbst das Handling interner Dokumente zum Projektmanagement würde davon profitieren. Bei vielen Projektpartnern lagen bereits Erfahrungen mit der freien Software „Git“ (siehe https://de.wikipedia.org/wiki/Git) vor.
Im Rahmen von AP 4.1 wurde daher über eine Anforderungsermittlung und eine Bewertung diverser technischer Optionen eine Entscheidung herbeigeführt, Git in der Inkarnation „GitLab“ (siehe https://de.wikipedia.org/wiki/GitLab) zu verwenden. Neben der reinen Versionsverwaltung standen dem Projekt damit auch mächtige Funktionalitäten zur Verwaltung hierarchisch gruppierter Repositories, zur Koordination der Zusammenarbeit via Issue-Tracking und für individuelle Benutzerberechtigungen zur Verfügung.

Für den GitLab-Einsatz wurde je ein Handbuch für „einfache“ Benutzer und für Repository-Verantwortliche mit spezifischem Regelwerk erstellt. Zusätzlich entstanden Anleitungen für Anwender mit Beispielen, wie mit bestimmten Artefakten umgegangen werden soll.
Im weiteren Verlauf des Projekts wurden die Projektteilnehmer kontinuierlich bei der Verwendung von GitLab unterstützt. Es wurden „Lessons Learned“ zum GitLab durchgeführt, aus denen deutlich erweiterte Versionen der Handbücher hervorgingen.

Manual zum Aufbau von und Umgang mit Repositories für Traffic Spaces und Scenarios

Im Rahmen von SET Level musste zunächst geklärt werden, wo genau Referenzszenarien gespeichert werden, ob das Mehrebenen-Konzept als Datenbanklösung nur die tiefste Ebene konkreter Szenarien oder auch höhere Ebenen funktionaler Szenarien mit Parameterräumen oder verbale Beschreibungen von Szenarien oder gar die Rohdaten aus konkreten Versuchsfahrten erfordert, ob diesbezüglich auf die PEGASUS-Datenbank aufgesetzt werden kann, ob ein eigenes Repository geschaffen werden muss, oder ob durch Referenzierung auf Szenarien-Repositories der IT-Vendoren in ausreichendem Maße Beispiele durchgespielt werden können, die vor dem Hintergrund der Brauchbarkeit der Schnittstellenformate, von Austauschbarkeit und Wiederverwendung den Anforderungen des Testkonzeptes genügen.
Nach einer Analyse der relevanten Anforderungen wurde beschlossen, eine eigene Sammlung von Traffic Spaces und Scenarios in GitLab aufzubauen.
Gemeinsam mit TP 1 hat AP 4.1 daher ein Handbuch entwickelt, das beschreibt, wie Traffic Spaces und die darauf aufbauenden Scenarios in Gruppen und Repositories des GitLab abzulegen sind (sog. „Ablagestruktur“), welches Benennungsschema anzuwenden ist und wie die unterschiedlichen Bedürfnisse der drei SUCs befriedigt werden können.
Die folgenden Abbildungen verdeutlichen einige Regeln dieser Ablagestruktur:

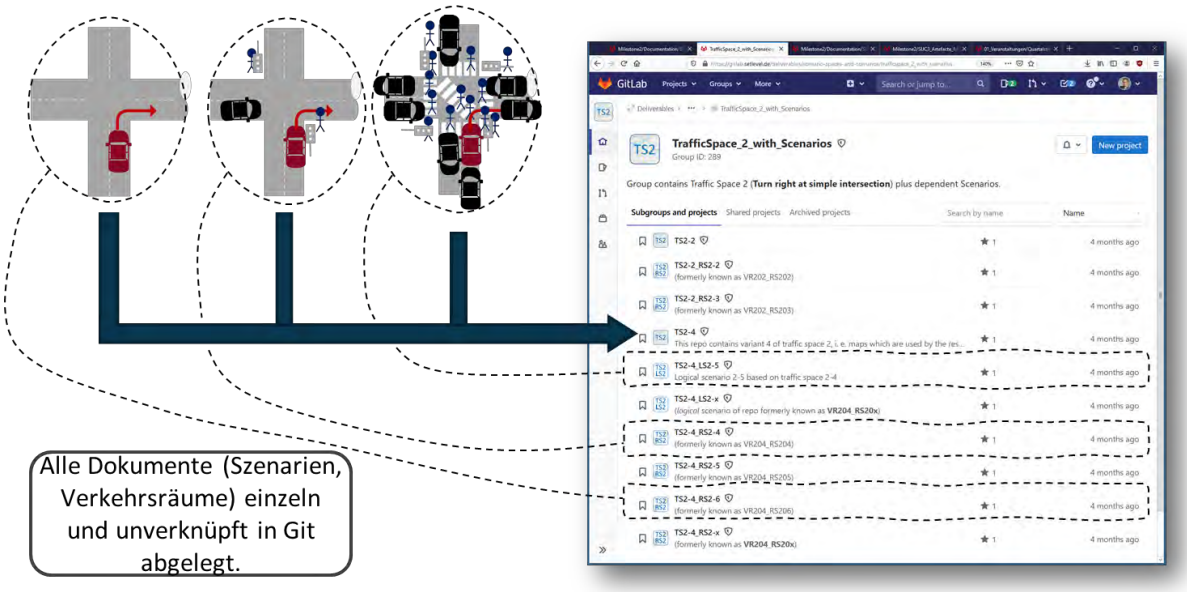


Abbildung 94: Ablage der Traffic Spaces in GitLab

Auf diese Traffic Spaces wird in den CSP-konformen Dokumentationen der SUCs, die mit Unterstützung von AP 2.1 erstellt wurden, per URL verwiesen (Abbildung 95):

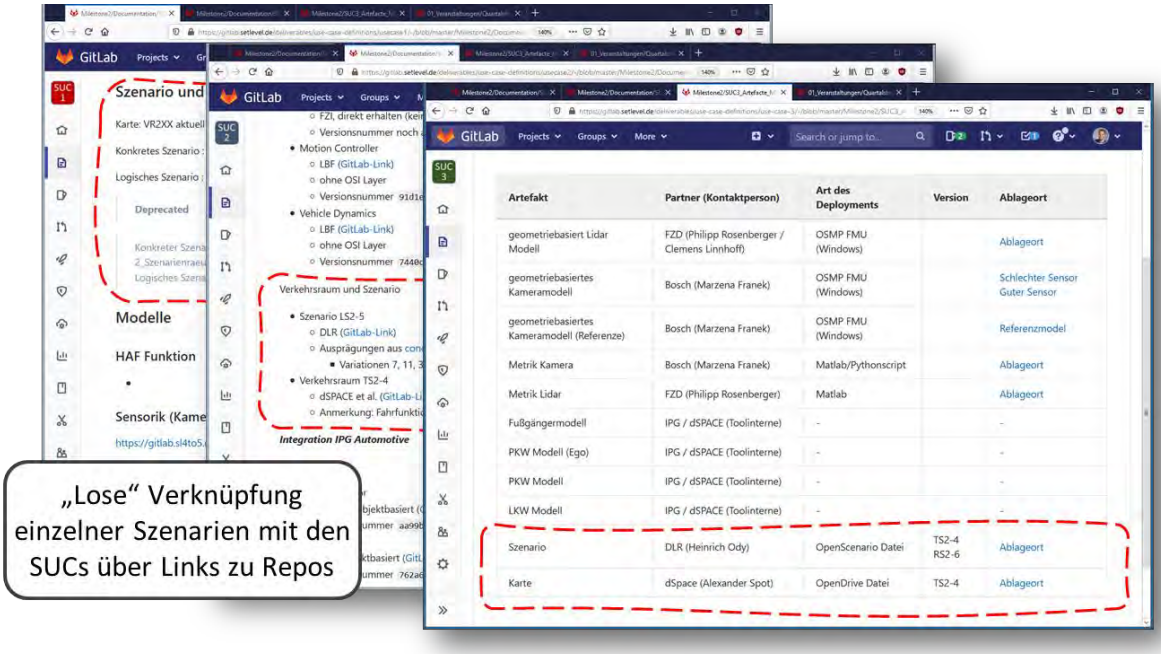


Abbildung 95: SUC und Szenarien verknüpft

Somit ist ein *Rückwärtsverweis* vom SUC auf seine Quellen gegeben.
Betrachtet man jedoch umgekehrt eines der Traffic-Space-Repositories, so ist *nicht* ersichtlich, in welchen Simulationen dieser Traffic Space verwendet wird:

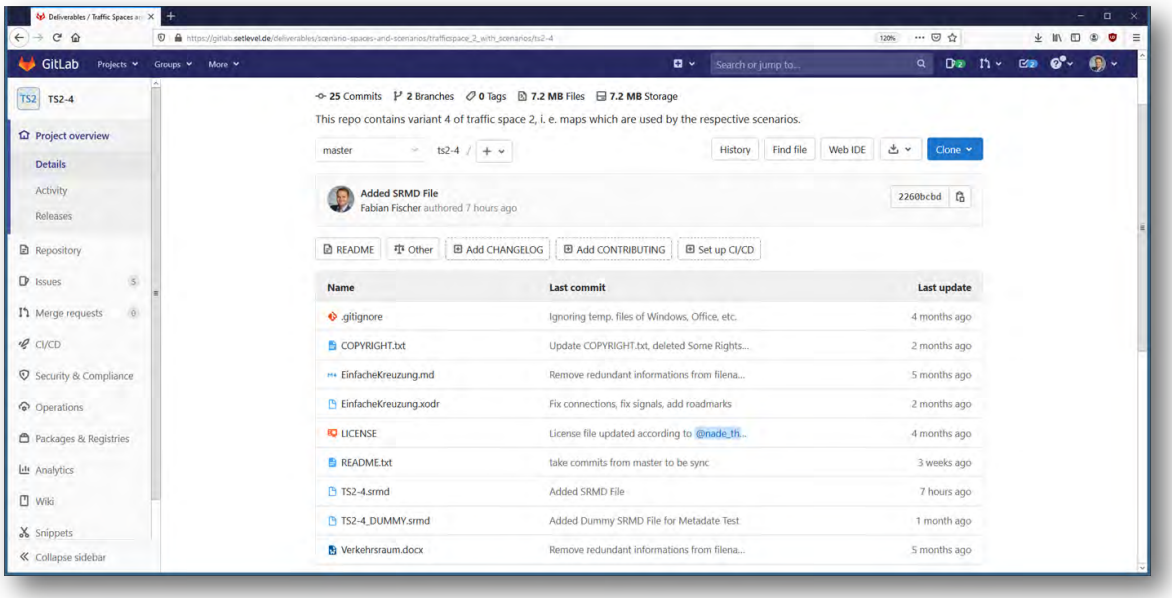


Abbildung 96: Traffic Space Repository

Benötigt wird hier eine *Vorwärtssuche*.

Die Traceability-Software „TRACY“ ermöglicht es in diesem Fall, über die Verknüpfung von SUC und Traffic Space via Trace-Links sowohl eine Vorwärts- als auch eine Rückwärtssuche durchzuführen:

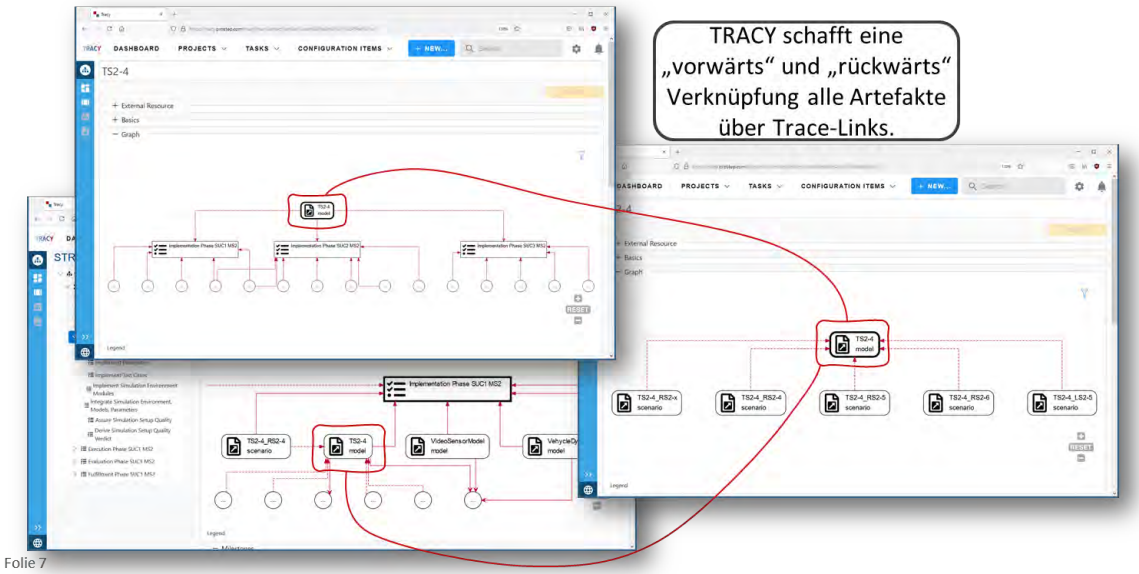


Abbildung 97: Trace-Links in TRACY

Mittels der dokumentierten Ablagestruktur und der Software TRACY konnte eine effiziente projektweite Zusammenarbeit zwischen den SUCs (TP 4) und den APs, die sich mit Verkehrsräumen, Referenzszenarien und erweiterten Szenarienbeschreibungssprachen (jew. AP 1.2) und der Spezifikation der Szenarienbeschreibung (AP 2.1) befassen, etabliert werden.
Ab 2021 fanden Austauschgespräche statt mit ADScene sowie der University of Warwick, welche „Safety Pool“ bereitstellt (siehe Abschnitt 1.5.2). In beiden Projekten wird an Szenario-Datenbanken gearbeitet.

Eigenes und fremdes geistiges Eigentum, Lizenzen

Die Entwicklung hoch automatisierter Fahrfunktionen erfordert eine sehr intensive Zusammenarbeit zwischen vielen Beteiligten, von Automotive OEMs über Tiers und Dienstleistern hin bis zu Forschungsinstituten. Dabei müssen Rechte an geistigem Eigentum (Intellectual Property, IP) allerdings immer gewahrt bleiben.
Dies gilt selbstverständlich auch für Rechte an Werken, die im Projekt SET Level verwendet werden, als auch für Werke, die im Rahmen von SET Level entstanden. Bei diesen Werken handelt es sich vor allem um folgende Typen:

- Software (im eigentlichen Sinne oder als Bestandteil von Verhaltensmodellen)
- Modelle von Verkehrsräumen und Szenarien
- Veröffentlichungen (wissenschaftliche Publikationen, Vorträge usw.)
- Sonstige Dokumentationen von Projektergebnissen

PROSTEP und BMW förderten daher gemeinsam eine entsprechende Diskussion und Bewusstseinsbildung bei den Projektteilnehmern. BMW benannte Experten, die als Ansprechpartner zur Verfügung standen. Es zeigte sich, dass für die weitere Nutzung von Projektergebnissen im Allgemeinen die aus der Software-Welt stammende Lizenzbildung „Creative Commons Attribution-ShareAlike 4.0 (CC BY-SA 4.0)" (siehe https://creativecommons.org/licenses/by-sa/4.0) angemessen ist. Sie wurde projektweit empfohlen, wobei jeder Urheber

unvermindert das Recht behielt, sich für andere Regelungen zu entscheiden. Es wurde allerdings verlangt, dass diese Entscheidung zusammen mit dem Werk, also im selben Repository, dokumentiert wird. PROSTEP prüfte die Einhaltung dieser Regel regelmäßig über einen wesentlichen Zeitraum des Projekts hinweg.
BMW engagierte sich für die Dokumentation der zu Modellen, Szenarien und Karten notwendigen Lizenzen. Beispielhaft wurden daher ALKS-Szenarien auf der TPX-Website bereitgestellt und hierfür die o. g. Lizenzbildung CC-BY-SA-4.0 ausgewählt.

2.3.3.1.2 Simulationsziele

Ergebnis der Arbeiten, welches durch das Fraunhofer LBF moderiert wurden, ist die Spezifikation einer Schnittstelle zu den Simulationszielen. Die Simulationsziele sollen im Entwicklungsprozess die Abbildung der Prozess- und Produktanforderungen auf die Fähigkeiten der Simulationsprozesse unterstützen. Dazu wurden die Anforderungsaspekte der Simulation Use Cases SUC 1-3 und die Prozessschnittstellen des Credible Simulation Process‘ aus AP 3.1 mit den verschiedenen Arbeitspaketen, welche diese Inhalte adressieren, abgeglichen.
Die Arbeiten waren darauf gerichtet, ein methodisches Vorgehen zur spezifischen Gestaltung und Skalierung der Simulationsumgebung für einzelne Simulationsaufgaben auf Basis der Simulationsziele zu beschreiben. Dabei werden die in Abbildung 98 dargestellten Wechselwirkungen mit dem Simulationsprozess berücksichtigt.

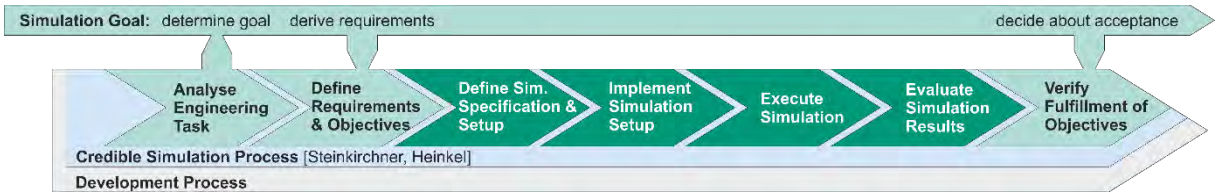


Abbildung 98: Die Simulationsziele wechselwirken mit den Schritten des Simulationsprozesses

Folgende für das Projekt SET Level relevante Simulationsziele wurden unter den Aspekten **A**nalyse, **O**ptimierung und **T**est definiert:

1. Exploration des Szenarienraums [A]
2. Bestimmung der Kritikalität [A]
3. Plausibilisierung von kausalen Zusammenhängen [A]
4. Beitrag zum Nachweis einer positiven Risikobilanz [A]
5. Nachweis ausreichender Abdeckung des Parameterraums [A]
6. Rückmeldung zu HW/SW/Konfig-Optimierungen [O]
7. Rückmeldung zur Entwicklung des Funktionscodes mit MiL/SiL-Simulationen [T]
8. Rückmeldung zur Komponenten- und Systementwicklung mit HiL-Simulationen [T]
9. Rückmeldung zur Umgebungswahrnehmung mit Open-loop HW-Reprozessierung [T]
10. Nachweis: Leistungseigenschaften und Degradationsverhalten wie spezifiziert [T]

Die Simulationsziele dienen als Grundlage für die Erfolgsbewertung einer Simulationsaufgabe, welche im Kontext von Entwicklungs- bzw. Testaktivitäten steht. Dabei wurde explizit eine Trennung der Simulationsziele in Entwicklungsaktivitäten und Testaktivitäten vorgenommen. Weiterhin wurde Entwickeln in Analysieren und Optimieren unterschieden. Die mit Simulationszielen gestützten Simulationsaufgaben leisten in der Regel einen Beitrag zur

Erfüllung von Entwicklungs- und Testzielen. Die Ziele wurden durch eine eindeutige Benennung, eine Beschreibung des Entwicklungskontextes und eine Definition eines Kriteriums für die Erfüllung formal beschrieben und wurden mit den Simulation Use Cases abgeglichen. Die Ergebnisse wurden dem Projektkonsortium präsentiert, zur Diskussion gestellt und durch das Projekt angenommen. Die formale Beschreibung der Simulationsziele kann für die Implementierung konkreter Instanzen von Simulationsketten herangezogen werden. Dies wurde am Beispiel der Simulation Use Cases gezeigt.

2.3.3.1.3 Architektur

Im Bereich „Architektur“ arbeitete das AP 4.1 eng mit AP 2.1 sowie mit den APs 4.2 bis 4.4 (den drei SUCs) und TP 3 (numerische Simulationsmodelle) zusammen.
Eine der Grundmotivationen des Projektes SET Level rührte aus der Erkenntnis, dass fortgeschrittene Fahrassistenzsysteme (ADAS) längst nicht mehr ausschließlich durch physikalische Tests verifiziert werden können. Simulation ist in sehr großem Umfang erforderlich. Eine integrative Simulationsarchitektur von Sensormodellen über Verkehrsszenarien bis hin zu Werkzeugen und Prozessen bietet eine Grundlage, um die entstehenden technischen und organisatorischen Herausforderungen bei der Simulation zu meistern. Sie hilft, Synergien zu nutzen und redundante Arbeiten sowie Fehlerquellen zu vermeiden, insbesondere in großen, heterogenen oder verteilten Teams.
Solch eine Simulationsarchitektur muss also insbesondere

- Orientierung bieten,
- eine hierarchisch beliebig tiefe Dekomposition je nach Simulationsziel ermöglichen,
- Bereiche der Architektur zulassen, die aus IP-Gründen nicht offengelegt werden,
- entsprechend modular aufgebaut sein,
- Definition und Standardisierung von Schnittstellen unterstützen,
- je nach Simulationsziel und Kontext unterschiedliche Sichten verschiedener Detaillierungsstufen ermöglichen.

Allein im Bereich der Werkzeuge existiert eine Reihe relevanter Software-Systeme. Neben den im Projekt vertretenen Implementierungen der Firmen dSPACE, ETAS und IPG sind die am Markt vertretenen Werkzeuge der Firmen Ansys, Hexagon, Mathworks, Siemens und weiteren Firmen zu erwähnen.

In anderen Bereichen der Architektur ist die Vielfalt mindestens ebenso groß.
Im Projekt wurden die daraus erwachsenden Herausforderungen mit modernen Methoden angegangen, die eine simulationsbasierte Analyse, Entscheidungsfindung (SmartSE project group, 2021) und Verifizierung ermöglichen. Die Methoden entstammen dem Model Based Systems Engineering (MBSE) und führen uns nahtlos von der abstrakten Systemarchitektur zu implementierbaren Simulationsmodellprototypen.
Um dieses Ziel zu erreichen, wurde das bewährte RFLP-Muster (Eigner, 2017) auf das Gesamtsystem der Simulation als solches angewendet. Das Gesamtsystem besteht nicht nur aus dem Simulationsmodell, sondern auch aus dem Simulator und der gemeinsamen Einbettung in einen glaubwürdigen Simulationsprozess zur Erfüllung einer Simulationsaufgabe. Der Vielfalt und Komplexität der realen Verkehrsteilnehmer wird durch ein Rahmenmodell mit konfigurierbaren Komponenten Rechnung getragen.
Die Architektur der untersten Abstraktionsebene kann direkt für den Zusammenbau des Modells in einer Simulationssoftware verwendet werden.

Als Modellierungssprache wird eine Teilmenge der standardisierten und weit verbreiteten Unified Modeling Language (UML)[8] verwendet. Dies garantiert, dass die Methode einfach zu verstehen und unabhängig vom Modellierungswerkzeug ist.
Belastbare Simulationsergebnisse erfordern eine Architektur, die eng mit einem glaubwürdigen Prozess verbunden ist, wie dem Credible Simulation Process (CSP)[9].
Zusammen ergibt sich daraus auch ein konsistenter Beitrag zum Enterprise Architecture Modeling (EAM) der Organisation.

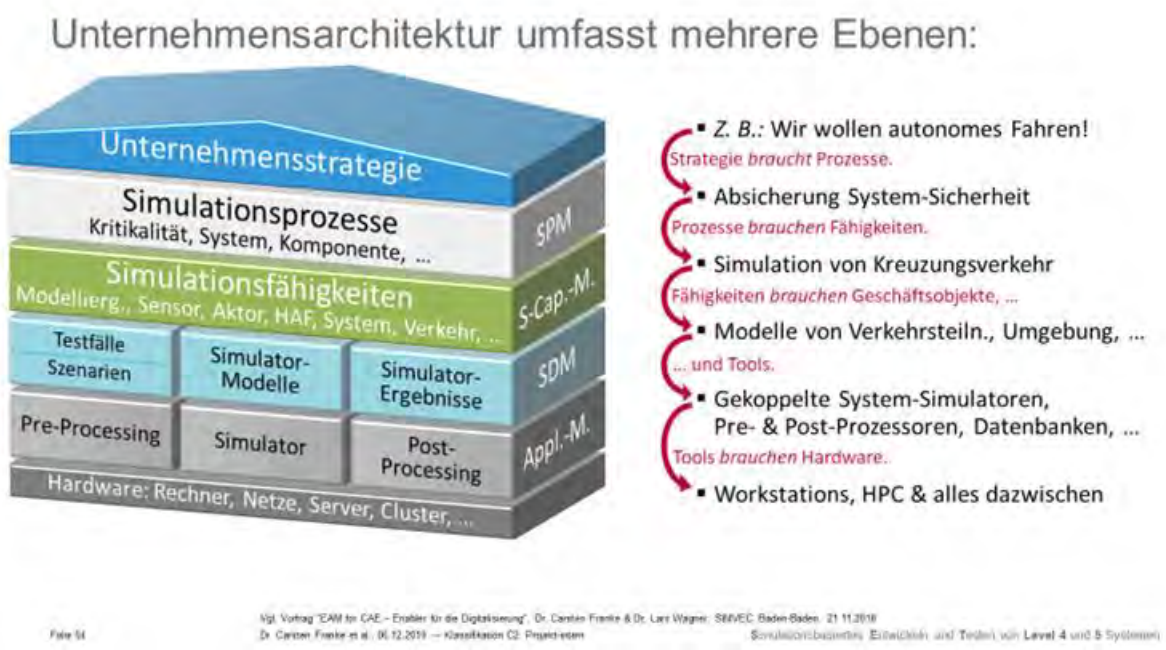


Abbildung 99: Schematische Darstellung der Ebenen der Unternehmensarchitektur im Kontext Simulation

Eine klug gestaltete Unternehmensarchitektur bringt auch wirtschaftliche Vorteile mit sich: Je mehr Komponenten der Architektur je allgemeiner verwendbar sind, umso besser lassen sich Entwicklungsgegenstände wiederverwenden, umso ökonomischer lassen sich Software-Lizenzen und Fortbildungsmaßnahmen einsetzen, umso weniger Overhead entsteht durch zusätzliche Schnittstellen, und umso kostengünstiger fällt eine Qualifizierung einer Tool-Architektur für Produktfreigaben aus.
PROSTEP hat in diesem Kontext sein Know-how im Bereich Unternehmensarchitektur und Systems Engineering eingebracht.
Dokumentation und Spezifikation der Integrationsarchitektur im AP 4.1 erfordern eine systematische und strukturierte Herangehensweise. Deshalb wurden Methoden der Modellierung von Unternehmensarchitektur herangezogen (siehe Abbildung 99).
Randbedingungen und Zielsetzung des Projekts bestimmen, dass von diesen Ebenen vor allem Prozesse, Daten und Werkzeuge, relevant sind. Es wurde identifiziert, dass sich diese drei Ebenen, wie abgebildet, auf die TPs 2, 3 und 4 des Projektes verorten (siehe Abbildung 100).

[8] Vgl. OMG. Unified Modeling Language (UML), Object Management Group® (OMG®), Milford, MA, USA. https://www.omg.org/spec/UML/About-UML/, 2017.

[9] Siehe u. a.
- Heinkel, H.-M., Steinkirchner, K., and the SET Level project. Credible simulation process. https://setlevel.de/assets/forschungsergebnisse/Credible-Simulation-Process-v1.0.pdf., 2021.
- Heinkel, H.-M. and Vettermann, S. Realizing traceability for safety and certainty. In *prostep ivip Symposium 2020*, Darmstadt. prostep ivip Verein, 2020.

Abbildung 100: Architektur-Dreibein

Das AP 4.1 fokussierte sich demnach auf die Tool-Achse, benötigte für seine Arbeit aber auch sehr die Architektur der Inhalte. Deshalb wurde eine enge und intensive Zusammenarbeit mit AP 2.1 etabliert.

Methode der integrativen Architektur der Simulation

Dem allgemeinen Ansatz des Projekts entsprechend konnte es *nicht* darum gehen, eine feste, für alle Partner verbindliche Architektur der Simulation zu formulieren. Vielmehr musste eine *Methode* erarbeitet werden, die es jedem Partner ermöglicht, diese Architektur gemäß seinen Bedürfnissen auszugestalten und dabei von den beschriebenen Vorteilen zu profitieren.
An der Entwicklung der gesuchten *Methode der integrativen Architektur der Simulation* haben sich u. a. BMW, Conti ADC, DLR, OFFIS, Opel, PROSTEP, VW und ZF über mehrere Quartale hinweg beteiligt. Die intensive Zusammenarbeit der APs 2.1 und 4.1 bis 4.3 zeigte bald essenzielle Fragen auf, die nur gemeinsam gelöst werden konnten:

1. Wie ist sichergestellt, dass Architektur-Sichten (Diagramme) nachhaltig zusammenpassen? (vgl. Abbildung 101)
2. Welche Architektur-Sichten sind relevant für wen?
3. Wie lassen sich Prozess- und Modell-Architektur optimal miteinander in Verbindung setzen? (Bezug zum CSP, vgl. TP 3)

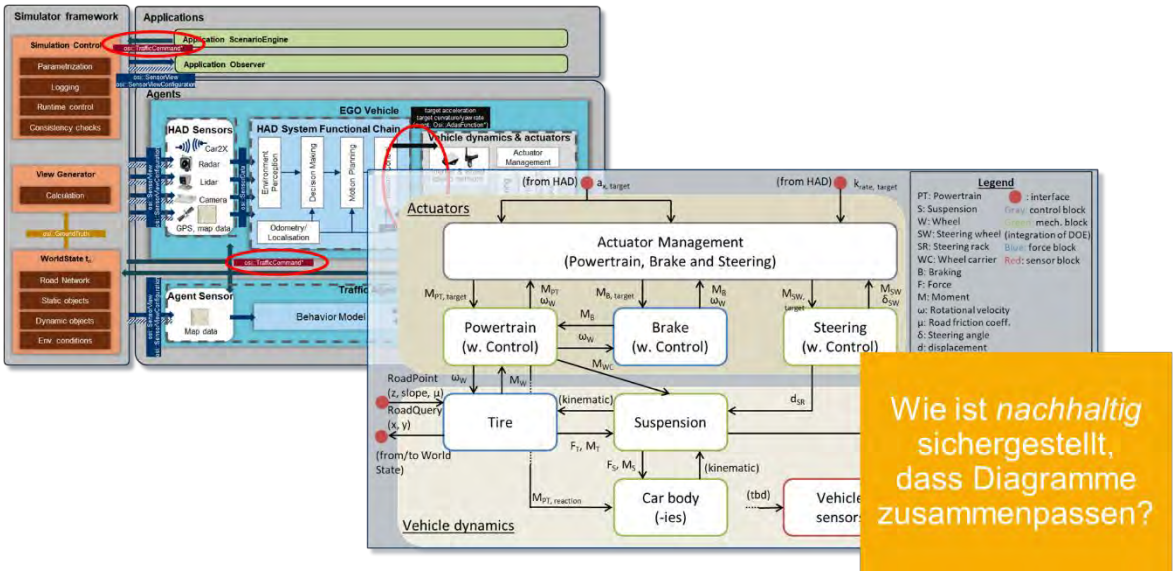


Abbildung 101: Nachhaltige Passung von Architektur-Sichten

Die Passung der Architektursichten wurde durch den Einsatz eines einheitlichen Design-Werkzeugs (Sparx Systems‘ Enterprise Architect, kurz EA) zur UML-Modellierung erheblich unterstützt. Dafür gab es viele gute Gründe:

- Die schiere Größe des Modells zwingt zum Einsatz eines geeigneten Werkzeugs.
- Es überträgt automatisch Änderungen von einem Element / Diagramm auf alle anderen Vorkommen,
- sichert dadurch, dass die Konsistenz gewahrt bleibt,
- und hilft, mühsame Arbeit und Fehlerquellen zu reduzieren.
- Das UML-Modell kann die Architektur einer Simulation präzise und unmissverständlich beschreiben. Dadurch fördert es zugleich Bestrebungen zur Standardisierung.
- Ein grafisches „Inhaltsverzeichnis“ (vgl. Abbildung 106) verschafft auch denjenigen einen Überblick, die keine Experten auf einem bestimmten Gebiet sind.

PROSTEP hat ein dazu passendes Konzept zum Betrieb eines gemeinsamen Datenbankservers für den EA als *Single Source of Truth* entwickelt. Der Server wurde vom DLR betrieben. Die Repräsentation in genau diesem Tool ist dabei nur als Beispiel zu verstehen. Andere SysML- und UML-Werkzeuge lassen sich hier genauso verwenden, denn es wurde im Sinne möglichst allgemeiner Verwendbarkeit darauf geachtet, möglichst grundlegende, gemeinsame und allgemein etablierte Konstrukte aus diesen Sprachen zu verwenden.
Neben der reinen Kollaboration förderte diese technische Infrastruktur auch die nachhaltige Konsistenzsicherung der Architektur-Sichten.
Zur *Methode der integrativen Architektur der Simulation* erschienen Präsentationen mit Artikeln beim NAFEMS World Congress 2021 und der VDI-Konferenz SIMVEC 2022.
Beim Abschluss-Event des Projektes war dem Thema ein Marktstand gewidmet. Entsprechend wurde ein Poster, ein Foliensatz und eine Vortragsaufzeichnung veröffentlicht.
Die Fortschritte und die Bedeutung des Themas „Architektur“ manifestierten sich darin, dass dazu in einer Serie von Quartalstreffen jeweils ein Hauptvortrag stattfand.

Model-based Systems Engineering (MBSE)
Mit zunehmender Komplexität eines Systems wird es immer schwieriger, alle Abhängigkeiten zwischen seinen Komponenten zu erfassen. Wie Flugzeuge und Satelliten bestehen auch hochautomatisierte Fahrzeuge aus einer Vielzahl unterschiedlicher Komponenten, die von Ingenieuren aus den Bereichen Mechanik, Elektrik/Elektronik und Software entwickelt wurden. Systems Engineering (SE) ist ein interdisziplinärer Ansatz, um die Realisierung solcher komplexen Systeme zu bewältigen. Er „*integriert alle Disziplinen und Fachgruppen in eine Teamarbeit und bildet einen strukturierten Entwicklungsprozess, der vom Konzept über die Produktion bis zum Betrieb reicht*“ (Walden, 2015). Dies geschieht durch die Zerlegung des Systems in überschaubare Teilsysteme mit definierten Grenzen und Schnittstellen, die dann von den jeweiligen Spezialisten entwickelt werden können.
Eine der großen Herausforderungen ist nach wie vor die Koordination zwischen den verschiedenen beteiligten Disziplinen. Hierfür schlägt das INCOSE[10] den MBSE-Ansatz vor, der „*die formalisierte Anwendung der Modellierung zur Unterstützung der Systemanforderungen, des Entwurfs, der Analyse, der Aktivitäten zu Verifikation und Validierung, beginnend in der Konzeptphase und fortlaufend über Entwicklung und spätere Lebenszyklusphasen*“ darstellt (Systems Engineering Vision Working Group, 2007). Die bevorzugten Modellierungssprachen

[10] International Council on Systems Engineering, https://www.incose.org

sind grafische Sprachen wie UML oder SysML[11], die den Vorteil einer universellen Darstellung des Systems über alle Domänen hinweg bieten.
In Kombination mit leistungsfähigen Modellierungswerkzeugen ermöglicht dies die Erstellung einer komplexen, aber konsistenten Systemarchitektur über alle Domänen und die verschiedenen Phasen der Systementwicklung hinweg.

RFLP – ein MBSE-Ansatz
RFLP (Eigner, et al., 2014) ist einer der bekanntesten Ansätze zur Strukturierung von MBSE. Ursprünglich wurde er für die Entwicklung technischer *Produkte* entlang des V-Modells entwickelt.

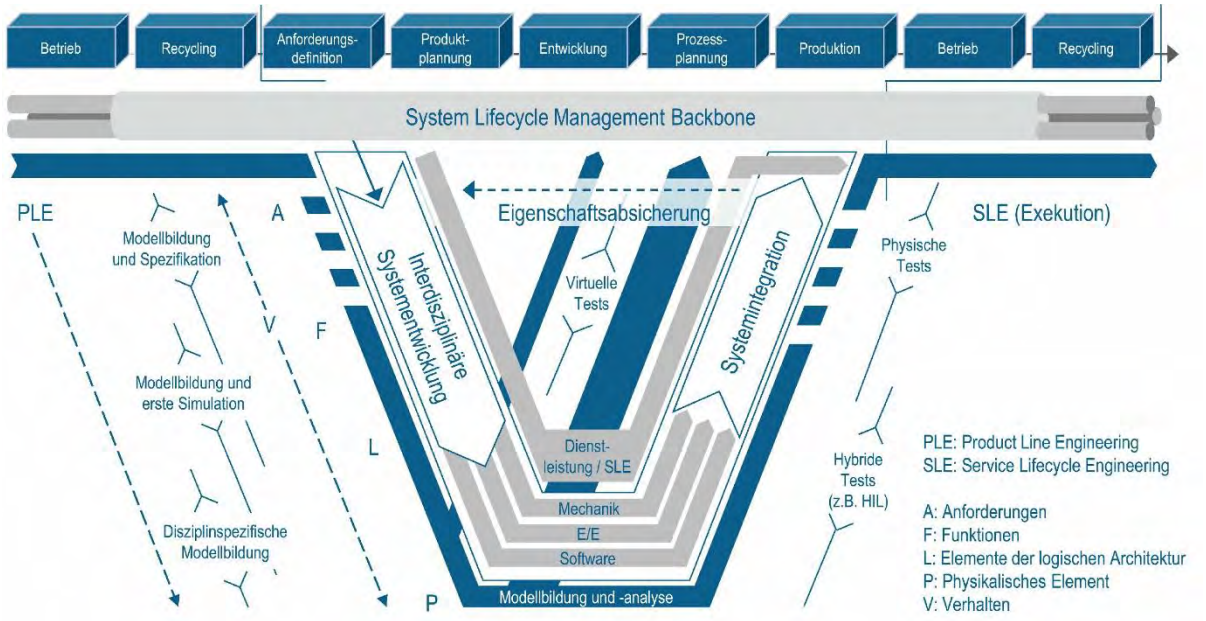


Abbildung 102: RFLP-ausgerichtetes V-Modell

Dementsprechend wurde eine RFLP-basierte Methode zur Beschreibung der Architektur von *Simulation* als Gesamtsystem entwickelt. Wie in Abbildung 102 dargestellt[12], differenziert RFLP die Ebenen

- der Anforderungen sowie der
- funktionalen,
- logischen und
- physikalischen Architektur und Modelle.

Anforderungen werden als Input für den Ansatz angenommen und z. B. von Kunden gestellt. Die *funktionale Architektur* beschreibt die Architektur der Funktionen des Systems, die für die Erfüllung der Anforderungen nötig sind oder auch physikalische und strukturelle Notwendigkeiten repräsentieren. Im Allgemeinen können Funktionen hierarchisch unterteilt werden.
Die *logische Architektur* definiert Komponenten, die die Funktionen implementieren. Je nach Bedarf werden in der Regel Funktionen mit geringer Relevanz für die aktuelle Simulationsaufgabe weggelassen oder vereinfacht. Häufig benötigt eine einzelne Funktion mehr als eine logische Komponente zur Implementation, und eine einzige logische Komponente unterstützt die Implementierung mehrerer verschiedener Funktionen. Daraus ergeben sich *n*:*m*-

[11] Vgl. OMG. System Modeling Language (SysML), Object Management Group® (OMG®), Milford, MA, USA. https://www.omg.org/spec/SysML/1.6/About-SysML/, 2019.

[12] Vgl. Bild 2 aus Eigner, M., Dickopf, T. und Apostolov, H. Interdisziplinäre Konstruktionsmethoden und -prozesse. *VDI e-Paper Reihe „Konstruktion“*, 09.11.2018.

Beziehungen. Innerhalb der logischen Architektur kann eine Unterebene zur Definition der technischen Basis der Komponenten und ihrer Schnittstellen abgegrenzt werden.
Die *physische Architektur* im Sinne von RFLP beschreibt das eigentliche, meist greifbare Produkt. In unserem Kontext übertragen wir diesen Begriff auf das *Simulationssystem* (Simulationsmodell, das für einen Simulationslauf bereit ist, ausgeführt von einem Simulatorkern) sowie auf die *Simulationsaufgabe* im Rahmen eines *Simulationsprozesses*. Um diesen Unterschied zu verdeutlichen, wurde diese Ebene *konkret* statt physisch genannt.
Die konkrete Architektur enthält also insbesondere die tatsächliche Anzahl von *instanziierten Objekten*, z. B. 8 konkrete Fußgänger-Instanzen desselben logischen Fußgänger-Modells. Wenn Modelle Parameter enthalten, muss die konkrete Architektur Regeln für deren Belegung bzw. Diskretisierung angeben. So werden parametrisierte Serien von Simulationsläufen für dieselbe Simulationsaufgabe erzeugt, wie sie z. B. für Optimierungsanwendungen oder Kritikalitätsanalysen typisch sind.
Der RFLP-Ansatz wurde den Kenntnisssen des Projekts entsprechend erstmals von der Entwicklung eines Produkts auf die Simulation als „Gesamt-Problemstellung“ übertragen.

Die Orientierungsmatrix der integrativen Architektur der Simulation

Eine erste Visualisierung dieses „Drei-Ebenen-Ansatzes“ zur Architekturbeschreibung verdeutlichte bereits die wesentliche Grundstruktur der integrativen Architektur der Simulation in einer Orientierungsmatrix, noch Office-basiert (vgl. Abbildung 103). Dabei bilden die Domänen bzw. Architektur-Gegenstände die hierarchische Gliederung der gesamten Simulation in Spalten. Diese Matrix konnte bereits als Schablone zur Sortierung von Modellen, Formaten, Tools etc. dienen.

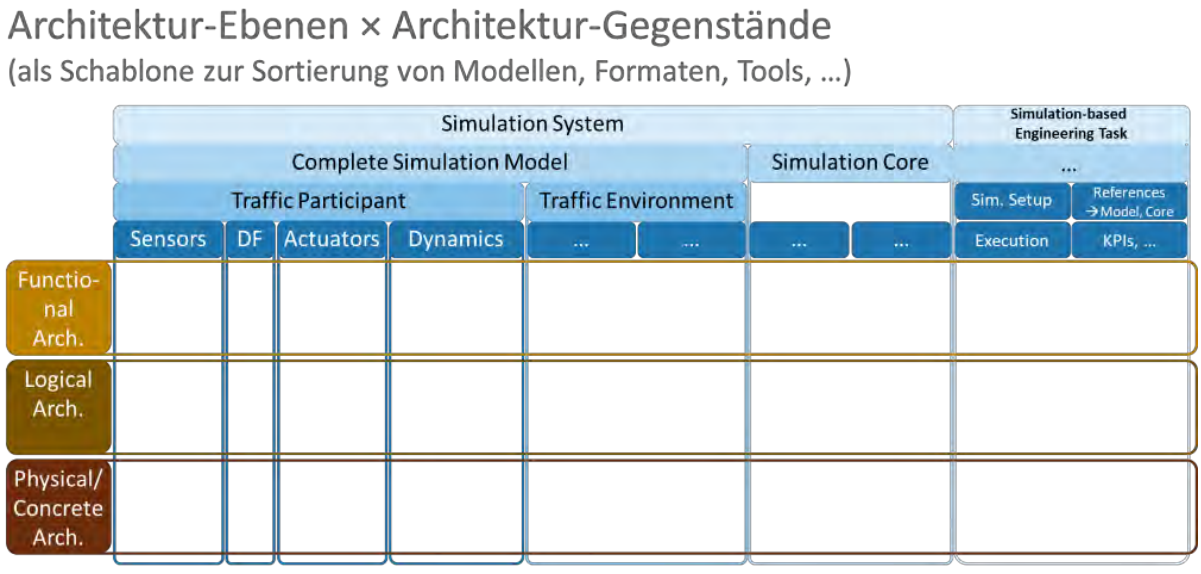


Abbildung 103: Erster Entwurf der Orientierungsmatrix

Die „Systems under Test“ (SuTs) der drei SUCs des Projekts verorten sich in unterschiedlichen Spalten der Schablone – von links nach rechts:

- Der SUC 3 befasst sich mit der Absicherung von Komponenten des Traffic Participants. Sein SuT ist also in *einer* der Spalten unterhalb „Traffic Participant“ angesiedelt.
- Der SUC 2 betrachtet einen kompletten Traffic Participant. Der SuT überdeckt daher den gesamten Bereich *aller* Spalten unterhalb „Traffic Participant“.

- Im SUC 1 schließlich geht es weniger um einen einzelnen Traffic Participant, sondern um kritische Situationen, die sich im Allgemeinen aus der *Interaktion mehrerer Traffic Participants* ergeben. Sein Ergebnis sind daher kritische Szenarien. Sie befinden sich im Schema unterhalb “Traffic Environment“.

Diesen Ansatz weiterverfolgend, gelangte das Projekt zu der Erkenntnis, dass sich in der Orientierungsmatrix auch Flüsse von Signalen, Informationen und Daten verorten lassen. Zusätzlich lassen sich die Bedarfe nach horizontaler und vertikaler Traceability sowie nach Glaubwürdigkeit (Credibility) darstellen:

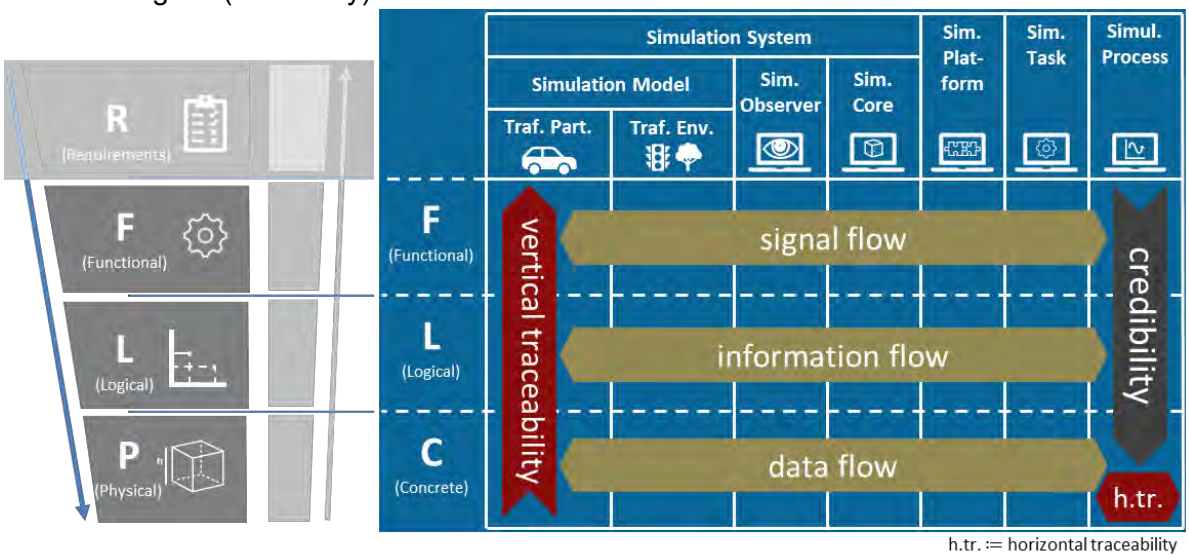


Abbildung 104: Endstand der Orientierungsmatrix

Die Orientierungsmatrix in dieser Gestalt

- adaptiert das RFLP-Muster aus MBSE an das Gesamt-Simulationssystem (eine Spalte pro Teilsystem/Domäne),
- ist Mittel zur Strukturierung und effizienten Kommunikation, (Jedes Feld enthält einen überschaubaren Teil der Architektur.)
- dient der Vollständigkeitsprüfung,
- stützt Informationsdurchgängigkeit und Passung der Schnittstellen „by design“, (Die zeilenweise Durchquerung erzeugt eine vollständige Liste der Schnittstellen.)
- veranschaulicht horizontale und vertikale Traceability,
- hilft Fehlerquellen zu reduzieren,
- um schließlich einen glaubwürdigen Simulationsprozess zu erreichen.

Im Bereich *Zusammenarbeit und Automatisierung* schafft die Orientierungsmatrix effiziente Kommunikation durch

- gemeinsame Sicht auf die Systemstruktur,
- gemeinsames Verständnis der Diskussionsebene
- und gemeinsames Wording.

Diskussionen können in einem Feld eingehegt werden. Das verhindert Missverständnisse und sorgt für *Effizienz*. Gemeinsames Verständnis und stabile Schnittstellen sind Voraussetzungen für die *Automatisierung*.

SSP-Dateien, ausgeleitet aus der konkreten Ebene des Architekturmodells im EA, erleichtern das automatisierte Modell-Assembly.
Die Orientierungsmatrix unterstützt die *Projekt- und Prozessplanung* bzgl.:

- Abschätzung des Bedarfs an Modellen und der dafür benötigten Zeit.
- Konfigurationsmanagement gemäß ISO 10007.[13]
- Ressourcenplanung: Menschen, Software, Hardware.

Auch die Wiederverwendung von Modellen (Re-use) wird durch die Orientierungsmatrix gefördert: Stabile, nachhaltige und effiziente Schnittstellen sind Voraussetzung für die Wiederverwendung von Komponenten. Auch Synergieeffekte zwischen verschiedenen Simulationstypen können so genutzt werden.

Safety First for Automated Driving (sog. SaFAD-Paper)

Im April 2019 erschien der Report „Safety First for Automated Driving“, der von namhaften Unternehmen der Automotive- und IT-Branche gemeinsam erstellt worden war (SaFAD, 2019). Er beschreibt unter anderem die funktionale Architektur eines realen hoch automatisierten Fahrzeugs, vgl. Abbildung 105. Aus diesem Papier entstand im Jahr darauf der *Report ISO/TR 4804:2020* (ISO). Als Nachfolger ist *ISO/AWI TS 5083* (ISO) bereits in Arbeit. Das SaFAD-Paper samt Nachfolger stellte einen wichtigen Ankerpunkt dar, gegen den die Arbeiten des Projekts SET Level kontinuierlich abgeglichen wurden. Es bestätigte den RFLP-Ansatz derart, dass auch das SaFAD-Paper auf der funktionalen Ebene ansetzte. Der Vergleich der Architekturen des *realen* und des *simulierten* Fahrzeugs lieferte wichtige Erkenntnisse. Diese Untersuchungen fanden im oberen linken Feld der Orientierungsmatrix (Zeile *Functional*, Spalte *Traffic Participant*) statt.

[13] Siehe https://www.iso.org/standard/70400.html

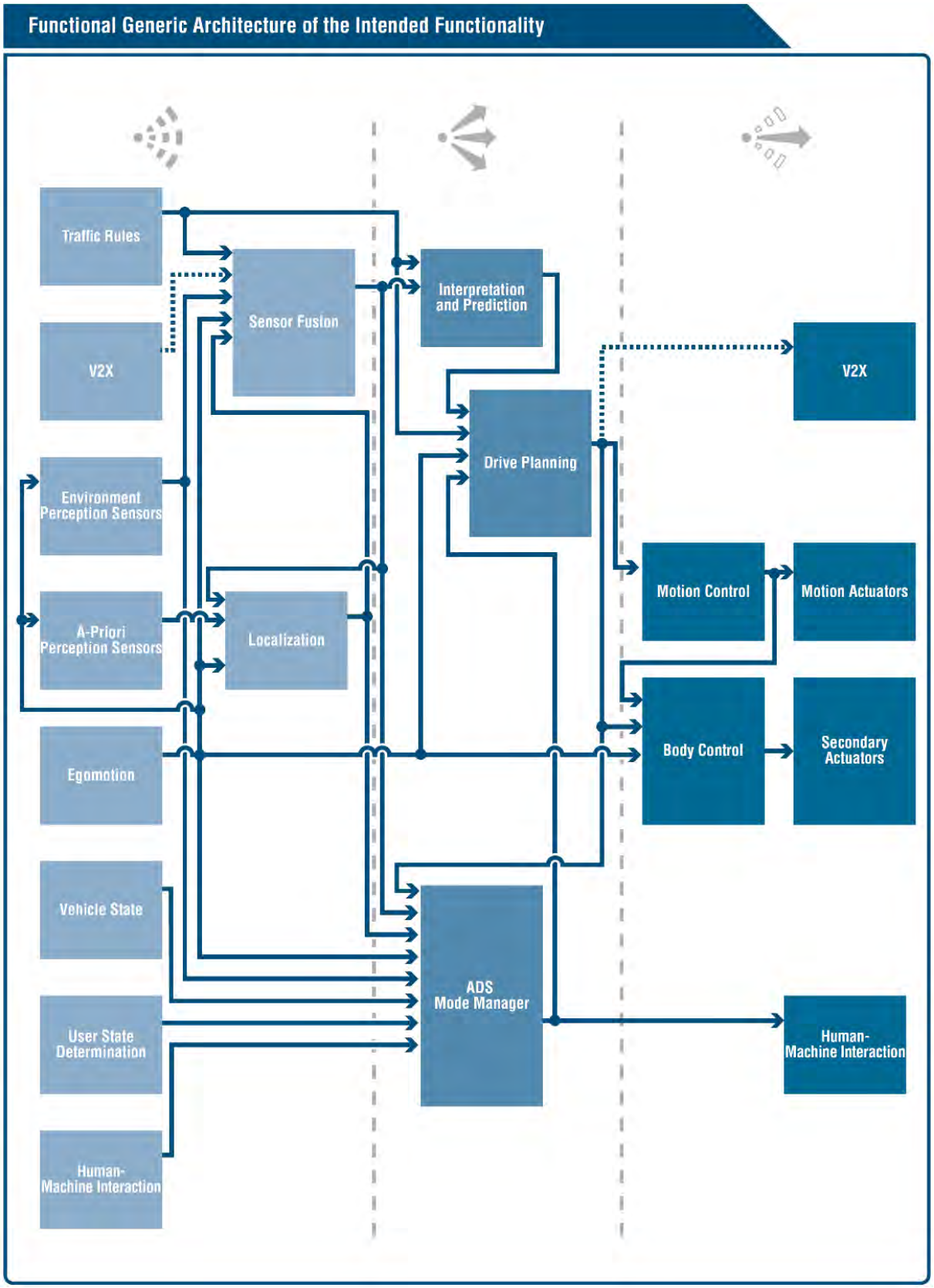


Abbildung 105: Funktionale Architektur eines realen Fahrzeugs (Abbildung 26 des SaFAD-Papers)

UML-Modell der integrativen Architektur der Simulation mit Anwendung

Die Orientierungsmatrix kann, trotz der vielen oben aufgezählten Vorzüge, *nur eine Sicht von vielen* auf die Gesamtarchitektur bzw. das gesamte Architekturmodell sein. Für die tägliche Arbeit werden viele verschiedene Sichten für unterschiedliche Fragestellungen mit

unterschiedlichen Schwerpunkten und unterschiedlichen Detaillierungsgraden benötigt. Bereits weiter oben wurde beschrieben, dass im Projekt der Einsatz eines geeigneten Tools (Enterprise Architect, kurz EA) entschieden wurde. Mit diesem Tool wurde ein UML-Modell der integrativen Architektur der Simulation erstellt und mit Fokus auf die SUCs 1 und 2 für die praktische Anwendung validiert. Die gewonnenen Erkenntnisse von der allgemeinen Vorgehensweise bis hin zu effizienten „Klickreihenfolgen“ im Werkzeug wurden in einer entsprechenden Modellierungsrichtlinie gesammelt.
Selbstverständlich wurde auch die Orientierungsmatrix in UML dargestellt, um die vielen einzelnen Architekturkomponenten des Gesamtmodells zu sortieren und eine Art *Inhaltsverzeichnis* zur Verfügung zu stellen:

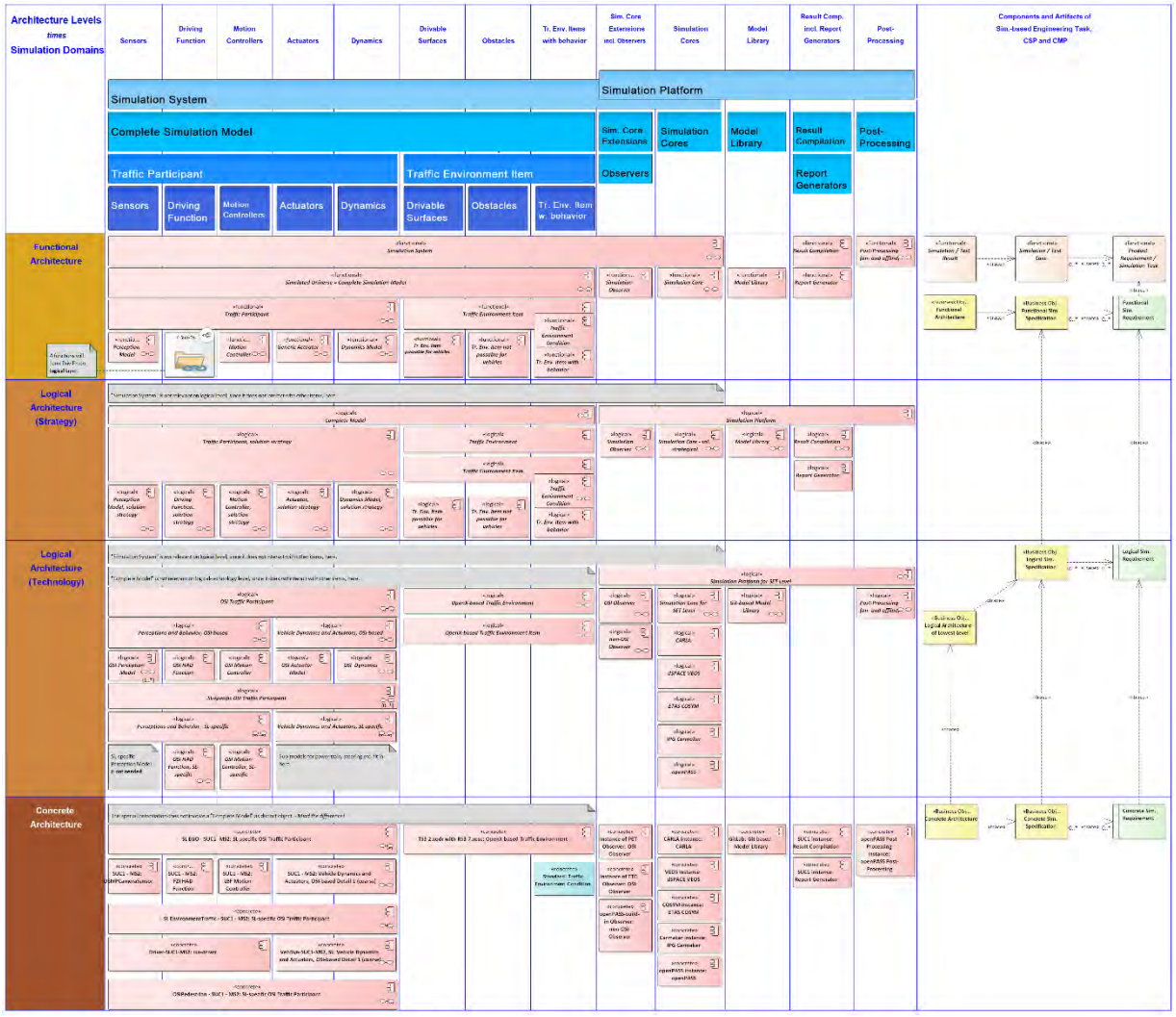


Abbildung 106: Orientierungsmatrix der wichtigsten Architektur-Komponenten, im EA dargestellt

Die vielfältigen *Beziehungen* zwischen diesen Komponenten sind hier der Übersicht halber nicht dargestellt. Dazu gehören vor allem

- *horizontale* Verknüpfungen, z. B.:
    - die Schnittstellen zwischen Traffic Participants, Simulation Core und Observern,
      oder übergeordnet
    - die Schnittstellen zum Credible Simulation Process und dessen Artefakten wie Spezifikationen, Testbeschreibungen usw.

- *vertikale* Verknüpfungen, z. B. die Ableitung von Komponenten und deren Verknüpfungen aus der Architektur des *realen* Fahrzeugs, wie sie im Paper „Safety First for Automated Driving“ (SaFAD) bzw. dem ISO TR 4804 beschrieben ist, über alle Ebenen der Simulationsarchitektur hinweg bis zur konkreten Architektur eines Simulationslaufs.

Das AP 2.1 hat aus der *funktionalen* Ebene des Gesamtmodells einen Liefergegenstand „Funktionale Beschreibung“ ausgeleitet.

Geschäftsobjekte der Simulation
Die Geschäftsobjekte (Business Objects), auf die man im Kontext der Simulation regelmäßig trifft, lassen sich in folgendem UML-Diagramm übersichtlich kategorisieren (siehe Abbildung 107). Die als FMU darstellbaren Objekte sind hier rosa hinterlegt, alle anderen gelb.

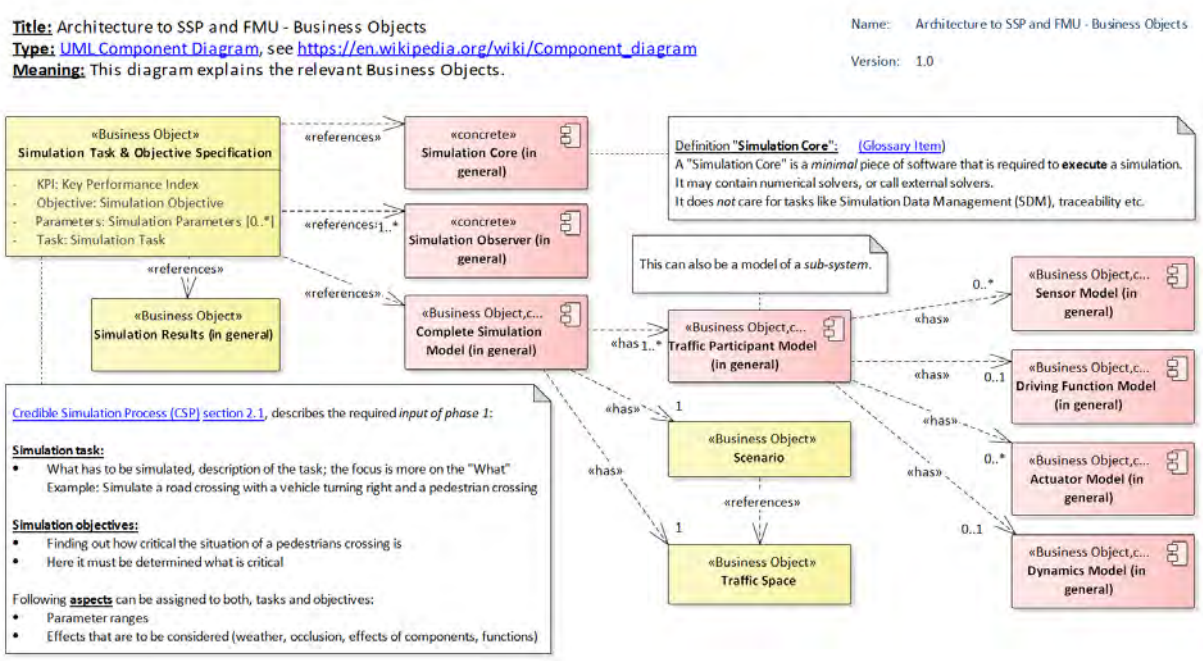


Abbildung 107: Geschäftsobjekte der Simulation

Unter den allgemeinen Simulationsergebnissen (Simulation Results in General) lassen sich *primäre*, *sekundäre* und Ergebnisse unterscheiden. Die folgende Abbildung 108 setzt sie in Verbindung mit Phasen des CSP:

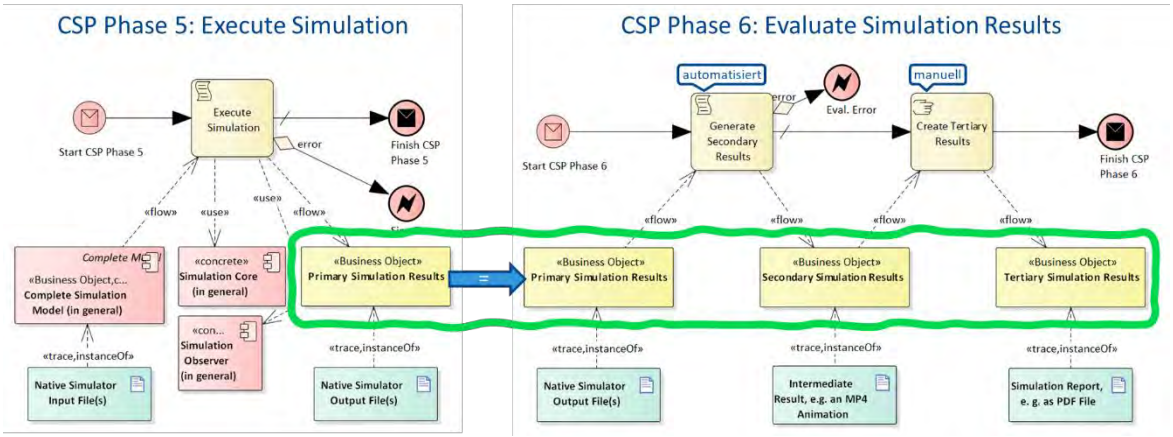


Abbildung 108: Primäre, sekundäre und tertiäre Simulationsergebnisse

Beschreibung der Methode am Beispiel einer Notbremsfunktion (AEB)

Die methodische Ableitung der in Abbildung 106 nicht dargestellten vertikalen Verknüpfungen zwischen den Komponenten auf den unterschiedlichen Architektur-Ebenen wurde an einem vereinfachten Prinzip-Beispiel anhand einer hochautomatischen Notbremsfunktion (AEB) praktisch evaluiert. Das folgende Diagramm zeigt die funktionale Architektur des *realen* Fahrzeugs gemäß SaFAD (Abbildung 105), in der die AEB-relevanten Komponenten blau markiert sind:

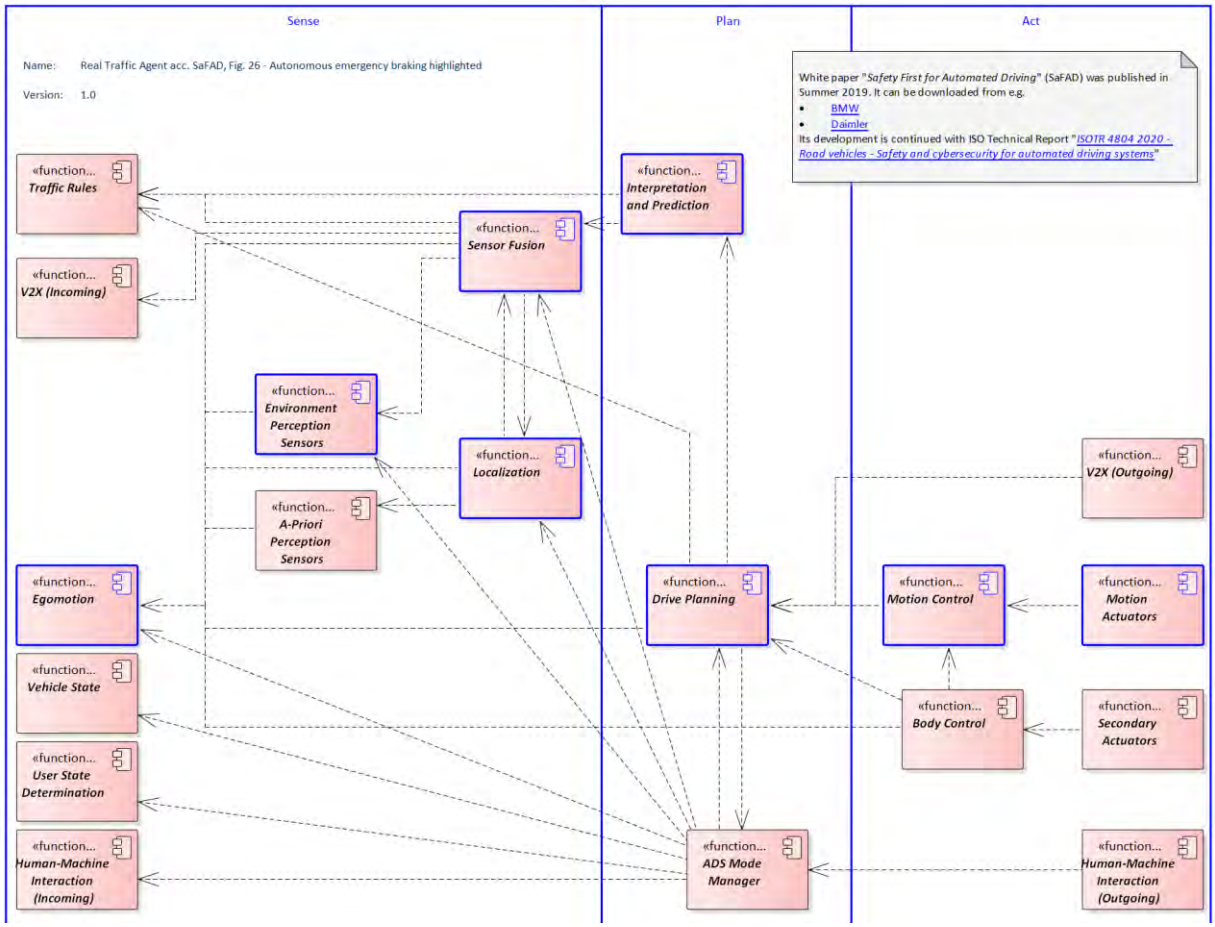


Abbildung 109: AEB-relevante Komponenten aus SaFAD

Daraus leitet sich die oberste Ebene des simulierten Fahrzeugs wie folgt ab:

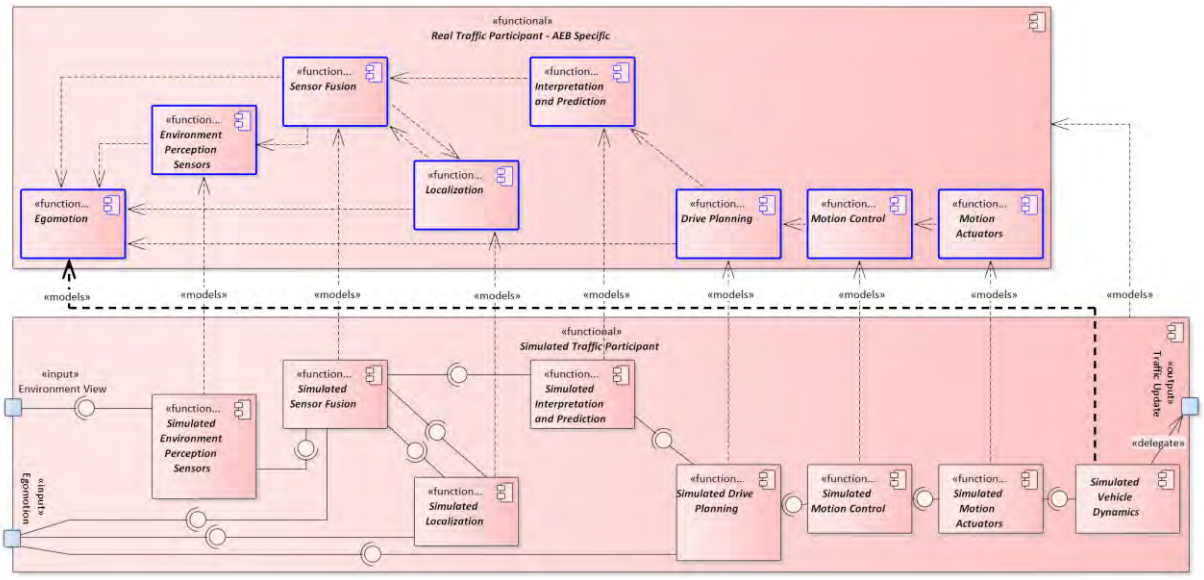


Abbildung 110: Oberste Ebene der AEB-Simulationsarchitektur

Die Abhängigkeiten (gestrichelte Pfeile in Abbildung 109) zwischen realen Komponenten werden in Abbildung 110 durch Assembly-Beziehungen zwischen simulierten Komponenten ersetzt. Simulierte Komponenten haben eine «models»-Abhängigkeit zu ihren realen Gegenstücken.
Die Komponente „Egomotion“ spielt in der SaFAD-Architektur eine besondere Rolle: Sie „beschreibt den aktuellen Zustand des Fahrzeugs in Form von Gierrate, Längsbeschleunigung, Querbeschleunigung und mehr. Weitere Werte, die den Fahrzustand beschreiben, können die Fahrzeuggeschwindigkeit oder der Schräglaufwinkel sein. Egomotion ist keine rein *interne* Komponente des Fahrzeugs, sondern eine Komponente, die das fahrdynamische Verhalten des Fahrzeugs *in seiner Umgebung* beschreibt. Die genannten physikalischen Messgrößen werden durch physikalische Naturgesetze bestimmt. In der Simulation ist die „Natur“ jedoch keine Instanz. Die Messgrößen können zu Beginn der Berechnung eines Zeitschrittes nicht als gegeben angenommen werden. Stattdessen müssen sie an dessen Ende explizit berechnet werden. Diese Funktion wird von der Komponente „Simulated Vehicle Dynamics“ bereitgestellt.

Der simulierte Verkehrsteilnehmer wird nun in den Kontext seiner Umgebung gesetzt. Auch dies folgt dem Vorbild der Realität, obwohl die Umgebung in SaFAD nicht untersucht wurde. Die Situation ist in der folgenden Abbildung dargestellt:

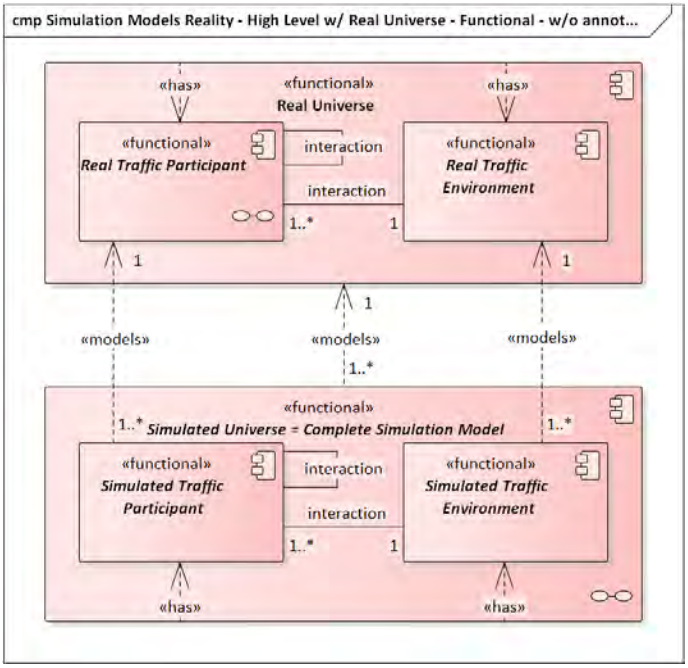


Abbildung 111: Reales und simuliertes Universum

Dementsprechend müssen die simulierten Sensoren mit Eingaben gefüttert werden, die aus der Ground Truth abgeleitet werden, aus anderen Verkehrsteilnehmern und der Umgebung. Die Ergebnisse der simulierten Fahrzeugdynamik müssen zurückgekoppelt werden, um die Ground Truth zu aktualisieren. In diesem Sinne unterscheidet sich das Ego-Fahrzeug nicht von anderen Verkehrsteilnehmern.
Es wird schnell klar, dass die wachsende Anzahl von Verkehrsteilnehmern, Elementen in der Verkehrsumgebung und deren komplexes Netzwerk von Interaktionen eine Orchestrierung erfordert. Außerdem müssen während und nach dem Simulationslauf Daten für die Nachbearbeitung extrahiert werden. Entsprechende Observer müssen mit den Simulationskomponenten verbunden werden. Dies ist der Punkt, an dem der Simulation Core ins Spiel kommt.

Insgesamt ergibt sich eine zweite Ebene der funktionalen Architektur der Simulation, die in der folgenden Abbildung des vollständigen Simulationssystems dargestellt ist:

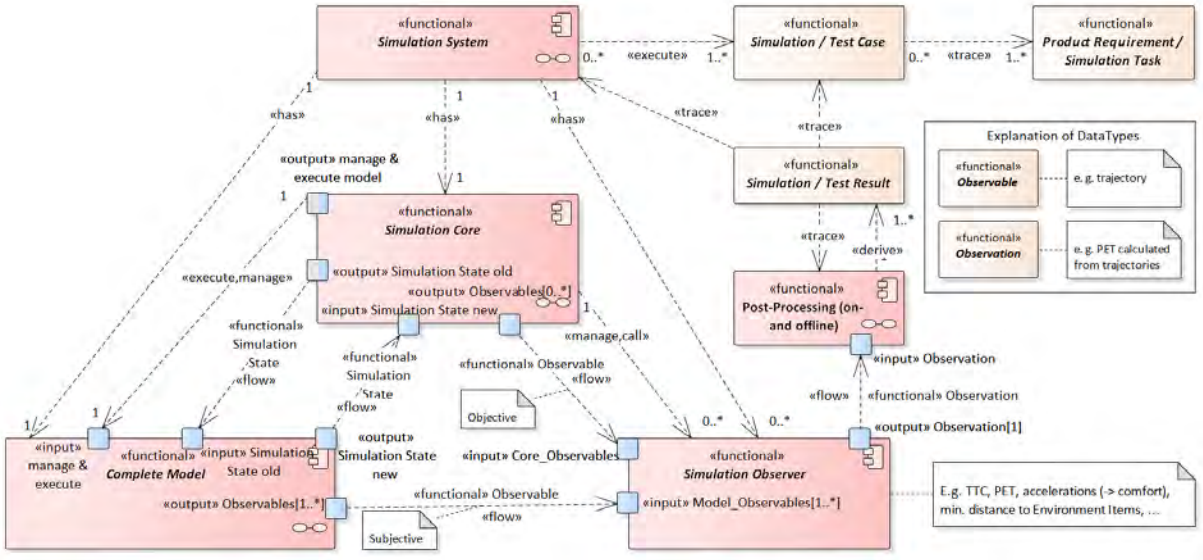


Abbildung 112: Funktionale Architektur des vollständigen Simulationssystems

Das Diagramm zeigt auch auf, wie das Simulationssystem mit dem Simulationsfall und dem Simulationsergebnis verknüpft ist.
Wesentlich für die Extraktion relevanter Daten aus einem Simulationslauf sind sog. *Observer*. Ein solcher *Observer* berechnet *Observations* aus *Observablen*. Eine *Observable* ist eine physikalische Größe, die (in der Realität) gemessen oder Variablen der Simulation entnommen werden kann, z. B. eine Trajektorie. Sie kann *objektiv* sein, d. h. der physikalischen Wahrheit entsprechen, oder *subjektiv*, d. h. der individuellen Interpretation der Realität durch einen Traffic Participant entsprechen. Eine *Observation* ist eine Größe, die aus einer oder mehreren Observablen *außerhalb* der eigentlichen Simulation *abgeleitet* wird, z. B. eine aus mehreren Trajektorien abgeleitete Time to Collision (TTC). Sie ist also in der Regel nicht direkt messbar.
In der *logischen* Architektur kann ein Observer fest im Core integriert sein, als FMU eingebunden werden oder separat nach Ende der Simulation aufgerufen werden.
Die interne Struktur des Gesamtmodells ist in der folgenden Abbildung 113 dargestellt. Ein wesentlicher Unterschied zu Abbildung 112 ist, dass es keine direkte Interaktion zwischen Verkehrsteilnehmern und Verkehrsumgebung mehr gibt. Stattdessen werden alle Verbindungen durch den Simulationskern geleitet. Da wir uns im Kontext einer dynamischen, in Zeitschritten organisierten Simulation befinden, erfüllt der Core mindestens folgende essentiellen Funktionenwährend der drei wesentlichen Phasen eines Simulationslaufs:

Initialisierung

- Laden, instanziieren und initialisieren derjenigen Komponenten des gesamten Simulationsmodells, die von Anfang an im Laufzeitsystem benötigt werden.
- Laden und initialisieren der Observer, sofern erforderlich.
- Herstellen aller Verbindungen zwischen Komponenteninstanzen und Observern, so dass später Informationen ausgetauscht werden können. (z. B. gem. einer SSP-Datei)

Nach der Initialisierung werden die Zeitschritte so lange wiederholt, bis ein Abbruchkriterium erfüllt wird.

Ausführung eines Zeitschrittes

- Einfügen neuer Traffic Participants in die Szene des anstehenden Zeitschritts, falls erforderlich. Dies umfasst die Initialisierung und Verbindung ihrer Komponenten.
- Versorgung der Modellkomponenten mit dem aktuellen Zustand der Simulation.
- Aufruf aller Modellkomponenten zur Berechnung ihrer Ergebnisse.
- Einsammeln aller dieser Einzelergebnisse.
- Aktualisierung und Konsolidierung eines konsistenten Gesamtzustands der Simulation entsprechend der Einzelergebnisse.
- Versorgung der Observer mit dem neuen Simulationszustand. Dies umfasst auch das Schreiben jeglicher Art von Ergebnis- oder Protokolldateien.
- Auswertung des Abbruchkriteriums und Beendigung der Simulation, falls erforderlich.
- Löschen von Traffic Participants aus der Szene, falls sie nicht mehr benötigt werden. Dies umfasst die Auflösung von Verbindungen und das Löschen von Komponenten.
- Bereitstellung des neuen Gesamtzustands der Simulation für die Neuverteilung im nächsten Zeitschritt.

Um Rechenzeit zu sparen, ist es üblich, an einzelne Komponenten nur den für sie jeweils relevanten Ausschnitt des Gesamtzustands weiterzuleiten.

Shut down

- Finalisierung und Schließen aller Arten von Ergebnis- und Protokolldateien.
- Auflösung aller Verbindungen und Löschen aller Komponenten aus dem Laufzeitsystem.

Diese Ebenen der funktionalen Architektur sind in der folgenden Abbildung dargestellt:

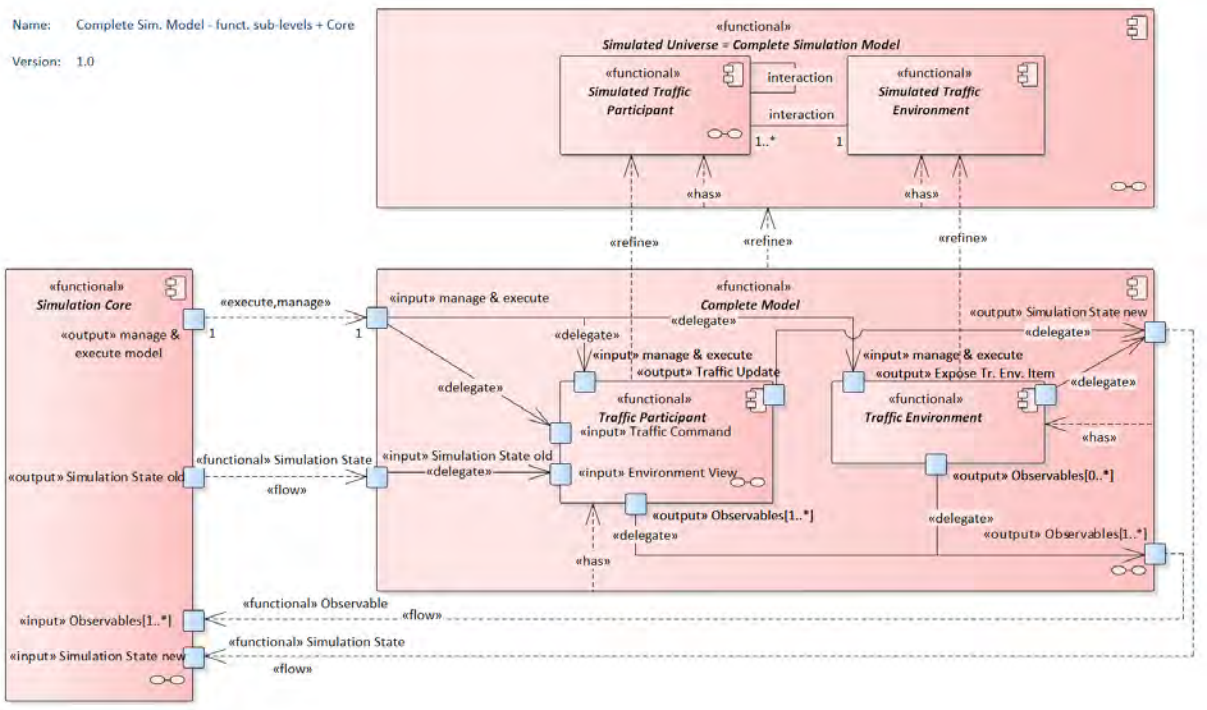


Abbildung 113: Beide Teil-Ebenen der funktionalen Architektur samt Simulation Core

Die obere Ebene in der Abbildung entspricht der unteren Ebene aus Abbildung 111.

Auf der zweiten Ebene der funktionalen Architektur erhält der SaFAD-konforme *Traffic Participant* modifizierte Schnittstellen an seinen Systemgrenzen (siehe Abbildung 114).

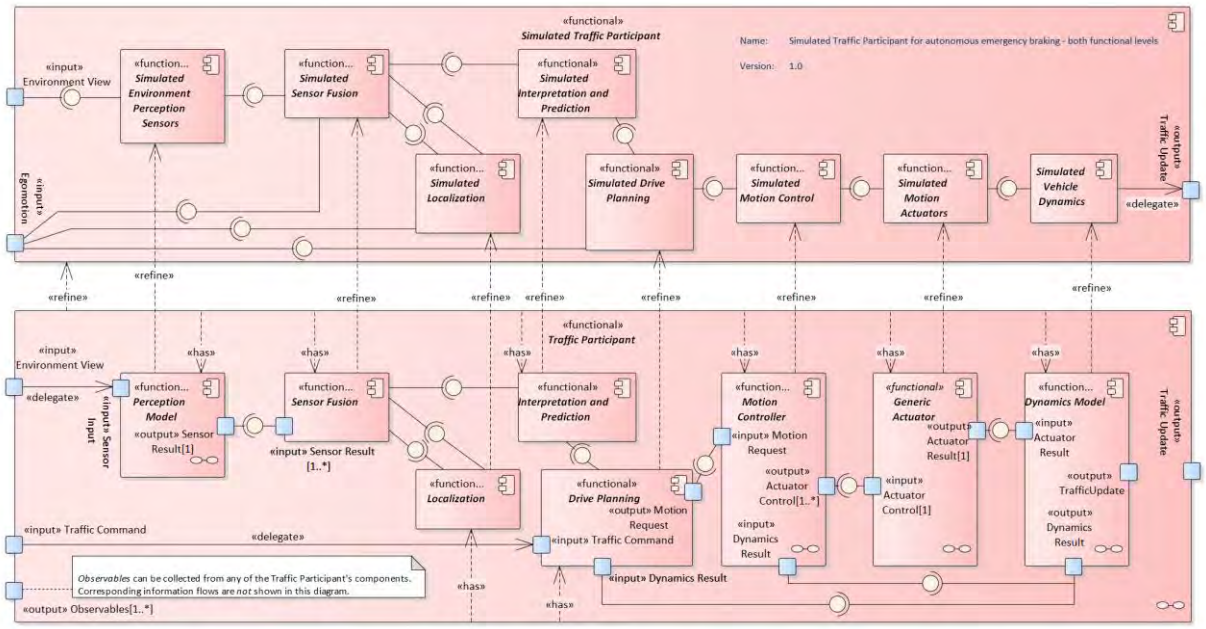


Abbildung 114: Beide Teil-Ebenen der funktionalen Architektur des Traffic Participants

Übergang zur *logischen* Architektur

Der *funktionalen* ist die *logische* Ebene der Architektur nachgelagert. Dort wird zunächst festgelegt, welche Funktionen von welchen logischen Komponenten der Simulation implementiert werden sollen. Beim Traffic Participant werden hier z. B. die vier Kernfunktionen *Sensor Fusion*, *Localization*, *Interpretation and Prediction* und *Drive Planning* zur Komponente *Driving*

Function zusammengefasst. Der Zusatz *solution strategy* unterscheidet die logische Sub-Ebene der Lösungsstrategie von der nachfolgenden Sub-Ebene der technischen Umsetzung:

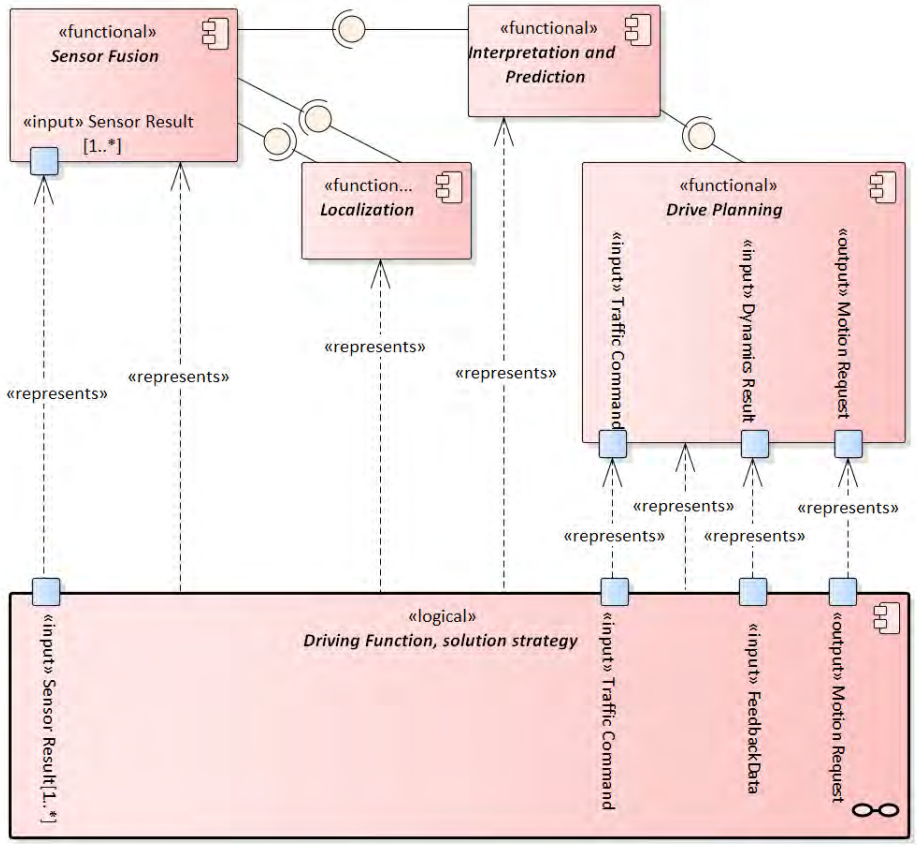


Abbildung 115: Aggregation von vier Funktionen zu einer logischen Komponente

Auf der logischen Sub-Ebene der technischen Umsetzung erfolgt dann die Festlegung auf zu verwendende Basis-Technologien, in unserem Fall FMI, SSP, OSI, OpenDRIVE und OpenSCENARIO.

Die Realität bietet eine unüberschaubare Vielfalt von Traffic Participants. Dazugehören nicht nur selbstfahrende PKW. Das Spektrum beginnt bei Tieren, die vor das Fahrzeug fliegen oder laufen, Bällen, die auf die Straße springen, Kindern, die womöglich hinterherrennen und reicht bis hin zu Schwerlasttransportern, Entsorgungsfahrzeugen, Bau-, Land- und Sondermaschinen. Es ist daher unbezahlbar bis unmöglich, all diese Traffic Participants vorab einzeln im Detail zu modellieren. Stattdessen hat das Projekt eine Methode aufgezeigt, wie sich die jeweils tatsächlich benötigten Traffic Participants effizient aus Einzelkomponenten *konfigurieren* lassen. Das geschieht in etwa so, wie auch ein Fahrzeughersteller entsprechend der jeweiligen Kundenwünsche ein individuelles Produkt herstellt, das oft vorher noch nicht in genau dieser Konfiguration produziert wurde. In beiden Fällen sind exakt definierte und leistungsfähige Schnittstellen die Voraussetzung für den Erfolg.

Diese Konfiguration erfolgt in mehreren Konfigurationsschritten gemäß folgender Abbildung:

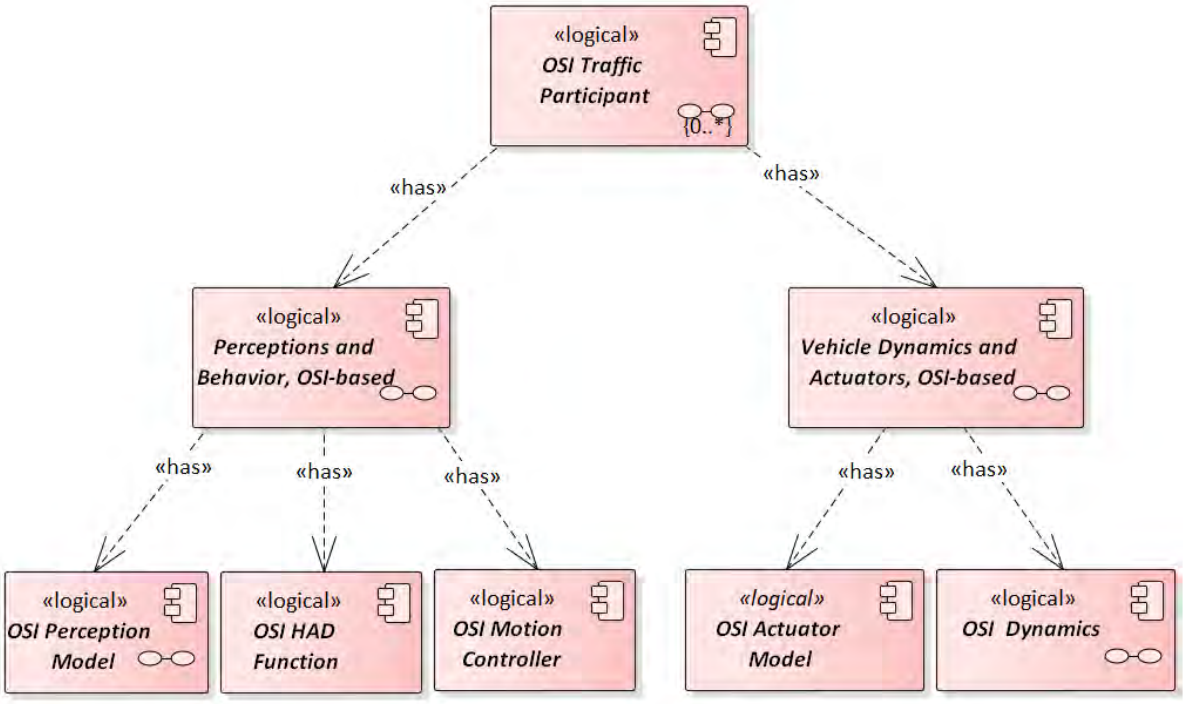


Abbildung 116: Konfigurationsgraph für den OSI-konformen Traffic Participant

Daher sieht der OSI-konforme Traffic Participant auf dem obersten dieser Konfigurations-schritte wie folgt aus:

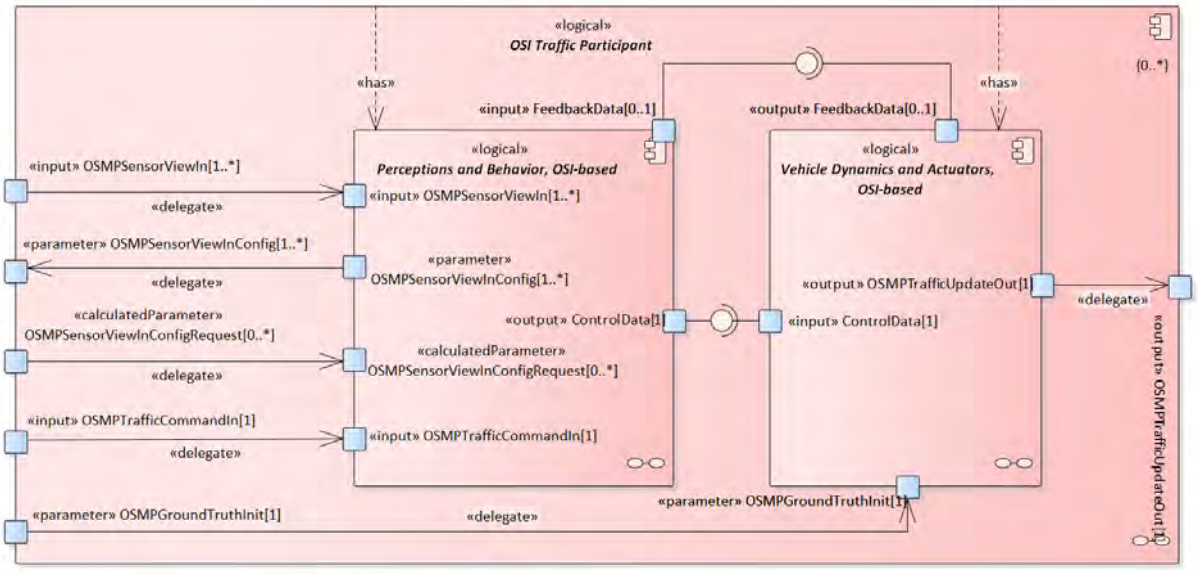


Abbildung 117: Konfigurierbarer OSI-konformer Traffic Participant

Für Perception and Behavior setzt sich das Schema wie folgt fort:

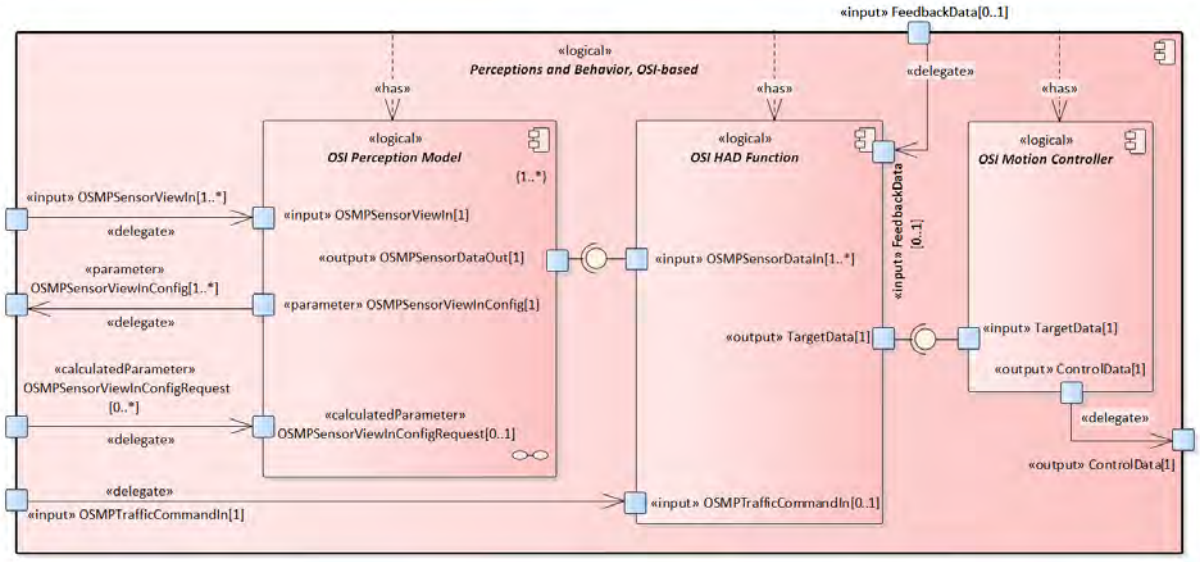


Abbildung 118: Konfigurierbare OSI-konforme Komponente Perceptions and Behavior

Diese Vorgehensweise ermöglicht zum einen die Instanziierung eines einfachen Modells des „Umgebungsverkehrs“ mit einem sehr rudimentären monolithischen Verhaltens- und Dynamikmodell und zum anderen die Instanziierung komplexerer Modelle durch Hinzufügen weiterer detaillierterer Komponenten.
Auf der nachfolgenden konkreten Ebene der Architektur kann der Verkehrsteilnehmer entweder als „Black Box“ mit nur den äußeren Schnittstellen aus Abbildung 118 oder als „Grey Box“ mit den internen Komponenten daraus instanziiert werden. Die Komponenten *Perception and Behavior* und *Vehicle Dynamics and Actuators* können dann wiederum entweder als „Black Box“ oder „Gray Box“ instanziiert werden. In diesem Fall zerfällt Komponente *Perception and Behavior* in *Perception*, *HAD Function* und *Motion Controller*.
Aus dieser einen Blaupause können nun die konkreten Architekturen aller Traffic Participants der Simulationsaufgabe erstellt werden. Die dafür benötigten Teilmodelle können in einer Modellbibliothek vorgehalten werden. Durch die modellierten Beziehungen bleiben Traceability und Konsistenz jederzeit erhalten.

Fazit
Im Verlauf der beispielhaften Durchführung der Methode wurden die drei Hauptebenen (funktional, logisch & physikalisch) der Architektur bedarfsorientiert weiter ausdifferenziert. Repräsentationen der Geschäftsobjekte in Datenobjekten bzw. Dateien spezifischer Formate

wurden beschrieben. Dies wird im folgenden Diagramm exemplarisch anhand der *logischen Architektur* dargestellt:

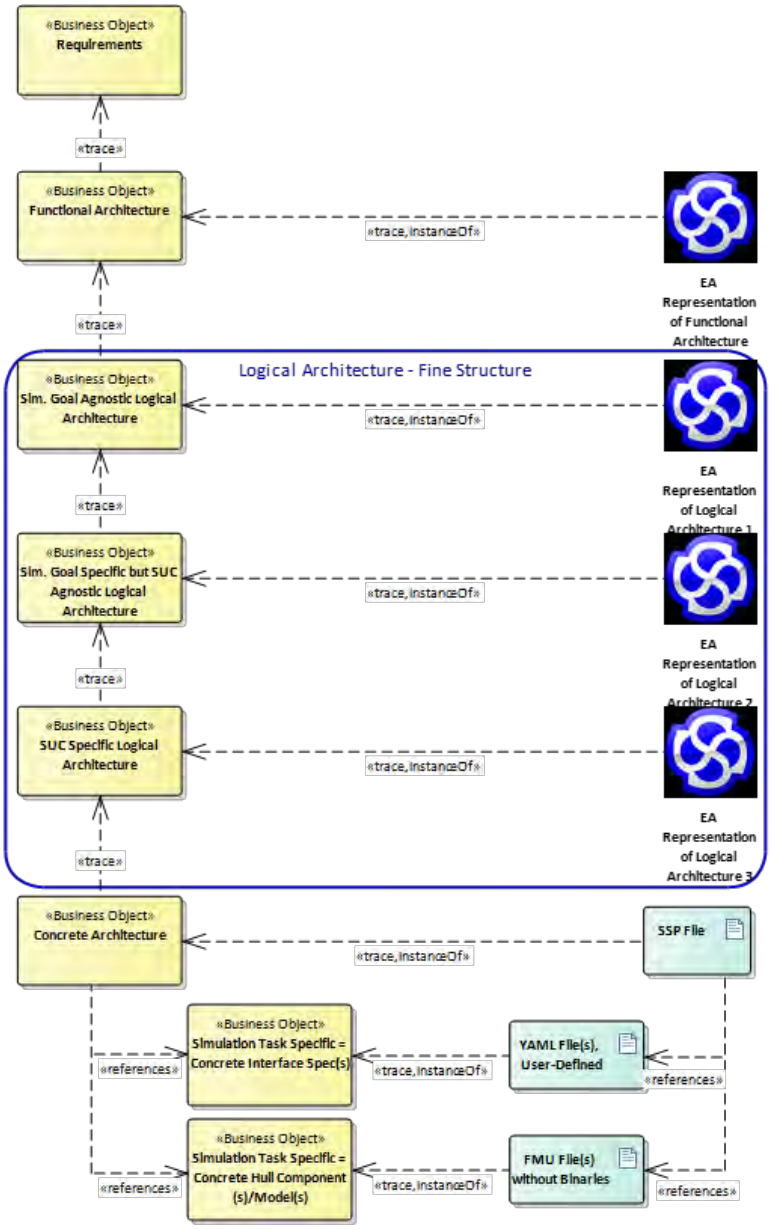


Abbildung 119: Ausdifferenzierung der logischen Architektur

Die rechte Spalte der Abbildung 119 stellt die Datenformate dar, die sich für die Arbeitsergebnisse (Artefakte) in der linken Spalte verwenden lassen.
Abschließend soll noch der Faden von weiter oben wieder aufgriffen und die funktionale Architektur eines realen Fahrzeugs gemäß des SaFAD-Papers (siehe Abbildung 105) im UML-Überblick der integrativen Architektur der Simulation verortet werden. Die nachfolgende Abbildung enthält dazu die Spalten der Orientierungsmatrix (siehe Abbildung 106) aus technischen Gründen in einer permutierten Reihenfolge. Die SaFAD-Architektur findet sich dabei am rechten Rand der zweiten Zeile von oben:

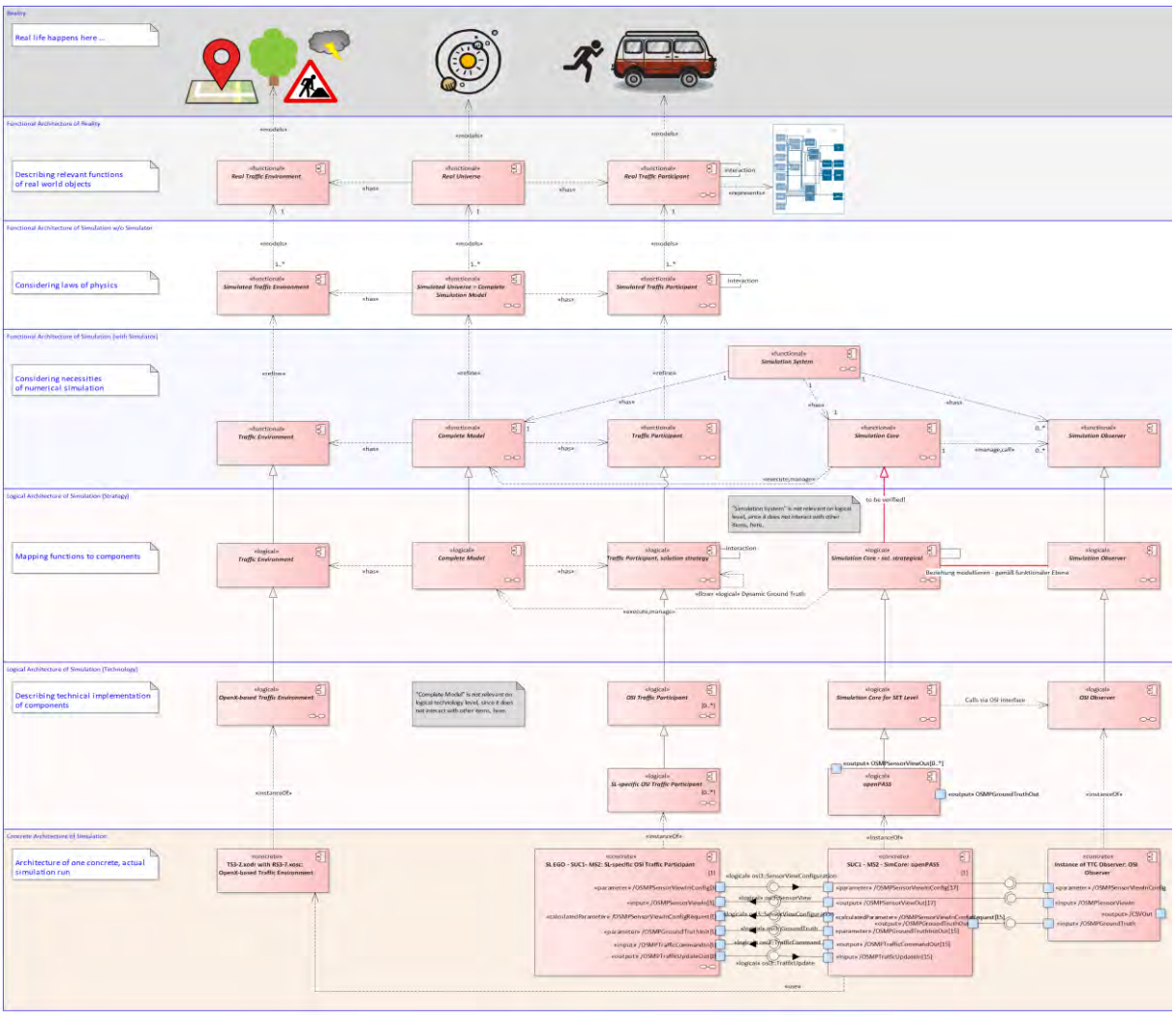


Abbildung 120: SaFAD-Paper im UML-Überblick der integrativen Architektur der Simulation

Diese Abbildung zeigt darüber hinaus die lückenlose Verknüpfung von Entitäten der realen Welt bis hinunter zu konkreten Simulationsmodellen. Alle Traceability-relevanten Entscheidungen lassen sich dabei einer spezifischen Architekturebene zuweisen, dort dokumentieren und in größtmöglichem Maße wiederverwenden. SSP-Dateien werden direkt von der untersten Ebene ausgeleitet.
Darüber hinaus wird klar ersichtlich, dass *Produkt*architektur stets einen zwar wichtigen, aber nur vergleichsweise kleinen Teil der *Simulations*architektur ausmacht.

CSP-Einbindung der Methode der integrativen Architektur der Simulation

Glaubwürdigkeit (Credibility) ist wichtig für den Einsatz von Simulation zur Produktfreigabe. Deshalb hat AP 3.1 der Weiterentwicklung des Credible Simulation Process (CSP) große Aufmerksamkeit gewidmet.
Architekturergebnisse stellen einen wichtigen Bestandteil diverser Spezifikationen dar. Gemäß CSP müssen alle Artefakte, insbesondere Spezifikationen, auf glaubwürdige und nachvollziehbare Weise auseinander hervorgehen. Dementsprechend gilt dies erst recht für Architekturergebnisse.
Architekturergebnisse werden nicht in der Art „implementiert“ wie etwa Simulationsmodelle. Daher sind für sie nur die Phasen 1 bis 3 des CSP relevant.
Die Architekturergebnisse auf der *konkreten* RFLP-Ebene hängen von denen der *logischen* Ebene, und die wieder von den Ergebnissen der *funktionalen* Ebene ab.

Die typischen Zykluszeiten von Architektur-Änderungen unterscheiden sich erheblich zwischen diesen Ebenen:

- funktional Jahre
- logisch Monate
- konkret Tage,
  bei bloßer Parameteränderung ggf. auch nur Minuten.
  (Einbauposition der Kamera, Geschwindigkeit des Radfahrers …)

Deshalb ist es vernünftig, pro Architekturebene eine eigene Instanz des CSP vorzusehen. Die nachfolgende Abbildung 121 stellt grafisch dar, wie die Ergebnisse dieser Instanzen in nachgeordnete Instanzen und letztlich in den CSP-konformen Simulationsprozess eingehen:

- *Funktionale* und *logische* Architektur gehen in die *Requirements* Spec. des CSP ein.
- Die *konkrete* Architektur geht in die *Design* Specification des CSP ein.

Abbildung 122 erweitert die Darstellung auf Details zu den CSP-Phasen 4 bis 6.

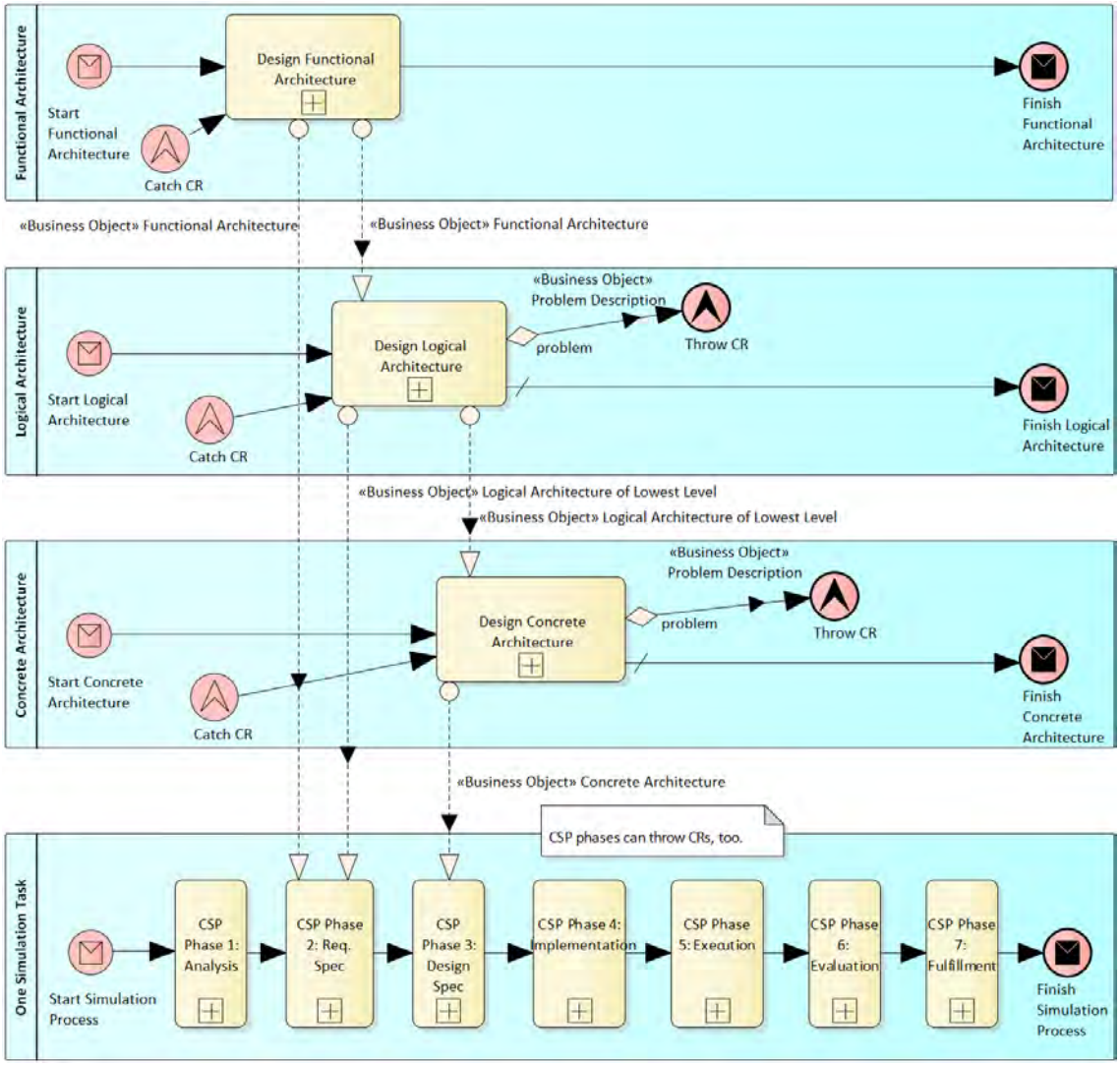


Abbildung 121: Zusammenspiel integrative Architektur und CSP

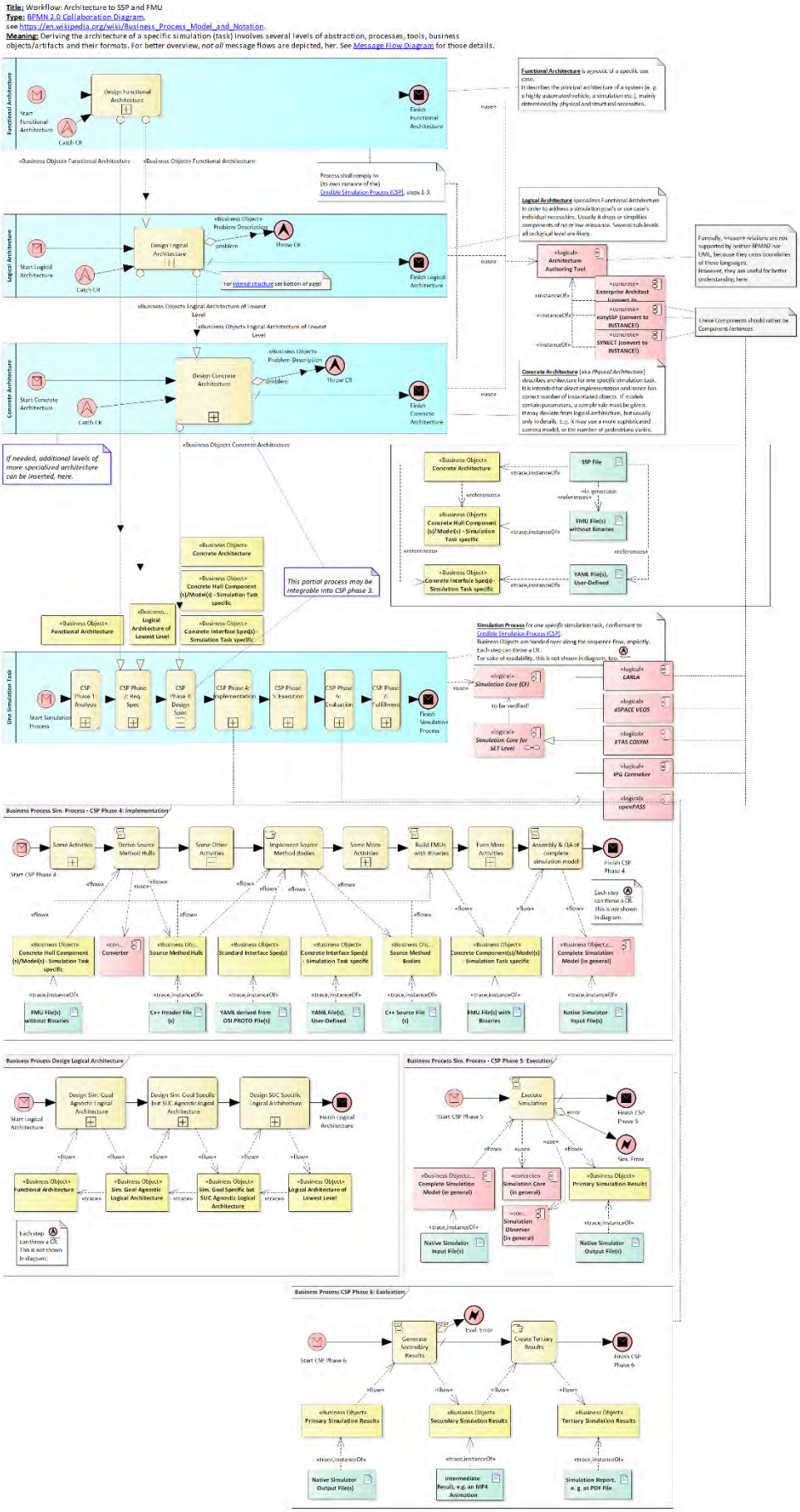


Abbildung 122: CSP-konforme Ableitung konkreter Architekturen aus abstrakteren Architekturen

Anwendungsfälle (Use Cases) und Motivation für die Architekturarbeit

Selbstverständlich muss auch für Architekturarbeiten eine Kosten-Nutzen-Betrachtung erfolgen. Hierbei müssen wichtige Anwendungsfälle und Motivationen identifiziert werden. Einige davon wurden bereits in der Einleitung zu Kapitel 2.3.3.1.3 aufgeführt:

- Synergien nutzen.
- Orientierung bieten.
- Kommunikation erleichtern und Missverständnissen vorbeugen.
- Redundante Arbeiten sowie Fehlerquellen vermeiden.
- Direkte Unterstützung des Modellaufbaus in einer Simulationssoftware.
- Nachweis der Glaubwürdigkeit.
- Unterstützung der Planung und Koordination der Entwicklungsarbeiten.

Hinzu kommt der Aspekt der *Kostenoptimierung von Software-Lizenzen bis hin zur Qualifizierung eingesetzer Toolketten*. Hierzu wurde bereits früh im Projekt mit Fokus auf die Tool-Architektur die Frage identifiziert, wie weit sich eine SUC-*übergreifende, einheitliche Tool-Landschaft* definieren lässt. Die Frage erweiterte sich dann auch auf die Landschaften der Modelle und der Test- bzw. Simulationsfälle.
Je mehr dieser Architektur allgemein verwendbar ist, umso *ökonomischer* lassen sich Softwarelizenzen einsetzen, umso *weniger Overhead* entsteht durch zusätzliche Schnittstellen, und umso *kostengünstiger* fällt eine Qualifizierung einer Tool-Architektur für Produktfreigaben aus.
Die Ausrichtung von Arbeitsergebnissen auf der richtigen Architekturebene fördert deren Wiederverwendbarkeit und trägt so ebenfalls zur Kostenminimierung bei.

Architektur-gebundener Einsatz von SSP: Motivation und Umsetzung

Der Standard *Functional Mockup Interface (FMI)* hat sich bereits über viele Jahre weiterentwickelt und bewährt. Er dient dazu, die Schnittstelle eines Verhaltensmodells zu beschreiben. Den *Zusammenbau*, insbesondere die *Verschaltung* solcher Modelle zu größeren Einheiten kann er jedoch nicht beschreiben. Diese Lücke schließt der Standard *System Structure and Parameterization (SSP)* erst seit März 2019, also im Wesentlichen zeitgleich zum Beginn des Projektes SET Level.
Daher war es ein Ziel des Projektes, diesen jungen Standard zu evaluieren.
Dementsprechend wurde von AP 4.1 gemeinsam mit AP 2.1 ein „Proof of Concept“ (PoC) entwickelt zur Darstellung eines durchgängigen, toolgestützten, CSP-konformen Prozesses von den höchsten, auf funktionaler Ebene abstrahierten Modellarchitekturen bis hin zu Source-Code-Rümpfen für z. B. Sensor-Modelle und deren Verschaltung mittels SSP.
Die theoretische Vorgehensweise dabei von der funktionalen bis zur konkreten Architekturebene wurde bereits in den vorangehenden Abschnitten beschrieben. Die *organisatorische*

Umsetzung des Ansatzes in der Praxis unter Einsatz einer Modell-Bibliothek ist in folgendem Diagramm dargestellt:

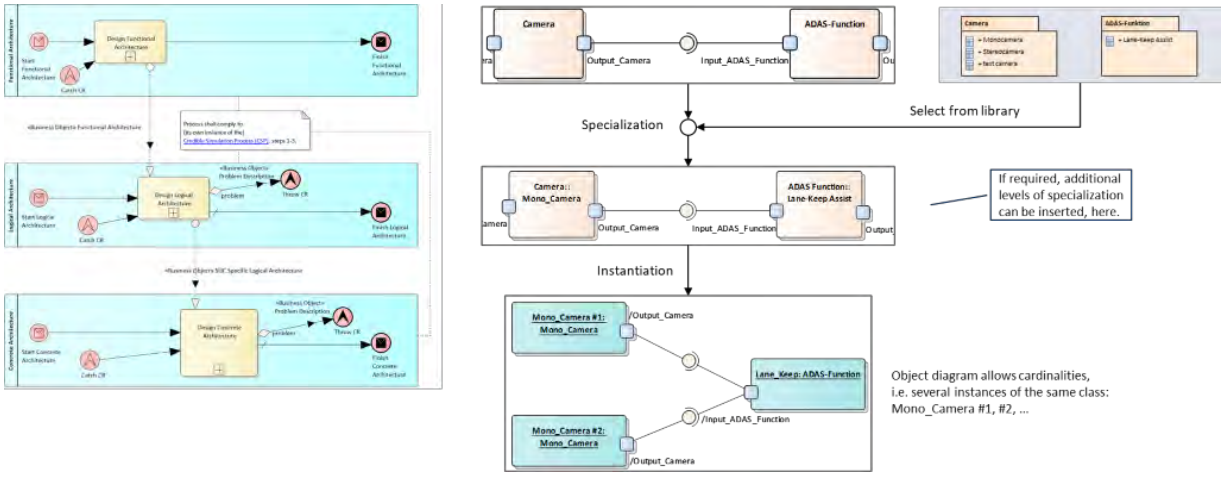


Abbildung 123: Organisatorische Umsetzung der 3-Ebenen-Architektur

Darum wird im Folgenden die Weiterverarbeitung der aus dem Enterprise Architekt im XMI-Format ausgeleiteten konkreten Architekturen beschrieben.
ZF hat ein Werkzeug beigesteuert, das diese XMI-Dateien zur Weiterverarbeitung in anderen Simulationswerkzeugen ins Format SSP konvertieren kann. Dort kann dann z. B. nach folgendem Schema weitergearbeitet werden:

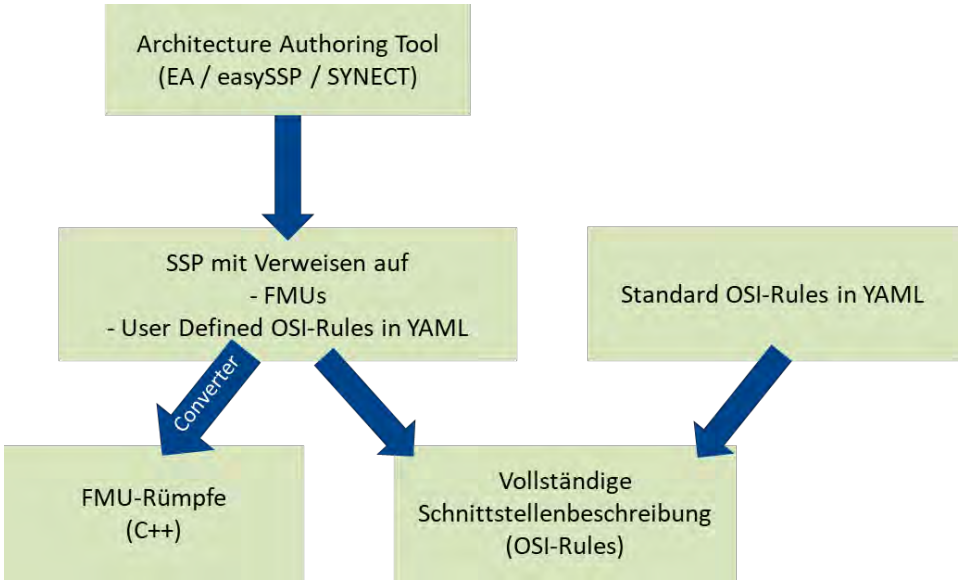


Abbildung 124: Ableitung von Modell-Rümpfen aus SSP-Dateien

Der entsprechende Prozess wurde an zunehmend komplexen Beispielen validiert (siehe Abbildung 121 und Abbildung 122).

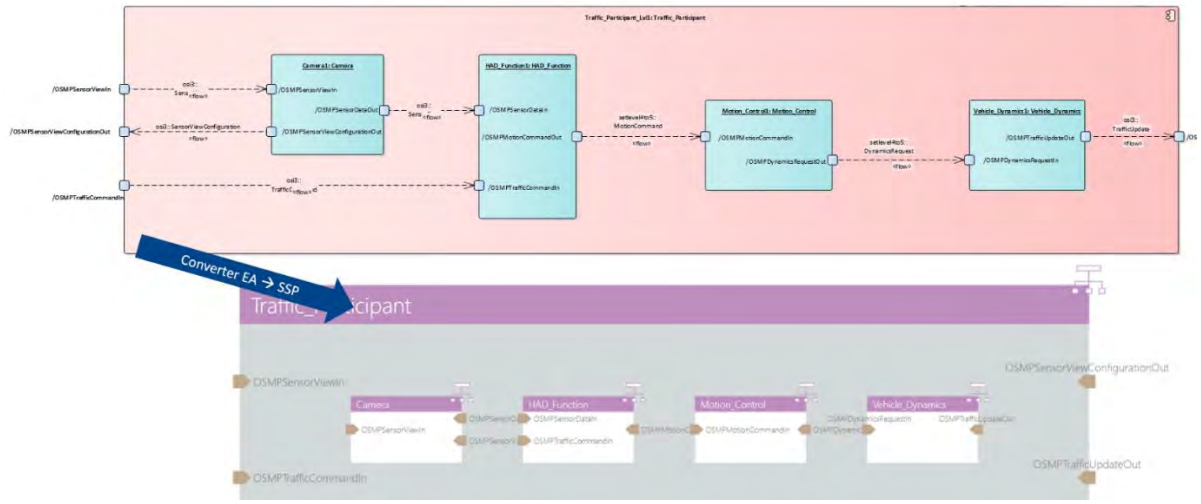


Abbildung 125: Erster Versuch zur Übertragung von Architektur nach SSP und weiter in andere Tools

Eine Validierung anhand eines komplexeren Beispiels folgte nach:

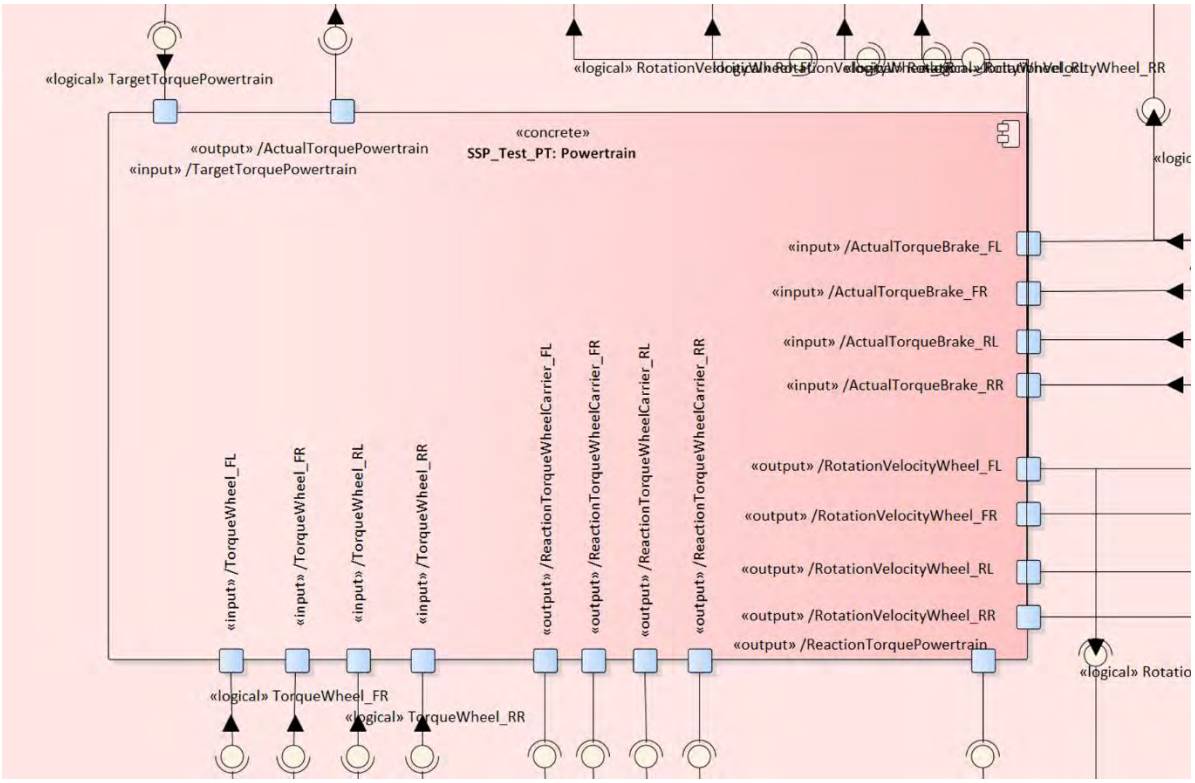


Abbildung 126: Darstellung des Powertrain-Blocks aus „Vehicledynamic and Actuators Model Detail 2“

Daraus wurde eine SSP-Datei erstellt und erfolgreich ins Tool easySSP importiert, wie in Abbildung 127 zusehen:

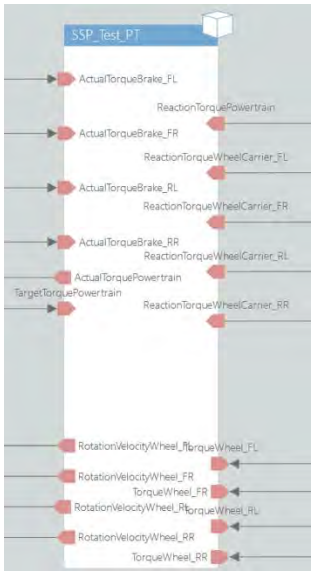


Abbildung 127: Graphische Darstellung des SSP zum oben abgebildeten Powertrain-Block

Zum Meilenstein 3 schließlich wurde gemeinsam mit AP 4.2 die konkrete Architektur eines kompletten Simulationslaufs von SUC 1 betrachtet. Die folgende Abbildung stellt sie auf höchster Ebene (Simulatorkern (sog. Core) mit Verkehrsteilnehmern) dar:

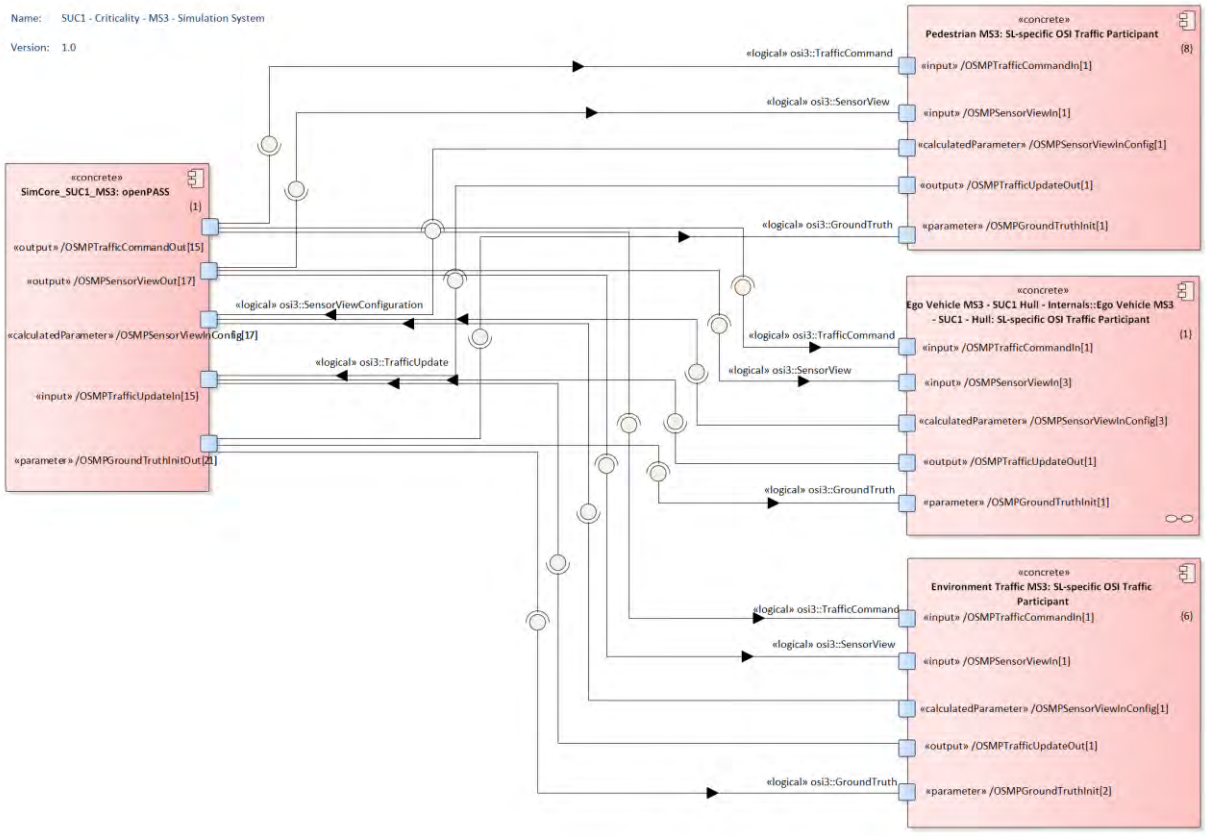


Abbildung 128: Konkrete SUC 1-MS 3 Architektur, Gesamtsicht

Die inneren Strukturen der Verkehrsteilnehmer und des Cores wurden in weiteren Diagrammen aufgelöst. Für das *Ego Vehicle* sieht dies wie folgt aus:

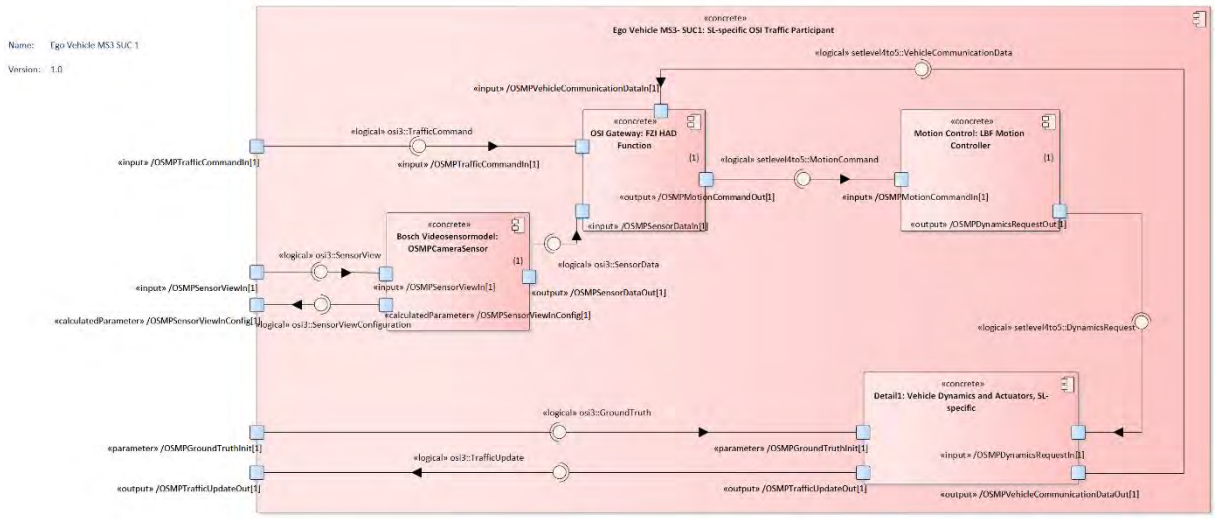


Abbildung 129: SUC 1-MS 3 Architektur, Sicht aufs Ego Vehicle

Dieses Diagramm wurde via SSP nach easySSP übertragen und dort korrekt interpretiert:

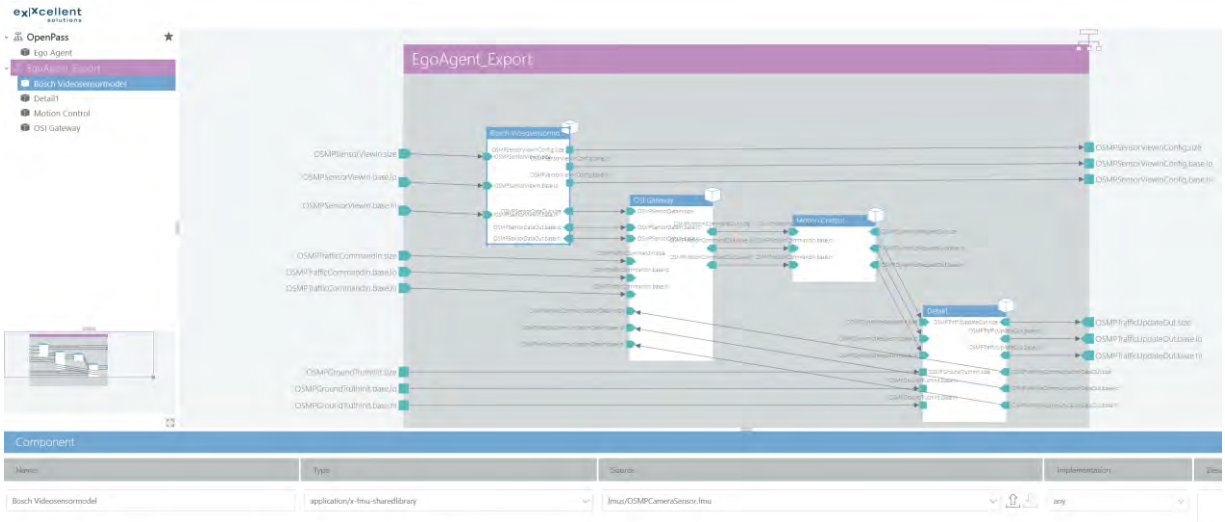


Abbildung 130: SUC 1-MS 3 Architektur, Sicht aufs Ego Vehicle, in easySSP importiert

In diesem Kontext wurde auch an der Weiterentwicklung und Validierung der SSP-Unterstützung von openPASS gearbeitet und zum MS 3 ein deutlich reiferer Stand erreicht.

2.3.3.1.4 Metadaten

Bedeutung von Metadaten

In Produktentstehungsprozessen ist es zumeist *nicht* ausreichend, nur ein bestimmtes Arbeitsergebnis (Spezifikation, Konstruktion, Prototyp, Validierungsergebnis, usw.) zu erzeugen. Jedes Ergebnis muss auch mit Metadaten ausgestattet werden. Selbst wenn nur eine einzelne Person beteiligt wäre, würde sie nach einiger Zeit nicht mehr in der Lage sein, ein bestimmtes Ergebnis zügig aufzufinden oder über dessen Bedeutung und Status Auskunft zu geben. Umso diffiziler ist die Situation, wenn mehrere Personen zusammenarbeiten.

Diese Situation ist selbstverständlich auch im Bereich der Simulation automatisierter Fahrfunktionen sehr ausgeprägt. Jedes Objekt (Arbeitsergebnisse, aber auch Simulationswerkzeuge, Prozessschritte usw.) muss mit den richtigen *Metadaten* ausgestattet sein. Im Rahmen von SET Level hat AP 4.1 diese Thematik insbesondere im Hinblick auf folgende *Anwendungsgebiete* untersucht:

- Traceability / Verwendungsnachweis für Ergebnisse (sog. „Vorwärtssuche“)
- Verwendete Ressourcen und Eingabedaten („Rückwärtssuche“)
- Überführung der Auswahl in ein ausführbares Modell („Technik“)
- Automatisierbarer Aufbau Gesamtmodell („Delivery”)
- Austausch über Organisationsgrenzen hinweg
- Spezifisch für Verkehrsräumen und Szenarien:
  Schnelle Identifikation von zueinander passenden Verkehrsräumen und Szenarien

Besondere Aufmerksamkeit wurde der Identifikation von Metadaten für Verkehrsräume und Szenarien gewidmet.

Arten von Metadaten im Allgemeinen

Als Stand der Technik kann von einer Klassifikation der Metadaten wie im folgenden Kasten ausgegangen werden. Die SET Level-spezifische Umsetzung ist darin ebenfalls mit angegeben:

Descriptive metadata, e.g.:

- model creator’s contact address of an artifact
- the context, in which an artifact can be used
- the parameter ranges for which an artifact is validated (if any)
- the model quality of an artifact
- other keywords for information retrieval

Descriptive metadata are typically re-presentable as key-value-pairs.

Structural metadata, e.g.:

- location of an artifact within a hierarchy
  - In SET Level, this is covered by position in GitLab’s group/project hierarchy.
- location of a simulation model within a hierarchical simulation model structure
  - In SET Level, this is covered by several means, such as <Catalog/> elements, SSP files, simulator configurations etc.
- “horizontal” relations between artifacts,
  e.g. SUC → scenario → traffic space → specification
  - In SET Level, for the relation scenario → traffic space, this is covered by position in GitLab’s group/project hierarchy.
    However, for (future) exchange e.g. in OEM-TIER1-situation, an *explicit* representation in SRMD will be mandatory.

Structural metadata are typically re-presentable as relations.

Administrative metadata, e.g.:

- time stamps of creation, last modification

- In SET Level, this is covered by Git functionality.

- required privileges / information for access control
  - In SET Level, this is covered by GitLab functionality.
- artifact type, e.g. requirement, specification, traffic space, scenario, …
- release status of an artifact, e.g. draft, ready for test, ready for productive use, withdrawn, …
  - In SET Level, this is can be achieved using Git tagging functionality.

Administrative metadata usually is mandatory and allows for explicitly enumerated values, only.

Dementsprechend hat AP 4.1 Metadaten für Verkehrsräume und Szenarien gruppiert.

Kooperation mit TP 3

AP 4.1 hat die Ergebnisse von TP 3 im Hinblick auf Metadaten von Test- bzw. Simulationsfällen, FMU-Modelle und Simulationswerkzeugen reviewed und bei der Formulierung von Use Cases unterstützt.

Metadaten für Verkehrsräume und Szenarien

Gemeinsam mit TP 1 hat AP 4.1 herausgearbeitet, welche Metadaten für Verkehrsräumen und Szenarien relevant sind. Es wurde dabei von den weiter oben aufgeführten Anwendungsgebieten ausgegangen und es wurden die identifizierten Metadaten klassifiziert. Die identifizierten Metadaten wurden darüber hinaus danach strukturiert, ob sie

- für Verkehrsräume, (23 Stück)
- für Szenarien, (24 Stück)
- oder für beide Arten von Artefakten (36 Stück)

anwendbar sind.
Die Ergebnisse wurden in https://gitlab.setlevel.de/open/processes_and_traceability/traceability_data/-/tree/main/metadata_for_trafficspaces_and_scenarios veröffentlicht. Die unter https://gitlab.setlevel.de/open/traffic-spaces-and-scenarios/tss veröffentlichten Verkehrsräume und Szenarien wurden mit beispielhaften Metadaten in Form von SRMD-Dateien ausgestattet.
Zur Simulation müssen Verkehrsräume und Szenarien die Komplexität und Vielfaltigkeit der realen Welt wiedergeben können. Eine vollständige Verschlagwortung der realen Welt ist aber nicht möglich, schon gar nicht im Rahmen der begrenzten Ressourcen von SET Level. Trotzdem müssen Verkehrsräume und Szenarien möglichst prägnant und effizient bedatet werden. Deshalb wurden folgende Standards und Projektergebnisse Dritter analysiert und in den Arbeiten des Projekts SET Level berücksichtigt:

- ASAM OpenLabel
- ASAM OpenODD
- ASAM OpenSCENARIO 1.0
- ASAM OpenSCENARIO 2.0
- ASAM OpenDRIVE

- BSI PAS 1883
- OpenStreetMap
- CityGML

Die eigenen, im Projekt identifizierten Bedarfe an Metadaten wurden mit diesen Quellen abgeglichen und Möglichkeiten zum Re-Use externer Klassifikationsschemata beschrieben. Die meisten dieser Quellen befinden sich gegenwärtig in der Entwicklung – dem Projekt standen aber viele vorläufige Ergebnisse zur Verfügung. Auch aus diesem Grund ist ein flexibler Ansatz notwendig. Entsprechend wurde eine Methode entwickelt, externe Attributierungsschemata mit minimalem Aufwand für SRMD zu erschließen.

SRMD-Datenformat

Als Syntax zur Beschreibung der Metadaten eines Verkehrsraums oder Szenarios wurde das auch an anderen Stellen im Projekt verwendete Format SRMD festgelegt. Damit wurden insbesondere in TP 3 bereits zuvor im Bereich der Modelle für Fahrzeugkomponenten positive Erfahrungen gesammelt.
Die allgemeine Beschreibung eines Paares aus KEY und VALUE hat demnach in SRMD folgende Gestalt:

```
<?xml version="1.0" encoding="UTF-8"?>
<srmd:SimulationResourceMetaData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://apps.pmsf.net/STMD/SimulationResourceMetaData SRMD.xsd"
    xmlns:srmd="http://apps.pmsf.net/STMD/SimulationResourceMetaData"
    version="0.5" name="Traffic Space XXX Meta-Data">
    <srmd:Classification type="de.sl4to5.srmd.model-meta-data">
        <srmd:ClassificationEntry keyword="KEY">VALUE</srmd:ClassificationEntry>
    </srmd:Classification>
</srmd:SimulationResourceMetaData>
```

Abbildung 131: Beschreibung eines Paares aus KEY und VALUE in SRMD

In manchen Fällen ist zu beschreiben, dass ein *einzelnes Artefakt* sich tatsächlich aus *mehreren Dateien* zusammensetzt. Beispielsweise kann eine Spezifikation bestehen aus

- einer Zielbeschreibung, z. B. als Markdown oder Office-Text,
- einer Lösungsbeschreibung, z. B. als Foliensatz und
- einer Liste von Akzeptanzkriterien, z. B. als CSV-Datei.

Zu diesem Zweck ist es erlaubt, Einträge zum selben KEY, aber mit unterschiedlichem TITLE zu wiederholen:

```
<srmd:ClassificationEntry keyword="model.specification" xlink:href="https://my.server.de/objectives.md" xlink:title="objectives"/>
<srmd:ClassificationEntry keyword="model.specification" xlink:href="https://my.server.de/solution_design.pptx" xlink:title="solution design"/>
<srmd:ClassificationEntry keyword="model.specification" xlink:href="https://my.server.de/acceptance.xlsx" xlink:title="acceptance criteria"/>
```

Abbildung 132: SRMD-Einträge zum selben KEY, aber mit unterschiedlichem TITLE

Soll die Befüllung eines Metadatums nach einem *externen Attributierungsschema* erfolgen, so wurde dafür folgende Syntax vorgeschlagen:

```
<srmd:ClassificationEntry
  keyword="model.classfication"
  xlink:href="https://wiki.openstreetmap.org/wiki/Category:Features"
  xlink:title="highway=pedestrian, access:conditional=delivery @ (00:00-11:00), bicycle=yes,
               emergency=fire_hydrant, man_made=obelisk"
/>
```

Abbildung 133: Verwendung von OpenStreetMap als externes Attributierungsschema

Dieses Beispiel folgt dem Attributierungsschema von OpenStreetMap und attribuiert eine Fußgängerzone, in der Lieferverkehr nur bis 11 Uhr erlaubt ist, Fahrräder benutzt werden dürfen und in der es einen Feuerhydranten und einen Obelisken gibt.
Das entwickelte Verfahren eignet sich gleichermaßen, um real *vermessene* Verkehrsräume, daraus abgeleitete *vereinfachte* Verkehrsräume oder völlig synthetisch *erzeugte* Verkehrsräume zu attributieren:

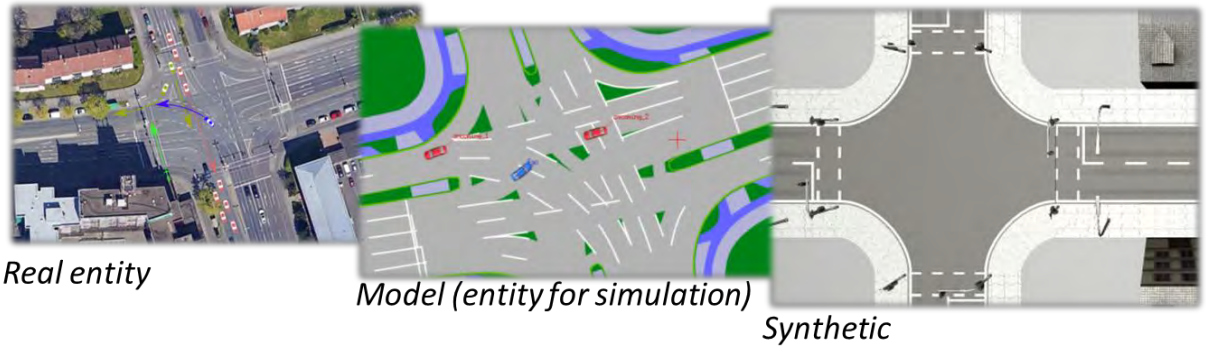


Abbildung 134: Realität – Simulationsmodel – Synthetisches Scenario

Mehr Details zur Syntax wurden in https://gitlab.setlevel.de/open/processes_and_traceability/traceability_data/-/blob/main/SETLevel_SRMD_and_classifications_for_metadata.pdf veröffentlicht.
AP 4.1 hat das erarbeitete Metadatenkonzept mit Beispieldaten aus SUC 1 untermauert. Abbildung 135 zeigt beispielhaft einen Teil der befüllten SRMD-Datei für das SUC 1 MS2 Szenario „TS-2-4_RS-2-4“.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<srmd:SimulationResourceMetaData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://apps.pmsf.net/STMD/SimulationResourceMetaData SRMD.xsd"
    xmlns:srmd="http://apps.pmsf.net/STMD/SimulationResourceMetaData"
    version="0.5" name="Scenario Model TS2-4_RS2-4 Meta-Data">
    <srmd:Classification type="de.sl4to5.srmd.model-meta-data">
        <!-- Scenario related information -->
        <srmd:ClassificationEntry keyword="model.type">Scenario</srmd:ClassificationEntry>
                <srmd:ClassificationEntry keyword="scenario.family">Intersection with occlusion</srmd:ClassificationEntry>
                <srmd:ClassificationEntry keyword="scenario.maneuver">turn right</srmd:ClassificationEntry>
                <srmd:ClassificationEntry keyword="scenario.name">TS2-4_RS2-4</srmd:ClassificationEntry>
                <srmd:ClassificationEntry keyword="scenario.nrtrafficparticip">3</srmd:ClassificationEntry>
                <srmd:ClassificationEntry keyword="scenario.trafficparticiptypes">Pedestrians (2), vehicle (1)</srmd:ClassificationEntry>
                <srmd:ClassificationEntry keyword="scenario.environmental">Sunny, 20°C</srmd:ClassificationEntry>
                <srmd:ClassificationEntry keyword="scenario.handedtraffic">Right-hand traffic</srmd:ClassificationEntry>
                <srmd:ClassificationEntry keyword="scenario.country-specific.tp">-</srmd:ClassificationEntry>
                <srmd:ClassificationEntry keyword="scenario.country-specific.sign">-</srmd:ClassificationEntry>
                <srmd:ClassificationEntry keyword="scenario.sunelevation">"90°"</srmd:ClassificationEntry>
                <srmd:ClassificationEntry keyword="scenario.sunazimuth">"180°"</srmd:ClassificationEntry>
                <srmd:ClassificationEntry keyword="scenario.wetstreet">false</srmd:ClassificationEntry>

                <!-- Description of the entity for simulation (aka model) -->
                <srmd:ClassificationEntry keyword="scenario.provider">SETLevel</srmd:ClassificationEntry>
                <srmd:ClassificationEntry keyword="scenario.version">1.0.0</srmd:ClassificationEntry>
                <srmd:ClassificationEntry keyword="scenario.recordingtype">synthetic</srmd:ClassificationEntry>
                <srmd:ClassificationEntry keyword="scenario.recordingtime">-</srmd:ClassificationEntry>
```

Abbildung 135: Metadaten des Scenarios TS2-4_RS-2-4

2.3.3.1.5 Tool-Validierung und Performance

Im Bereich der Simulation automatisierter Fahrfunktionen sind die eigentlichen Simulationswerkzeuge von ausschlaggebender Bedeutung. Sie sollten idealerweise schnell und zuverlässig arbeiten. Ihr Ressourcenbedarf sollte bei wachsender Modellgröße und -komplexität in akzeptabler Weise skalieren. Sie sollen sich leicht in eine anwendungsspezifische Simulationslandschaft integrieren und auch schnell austauschen lassen.
Zur Validierung der *Qualität* der Ergebnisse, die einzelne Simulationswerkzeuge in gegebenen Simulationsaufträgen mit zugehörigen Szenarien erzielen, ist ein Vergleich mit den Ergebnissen *physikalischer* Experimente notwendig. Die Durchführung solcher Experimente befand sich nicht im Scope des Projekts SET Level.
Daher konnten die im Projekt vertretenen Simulationswerkzeuge lediglich bzgl. der damit erzielbaren *Geschwindigkeit* der Simulation und bzgl. des Maßes der *Unterstützung einschlägiger Standards* betrachtet werden.

Validierungsverfahren & -ergebnisse

Im Projekt war vereinbart, die einzelnen Verhaltensmodelle (FMUs gemäß FMI-Standard) mittels OSI ans zentrale Simulationswerkzeug (den Core) anzubinden. Dementsprechend wurde die zeitliche Performance, mithin die Bandbreite dieser Anbindung untersucht. OSI-intern werden zur Serialisierung der zu übertragenden Datenstrukturen „Google Protocol Buffers“ verwendet.[14] Eine technische Alternative könnte in „Google FlatBuffers“ bestehen.[15]
Um diese und ähnlich gelagerte Fragen verlässlich zu beantworten, müssen Laufzeitversuche unternommen werden. Dazu müssen u. a. die Dauern von Serialisierung, Übertragung und Deserialisierung von Nachrichten möglichst exakt erfasst werden.
OSI selber bringt entsprechende Profiling-Funktionalitäten *nicht* mit.
Experten der Tool-Vendoren, Mitglieder der SUCs, MitarbeiterInnen der Forschungsinstitute und der OEM/Tier1 definierten dementsprechend gemeinsam, welche Nachrichten relevant sind, an welchen Stellen im Code welche Zeiterfassungen stattfinden sollen, und wie die

[14] Siehe https://developers.google.com/protocol-buffers?hl=de

[15] Siehe https://google.github.io/flatbuffers/

Messergebnisse abgelegt werden sollen. Die Entscheidung fiel auf ein Logfile-Format auf JSON-Basis. Es wird erwartet, dass auch zukünftige Erweiterungen damit abgebildet werden können.
Darauf aufbauend wurden die Schnittstellen eines entsprechenden, neu zu erstellenden Mess-Frameworks festgelegt. Es ist generell fürs OSI-Profiling ausgelegt, sollte also auch außerhalb des Kontextes von SET Level verwendbar sein. Daher wurde es in das OSMP-OSI Framework integriert. Das ermöglicht, Performance-Information aus allen OSI-FMU bekommen zu können, die mit diesem Framework entwickelt werden.
Das Framework wurde vom FZD implementiert.
Die SUC unterstützten nachfolgend tatkräftig mit der Durchführung der Evaluation.
Ein Verfahren zur graphischen Aufbereitung der Logdaten wurde entwickelt und realisiert. Dadurch konnte unter anderem sichergestellt werden, dass die gesammelten Performance-Daten von allen Partnern gleich interpretiert werden.
Den phänomenologisch teilweise bekannten Performance-Probleme von OSI für Ray-Tracing-basierte Sensoren konnten durch die Arbeiten im AP 4.1 zum ersten Mal präzise nachgewiesen werden.
Die Ergebnisse wurden mit den FMU-Modell-Lieferanten von SET Level in Rahmen von TP 3 geteilt. Diese Arbeiten wurden direkt von der Industrie angenommen und wurden im Rahmen der ASAM OSI Evaluierung angewendet.

2.3.3.1.6 Traceability und Traceability-Tool „TRACY“

Traceability

ISO 10007 „Quality management – Guidelines for configuration management“ definiert:

> *Configuration management is a **management activity** that applies technical and administrative direction over the life cycle of a product and service, its configuration **identification** and **status**, and related product and service configuration **information**.*

Wikipedia führt dazu aus:[16]

> *Its guidance is specifically recommended for meeting “the product identification and traceability requirements” introduced in ISO 9001:2015 “Quality management systems – Requirements” and AS9100 Rev D “Quality Management Systems – Requirements for Aviation, Space, and Defense Organizations”.*

Traceability ist Grundlage für die Glaubwürdigkeit von Simulationsergebnissen. TP 3 befasste sich unter dem Oberbegriff „Credible Simulation Process (CSP)“ intensiv damit.

TRACY – das Traceability-Tool

Mit Unterstützung eines leistungsfähigen Software-Werkzeugs ist es möglich, die Vielzahl an Artefakten, Teilprozessen und ihrer wechselseitigen Abhängigkeiten für die Simulation automatisierter Fahrfunktionen unter Kontrolle zu halten.
Daher wurde im Rahmen vom Projekt SET Level ein solches Software-Werkzeug als Demonstrator für Traceability unter dem Namen „TRACY“ entwickelt. Zu seiner Validierung wurde der CSP verwendet.

TRACY – Use Cases und Anforderungen

Bereits zu Projektbeginn wurde folgender Use Cases identifiziert:

- TRACY muss dokumentieren können:
  - Welche Szenarien wurden umgesetzt?

[16] Siehe https://en.wikipedia.org/wiki/ISO_10007

- Welche Modelle waren dazu nötig?
- Wie waren die Simulationstools konfiguriert?
- Welche Ergebnisse haben die Simulationsläufe gezeigt?

Die an AP 4.1 beteiligten Projektpartner haben sich an der Anforderungsspezifikation für TRACY aus den SUCs heraus beteiligt. Auf Grund der Bedeutung des CSPs fand eine besonders enge Abstimmung mit TP 3 statt.

TRACY – Artefakt-Arten

TRACY unterstützt u. a. folgende *Artefakt-Arten*:

- Requirement
- User Story
- Specification
- Test Case
- Model
- System / Systembestandteil

Für jedes Artefakt dieser Artefakt-Arten stehen folgende *Metadaten* zur Verfügung:

- id
- title
- rationale
- linkToResource
- repository
- idName
- displayName
- resourceType
- uri

Auch *eigene Metadaten* können angehängt werden. Alle Objekte verfügen dazu über eine Liste *generischer Key/Value/Type-Attribute*. Zulässige Typen sind Text, Numerisch, URL, Datum, Zeitstempel.
Darüber hinaus können Objekte mit *Tags* versehen werden.

Alle Informationen werden entsprechend des im TP 3 entwickelten **Credible Simulation Process (CSP)** aufgeführt (siehe Abbildung 136) und können insbesondere auf ihre Relationen hin untersucht werden (z. B. welches Modell welche Anforderung erfüllen muss).

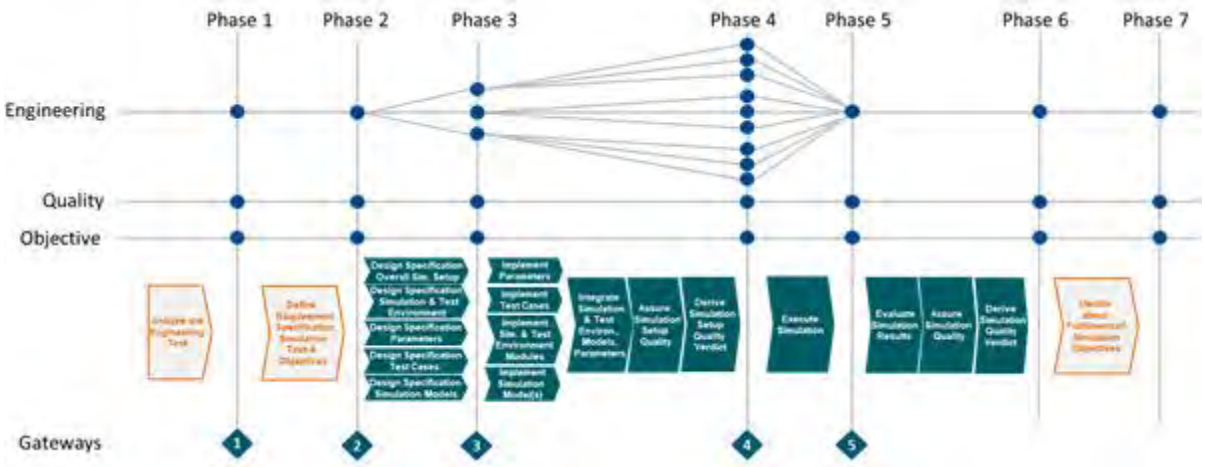


Abbildung 136: Schematische Darstellung der Artefakt-Relationen über den CSP hinweg

Der CSP wird von TRACY vollständig unterstützt.

TRACY – Funktionalitäten

Um die Anforderungen abzudecken, wurden in TRACY u. a. folgende Funktionalitäten realisiert:

- User-Management
- Rollen- und Rechte-Management
- Grafische Benutzeroberfläche (GUI)
- User-spezifische Cockpits und Sichten
- Anlegen und Suchen von Artefakten (inkl. Metadaten)
- Anlegen von Trace-Links zwischen Artefakten (Relationen)
- Abbilden des Credible Simulation Processes
- Befüllen von TRACY mit Projektdaten
- Workflow zur Qualitätssicherung entsprechend CSP
- Einbindung weiterer Systeme nach dem Standard OSLC (beispielsweise DOORS)
- Grafische Darstellung von verwalteten Artefakten und ihrer Relationen
- Möglichkeiten zur Durchführung dynamischer Impact-Analysen (Filter)
- Realisierung einer ersten Baselining-Funktionalität
- Aktualisierung der CSP- und Einführung der CMP-Templates
- Erweiterung der Darstellungs- und Filtermechanismen in der Graph-Ansicht
- Realisierung von Notifications
  (Hinweise, z. B. wenn einem Benutzer ein neuer Task zugewiesen wurde)
- Überführung der TRACY-Software in eine Cloud-Umgebung

- Löschen von den in TRACY verwendeten Entitäten
- Abspeichern von ersten individuellen User Settings

Auf einige dieser Funktionalitäten wird nun näher eingegangen.

TRACY – Benutzeroberfläche
TRACYs grafische Benutzeroberfläche (GUI) ist in drei Säulen gegliedert:
Die Graph-Darstellung der Informationsstrukturen steht im Zentrum der Applikationsoberfläche. Hier hat der Benutzer die Möglichkeit, über Filter Impact-Analysen durchzuführen. Die dargestellten Informationen wurden zudem angereichert. Beispielsweise wird der Reifegrad der Informationen angezeigt und auch dargestellt, ob es sich um verlinkte oder ausschließlich in TRACY gehaltene Informationen handelt.
Links daneben findet der Anwender die globale Navigationsleiste und daneben die Strukturierung der Informationen.
Rechts vom Graph findet der Nutzer Details zu selektierten Informationen, Metainformationen, Status, Historie und Abhängigkeiten:

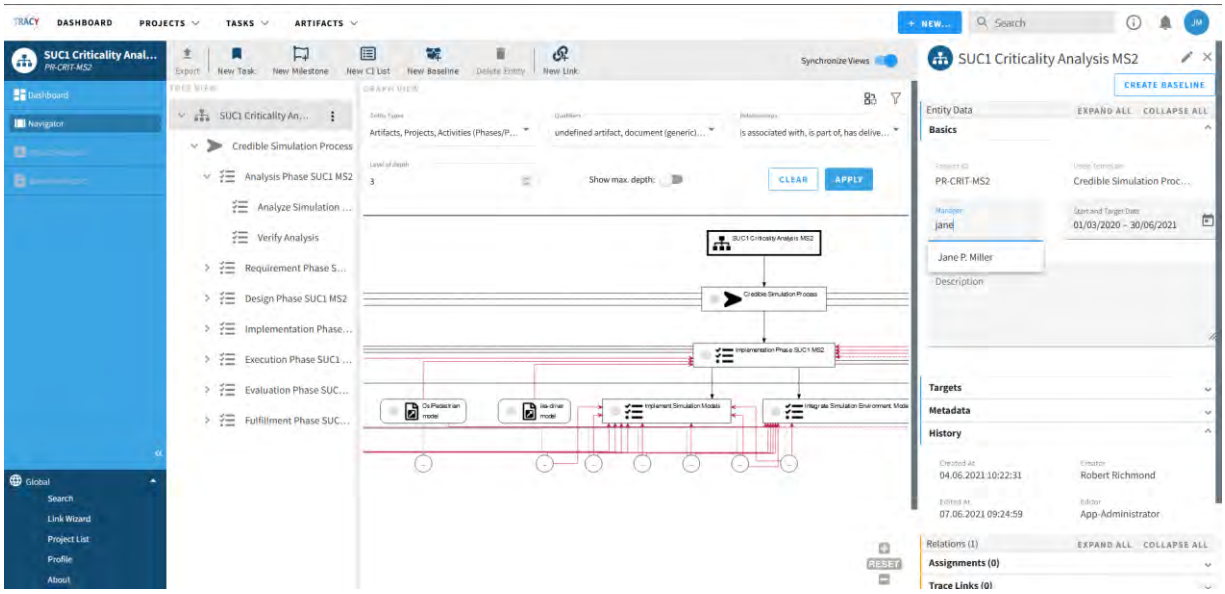


Abbildung 137: Gliederung der TRACY-GUI

TRACY – Metadaten-Funktionalitäten
Für das Konsortium ist es von entscheidender Bedeutung, neben den eigentlichen Modelldaten auch deren **Metadaten** (siehe auch Kapitel 2.3.3.1.4) zentral und einfach pflegen, suchen etc. zu können. Dazu wurde in TRACY ein Editor realisiert, der es jedem Anwender ermöglicht, Metadaten zu editieren und abzuspeichern. So ermöglicht TRACY dem Benutzer ein einfaches Management der jeweiligen Artefakte, das Pflegen der zugehörigen Metadaten sowie weiterer Daten wie Statusinformation, Zuständigkeiten, Anmerkungen (Rationales) etc. in einer Oberfläche (siehe Abbildung 138).

Dabei verwaltete TRACY die Metadaten der Artefakte sowie deren Beziehungen untereinander, nicht aber die Artefakte selbst, die im GitLab persistiert sind.

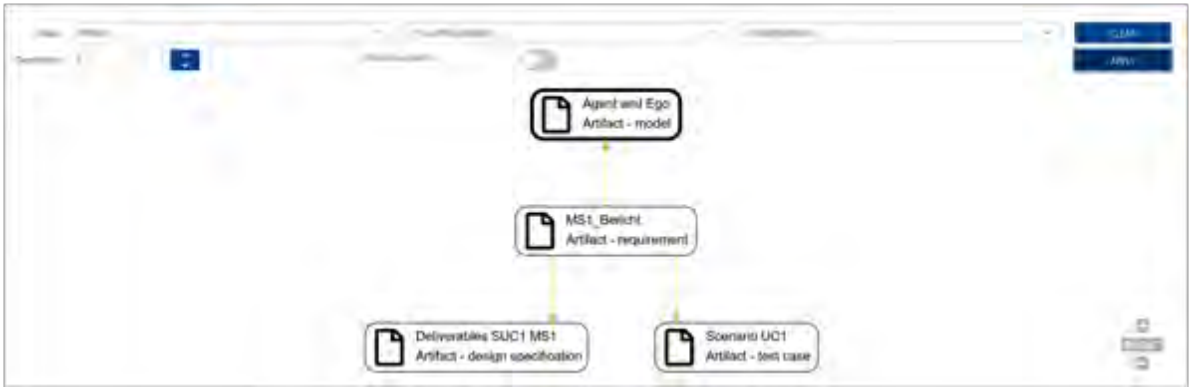


Abbildung 138: Graphische Darstellung von Artefakten und ihren Relationen (Filter aktiviert)

TRACY – Impact-Analyse

Relevante Informationen in Bezug zu setzen ist eine der wesentlichen Funktionalitäten von TRACY. Um effizient damit arbeiten zu können, benötigen Anwender jedoch auch visuelle bzw. graphische Unterstützung und die Möglichkeiten, nach gewissen Informationen zu suchen und zu filtern. Hierunter fallen sogenannte **Impact-Analysen**, die Auskunft darüber geben, welche Artefakte Einfluss auf welche anderen Artefakte haben. Anwender möchten zum Beispiel danach filtern können, welche Requirements mittelbaren oder unmittelbaren Einfluss auf die Modellentwicklung haben.

Durch einfaches Setzen von Filtern kann ein Anwender dies in TRACY grafisch visualisieren. In Abbildung 139 ist exemplarisch gezeigt, welche Projekte von einer spezifischen Anforderung (Requirement) beeinflusst werden. Ein Anwender kann sich so schnell einen Überblick darüber verschaffen, was z. B. bei einer Änderung der Anforderung betroffen ist und wer zu informieren ist. Generell können diese Abhängigkeitsuntersuchungen zwischen jeder Art von Artefakten durchgeführt werden.

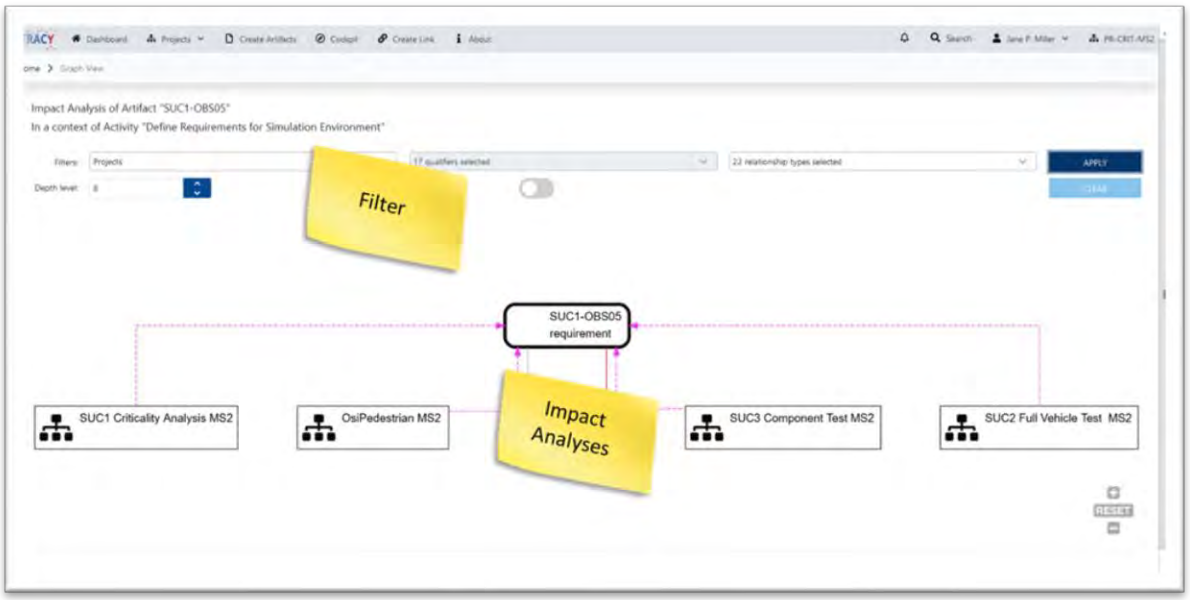


Abbildung 139: Exemplarische Anwendung des grafischen Filtermechanismus

TRACY – Grafische Analyse

Ein weiteres Feature ist die Graphenanalyse, die die Impakt-Analyse ergänzt und unterstützt. Neben der generischen Analyse, die durch komplexe Benutzereingaben konfigurierbar ist, wurden vorkonfigurierte Analysen implementiert.

In der *Where-Used*-Analyse werden alle Projekte gelistet, in denen die betrachtete Entität genutzt wird. Für Configuration Items sind zudem noch eine *Show-Impacted*-Entity- und eine *Show-Impacting*-Entity-Funktionalität zur Verfügung gestellt worden (siehe Abbildung 140). Diese Analysen ziehen automatisiert Verbindungen zwischen Configuration Items, die sich gegenseitig beeinflussen. Der Einfluss wird dabei durch die Verbindung über Planungselemente, wie bspw. Aktivitäten, und deren Relationen dazu (Input, Guideline und Output, Rationale) ermittelt (siehe Abbildung 141).

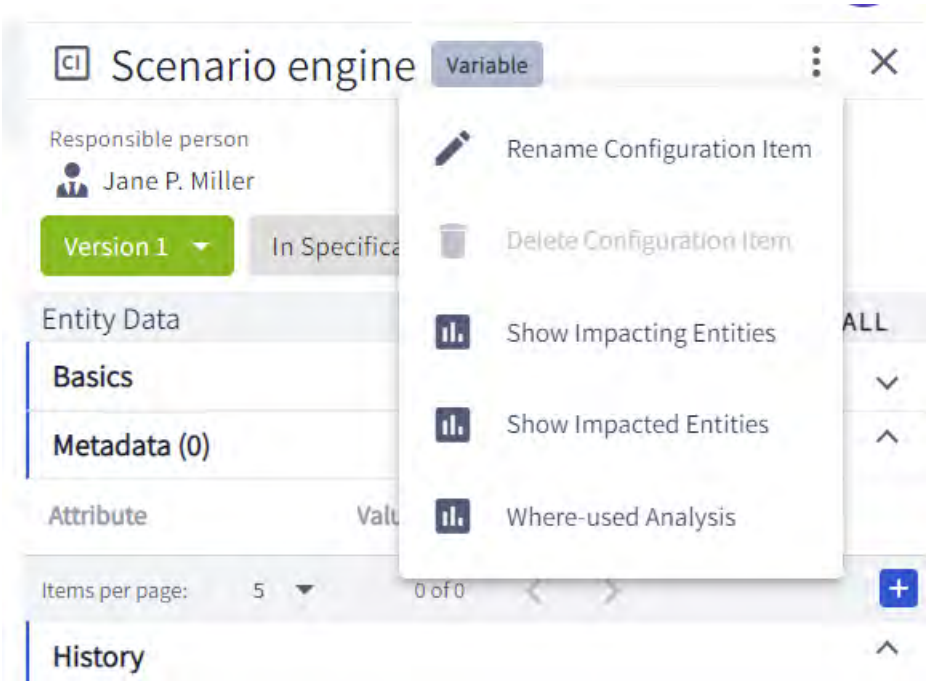


Abbildung 140: Auswahl der möglichen vorkonfigurierten Analysen

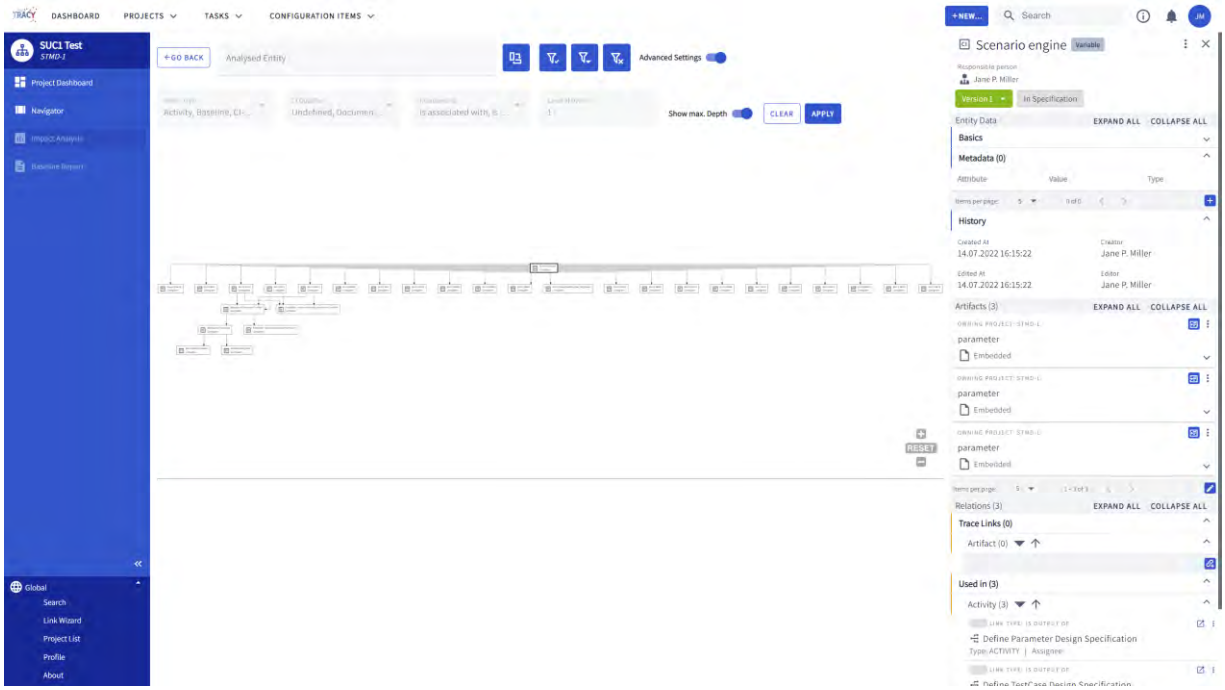


Abbildung 141: Exemplarische Durchführung der Impacted-Entities-Analyse

TRACY – Baselining

Mit einer ersten **Baseline**-Lösung (sog. *Ad-Hoc Baselines*) konnten selektierte Artefakte gesichert und mit einem Zeitstempel versehen werden.

Nachfolgend wurde die Erstellung von Baselines und das Ableiten von Reports weiterentwickelt. Dabei ging es darum, dass Arbeitsstände zu einem beliebigen Zeitpunkt *t* „eingefroren“ werden und somit jederzeit nachvollziehbar ist, was der Stand der Arbeit zum Zeitpunkt *t* war (z. B. der Stand zu einem Berichtszeitpunkt oder der Stand der Ergebnisse, die mit einem Partner ausgetauscht wurden, oder der Stand zum Projektende etc.). Darauf aufbauend

wurde eine Report-Funktion in TRACY realisiert, um einen Übersichtsbericht basierend auf der Baseline abzuleiten (siehe Abbildung 142).

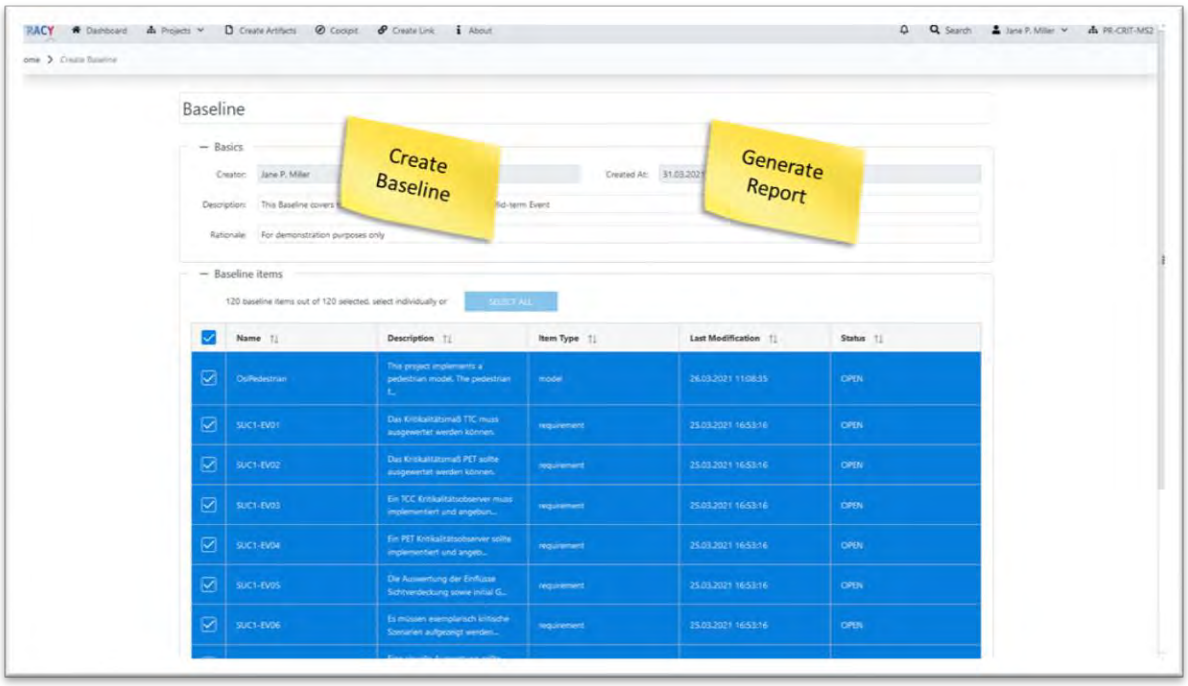


Abbildung 142: Exemplarische Darstellung zum Ableiten einer Baseline

Planned Baselines sind Vorlagen (Templates) für die oben beschriebenen Ad-Hoc Baselines. Eine Planned Baseline definiert Traversierungsregeln, um durch den Graphen der Datenbasis zu navigieren und außerdem Auswahlfilter dafür, welche Entitäten in einer gezogenen Ad-Hoc Baseline landen sollen. Ergänzt durch einen *Trigger*, der aus einem Element (bspw. Aktivität) und einem Trigger-Status (Release Status) besteht, kann die Planned Baseline automatisch instanziiert werden, sobald das Trigger-Element den angegebenen Trigger-Status erreicht. In dem Fall wird aus der Vorlage eine Ad-Hoc Baseline erzeugt, um die durch

die Traversierungs- und Filterregeln definierte Datenmenge einzufrieren. Abgesehen von dem automatischen Trigger ist auch ein manuelles Auslösen einer Planned Baseline möglich:

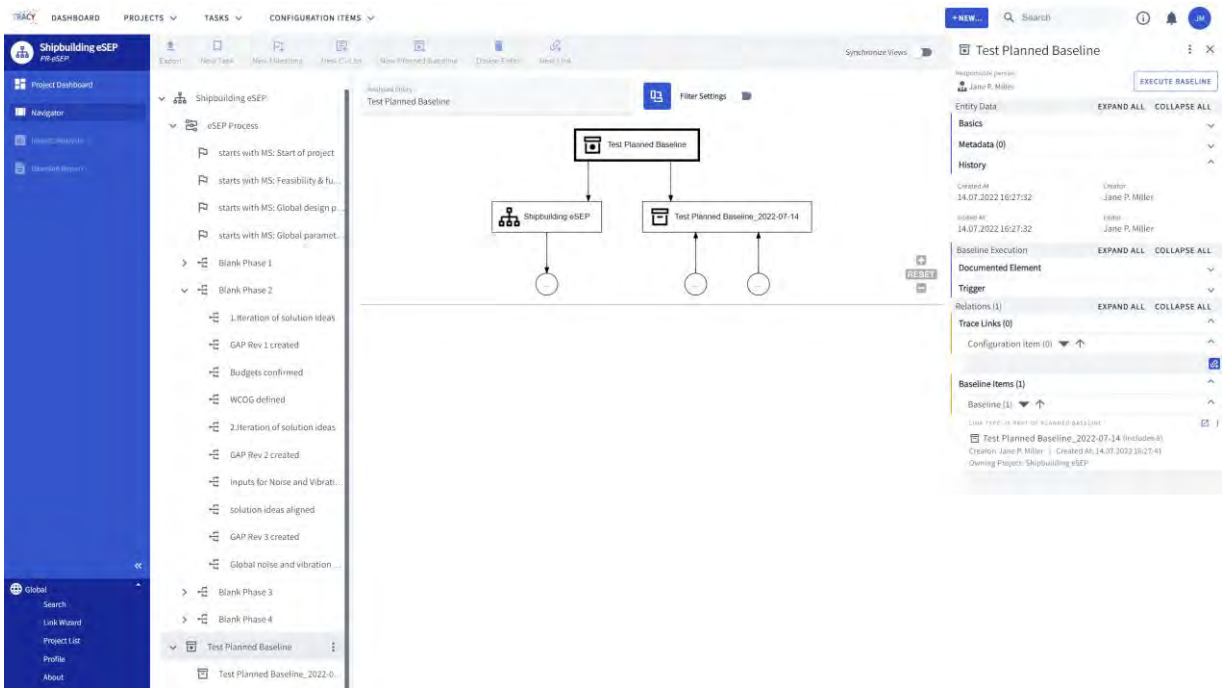


Abbildung 143: Planned Baseline im Navigator (Tree View links) und Detail View (rechts) mit ausgeführter Ad-Hoc Baseline

TRACYs Datenmodell gruppiert Objektrelationen und Trace-Links in sog. „Link Familien“, die bei Abfragen und Analysen einheitlich verwendet werden können. Aktivitäten wurden verallgemeinert, sodass Prozesse, Phasen und Aktivitäten lediglich durch einen Qualifier spezifiziert werden und keine unterschiedlichen Klassen sind.

TRACY – Configuration Items
Configuration Items (CIs) stellen eine Art Hülle um Artefakte dar, die eine Versionierung zulassen. Meta-Informationen, wie Qualifier, Release Status, Beschreibung, Bearbeiter, etc. werden dem CI zugeordnet. So kann sichergestellt werden, dass Artefakte stets unveränderlich bleiben und die Evolution eines Configuration Items über den Simulationsprozess dargestellt werden kann.
Beispiel: Die Anforderung an ein Modell beginnt als Einzeiler (Version 1), wird anschließend in einer Aktivität ausgearbeitet und ergibt ein Dokument (Version 2). Im Laufe der Simulations-Iterationen werden die Anforderungen ggf. überarbeitet (Version 3 und ggf. weitere Versionen). Am Ende der Simulation steht die aktuelle Version der Anforderung, aber in der Datenmenge ist erkenntlich, zu welchem Arbeitsschritt welches Artefakt für diese Anforderung vorgelegen hat. Entsprechend wurden Trace Links und Relationen in TRACY so erweitert,

dass entweder eine konkrete CI-Version oder eine Laderegel (bspw. aktuelle Version) hinterlegt werden kann:

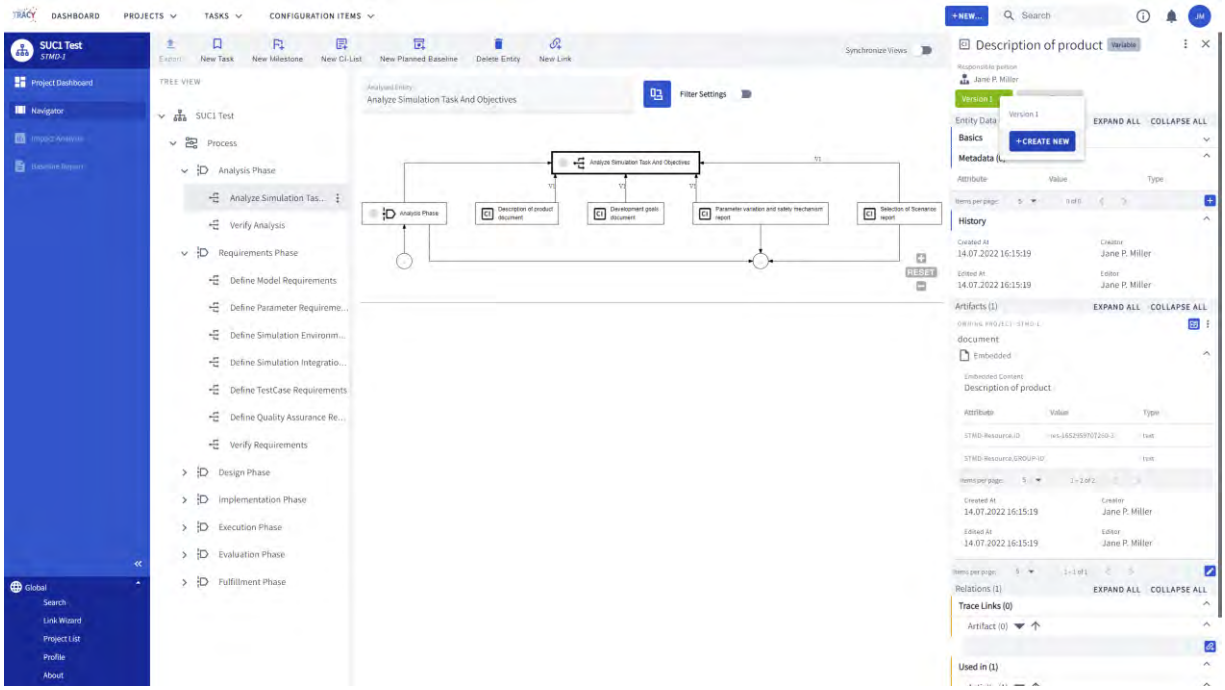


Abbildung 144: Ansicht eines Configuration Items in der Detail View (rechts) mit Auswahlmöglichkeit der Version

Configuration Items können zusätzlich genutzt werden, um mehrere Artefakte zu einem Objekt zusammenzufassen (sog. *Composite* Configuration Item). Dies ist beispielsweise von Nutzen, wenn mehrere Dokumente ein Modell beschreiben:

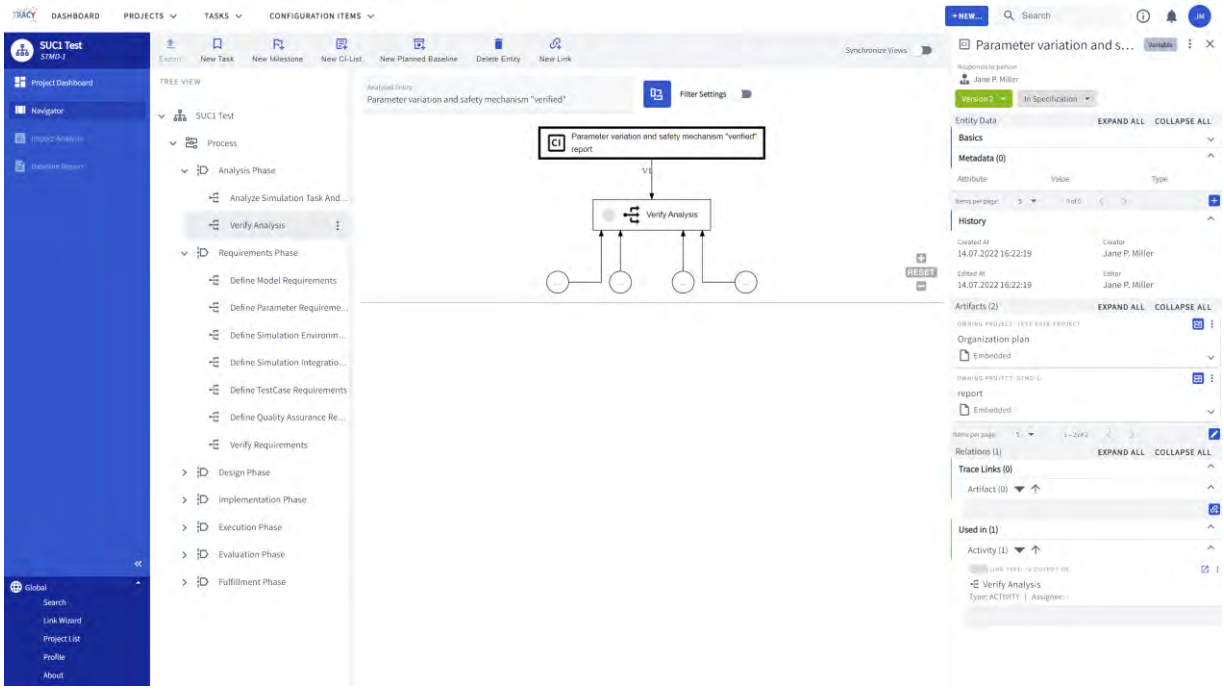


Abbildung 145: Darstellung eines Composite Configuration Items in der Detail View (rechts)

TRACY – Projektverwaltung
TRACY bietet eine rudimentäre **Projektverwaltung**. Im Rahmen solcher Projekte sind Personen Rollen zugewiesen. Prozesse und Artefakte haben einen Bezug zu Projekten.
Ein Projekt kann geschlossen werden und wird dadurch eingefroren. Geschlossene Projekte können aber auch wieder geöffnet oder gelöscht werden. Zudem wurde ein Projektexport in einem TRACY-nativen XML-Format implementiert, mit dem Daten ohne Verlust zwischen TRACY Instanzen ausgetauscht werden können:

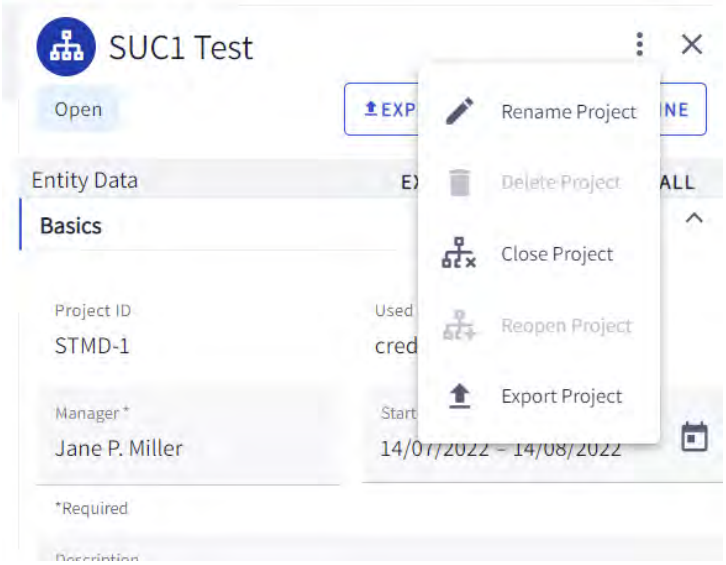


Abbildung 146: Löschen, Schließen, Öffnen und Exportieren von Projekten

TRACY – Dashboard
Zusätzlich zum globalen **Dashboard** (Überblick über *mehrere* Projekte) wurde ein projektspezifisches Dashboard eingeführt. Dieses Projekt-Dashboard zeigt lediglich Elemente an, die im Kontext *eines* Projektes angelegt bzw. genutzt wurden. Neben den bekannten Kacheln für Aktivitäten und Configuration Items wurden Kacheln für zugewiesene Elemente in bestimmten Release Phasen („In Work“ und „In Review“), letzte Aktivitäten im Projekt, sowie ein Gantt Chart zur Projektübersicht hinzugefügt. Die Kachel für Projekte ist im Projekt Dashboard selbst nicht vorhanden.

TRACY – Rollen & Rechte
TRACYs Rollen- und Rechte-Management umfasst u. a. einen Release-Prozess mit Zugriffsrechten derart, dass nur die aktuell zugewiesene Person die Aktivitäten bearbeiten kann. Die Fortschreitung des Release-Prozesses bleibt dabei unverändert. Auf Configuration Items wurde derselbe Release Prozess inkl. Zugriffsrechten angewandt.

TRACY – Schnittstellen

Multi-System-Connectivity ist für Traceability allgemein von entscheidender Bedeutung, da die Informationen von Interesse oft über eine Vielzahl von Systemen hinweg verstreut sind und erst über TRACY in Bezug zueinander gesetzt werden können.
Für das Projekt SET Level ist es von besonderer Bedeutung, dass TRACY auf *Artefakte im GitLab* zugreifen und sie persistent verlinken kann. TRACY besitzt daher eine **Git-Schnittstelle**, so dass sich TRACY mit dem projekteigenen GitLab-Server verbinden kann.

Der standardbasierte **OSLC-Connector** ermöglicht das Einbinden einer Vielzahl an Systemen:

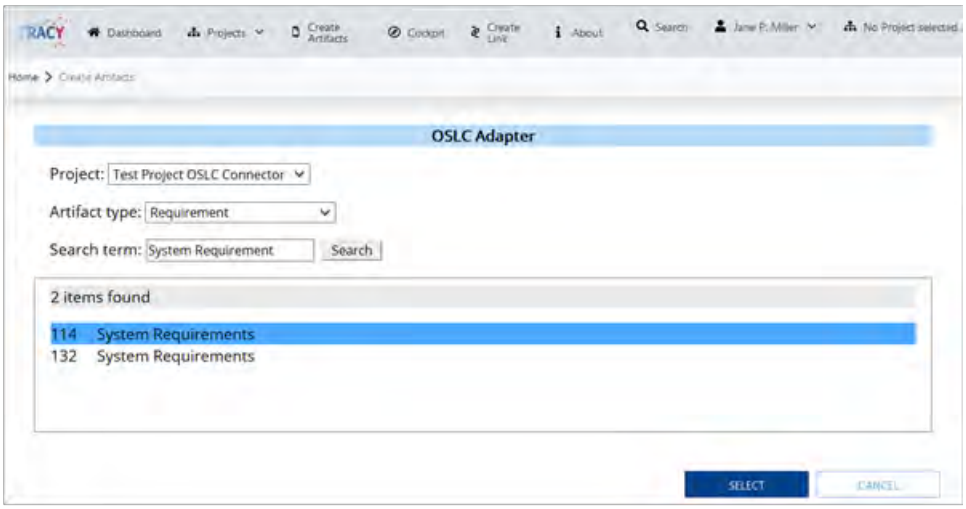


Abbildung 147: TRACY OSLC Connector, exemplarisches Einbinden von Requirements

TRACY – Erprobung

Während der gesamten Laufzeit von SET Level wurden neu entwickelte TRACY-Funktionalitäten umgehend dem Projekt zur Verfügung gestellt und erprobt. Die im Kontext der SUCs erzeugten Artefakte samt der Verkehrsräume und Szenarien dienten dabei direkt als praxisnahe Testdaten.
Das Befüllen von TRACY mit diesen Testdaten erfolgte in enger Zusammenarbeit mit den SUCs und AP 2.1.
Auf Betreiben der Industriepartner wurde TRACY in der Cloud bereitgestellt, um so eine schnellere und individuelle Verprobung auch mit eigenen Daten realisieren zu können. Zuvor stand dem Projekt lediglich eine einzelne, zentrale TRACY-Instanz zur Verfügung, in der die Arbeiten der SUCs dokumentiert und die im GitLab zur Verfügung gestellten Informationen und Modelle verlinkt wurden.
So konnten die Industrialisierbarkeit der Projektergebnisse in den Häusern der Partner intensiv vorangetrieben und Interaktionsszenarien zwischen den Partnern, auch unter realistischen Bedingungen, verprobt werden.
Der Einsatz von Cloudinstanzen wurde von den Projektpartnern als sehr wertvoll angesehen, war aber ursprünglich nicht geplant. Im Einzelnen wurden Instanzen für VW, BMW, Bosch, Continental, Ford und Opel bereitgestellt.
Die gesammelten Erfahrungen wurden über TP 5 in die anderen Teilprojekte zurückgespiegelt.

TRACYs Issue Management Process

Wie oben beschrieben, wurde TRACY in enger Folge der Partnerwünsche dynamisch weiterentwickelt. Um das Vorgehen dabei zu veranschaulichen, wurde ein prototypischer Prozess

für das Issue- und Anforderungsmanagement für ein Simulationswerkzeug anhand des Beispiels TRACY dargestellt:

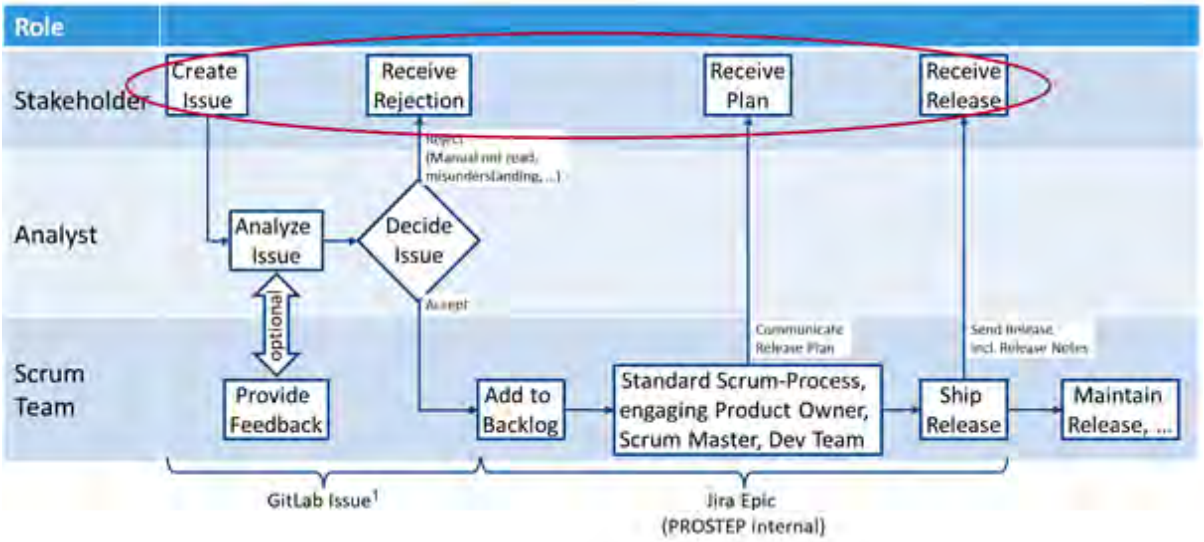


Abbildung 148: TRACY Issue Management Prozess

2.3.3.2 Closed-loop Verkehrssimulation

2.3.3.2.1 Closed-loop Verkehrssimulation für Analyseaufgaben

Eine Kritikalitätsanalyse wird zu verschiedenen Zeitpunkten der Entwicklung eines sicherheitskritischen Systems durchgeführt. In einem frühen Stadium, wenn eine Spezifikation des zu entwickelnden Systems inklusive seines Einsatzbereiches gegeben sind, werden anhand von bekannten sowie vorstellbaren Gefährdungsursachen mögliche kritische Ereignisse, die bei Nutzung des Systems auftreten könnten, und zugehörige Wirkketten identifiziert. In späteren Entwurfsphasen, wenn mehr Details des Systems festgelegt sind, werden solche Analysen wiederholt und vertieft.
Zur Analyse werden verschiedene Techniken eingesetzt. Während in frühen Phasen manuelle Techniken dominieren, können in späteren Phasen, je greifbarer das System geworden ist, wirkungsvoll auch Berechnungsverfahren eingesetzt werden. Insbesondere bietet sich hier für die Simulationstechniken aus SET Level ein interessantes und praktisch hoch relevantes Anwendungsfeld. Denn die Gefährdungen bei (hoch-)automatisierten Fahrzeugen manifestieren sich in Szenarien, welche in Unfällen enden oder diese knapp vermieden werden. Dies motiviert das Aufsetzen von Simulationen, wo eine automatisierte Fahrzeugsteuerung in Szenarioräumen agiert, die möglicherweise zu unerwünscht kritischen Situationen führen können. Eine solche Simulation ist „Closed-loop“: Die Automation agiert in einem Szenario, welche das Verhalten der Automation nicht beschränkt. Gemessen wird, wie kritisch unter Verkehrssicherheitsgesichtspunkten ein konkreter Simulationslauf ausfällt.
Sinnvoll für die meisten Anwendungen ist, dafür eine Vielzahl von Szenarien in einem logischen Szenario zu beschreiben, und dann den so aufgespannten Szenarioraum automatisiert zu explorieren. Auf diese Weise profitiert man in zweifacher Hinsicht von den Möglichkeiten virtueller Tests: Zum einen lassen sich bereits in frühen Entwicklungsstadien recht gute Aussagen über das Verhalten treffen, zum anderen erzielt man eine hohe Abdeckung von Ausprägungen der späteren Anwendungsumgebungen und ihrer Herausforderungen.
Innerhalb des Entwicklungsprozesses kann man solche Simulationen sehr vielfältig für Zwecke der Kritikalitätsanalyse nutzen.

- In frühen Phasen kann man bekannte Unfallkonstellationen mit einer noch sehr ungenauen Funktionalität der Automation auf Relevanz prüfen und, sofern dies gegeben ist, mit Ablaufdetails unterlegen.
- Beim Entwurf von Mitigationsmechanismen lässt sich, z. B. mit Hilfe einer prototypischen Umsetzung in einem Modell, die Wirksamkeit prüfen (zunächst qualitativ, mit mehr Aufwand auch quantitativ).
- Vertiefte Analysen der Kritikalität in späteren Entwurfsphasen durch Simulation werden wertvolle Informationen dafür liefern, welche Gefährdungen erfolgreich abgefangen werden können und wo spätere Untersuchungen – etwa die Bestimmung von Risikowerten – noch erforderlich sind.

Diese Überlegungen mündeten im Projekt in der Definition eines Rahmens für den Simulation Use Case 1 (SUC 1), mit den wesentlichen Setzungen:

- Logisches Szenario mit möglicherweise kritischen Instanzen
- Closed-Loop-Simulation einer Automation
- Kritikalitätsindikatoren als wesentliche Bewertungsfunktion einzelner Läufe
- Exploration des Szenarioraumes mit dem Ziel der Bestimmung kritischer Instanzen.

Zu Beginn des Forschungsprojekts SET Level wurde die Ausgestaltung der eben genannten Konzepte im Kontext des SUC 1 vorgenommen. Darin enthalten war die Begriffs- und Methodikdefinition der „Wirksamkeitsanalyse“ und der „Kritikalitätsanalyse“. Dabei beschreibt die „Wirksamkeitsanalyse“ eine Methode zur Bewertung der positiven Risikobilanzierung von ADAS und AD (unfall- bzw. verkehrsbasiert), während die „Kritikalitätsanalyse“ einen Ansatz zur Identifizierung von kritischen Verkehrssituationen darstellt (auf Basis von Kritikalitätsphänomenen). Die Grundlage für beide Analysen ist dabei die Durchführung von Verkehrssimulationen.
Im SUC 1 wurde eine „Kritikalitätsanalyse“ aufgebaut und durchgeführt; dabei wurde eine systematische Untersuchung des Szenarioraums durchgeführt in Bezug auf kritische Verkehrssituationen zur Identifikation von kritikalitätserhöhenden Parametern / Kombinationen von Parametern des Szenarios und auch des untersuchten Systems entsprechend eines oder mehrerer Kritikalitätsmaße. Zur Umsetzung des SUC 1 wurde im weiteren Verlauf der Detaillierung die Verwendung von openPASS als open source Simulationstool bestätigt, welches im Laufe des Forschungsprojekts bei BMW, LBF und DLR SE zum Einsatz kam; der Zeitplan und das weitere Vorgehen wurden unter den Partnern abgestimmt. Darüber hinaus fand kontinuierlich eine Interaktion mit dem Forschungsprojekt VVMethoden sowie ein Erfahrungsaustausch statt. Die vom Forschungsprojekt VVMethoden zur Verfügung gestellten Definitionen wurden analysiert und kommentiert und über den Partner DLR SE wieder zurückgespielt, da das DLR SE in beiden Forschungsprojekten vertreten ist.
Die Weiterentwicklung von openPASS erfolgte dabei im open source Verbund mit der openPASS Working Group innerhalb der Eclipse Foundation (siehe https://openpass.eclipse.org/). Diese Working Group besteht seit 08/2016; die Gründungsmitglieder sind BMW, Mercedes-Benz, Volkswagen und itk. 2018 sind drei weitere Mitglieder beigetreten: TÜV Süd, Toyota und Bosch. Die Beiträge des Forschungsprojekts SET Level zum open source Projekt openPASS können funktional unter dem Stichwort „Befähigung für Stadtszenarien“ und technisch als „Erweiterung der Unterstützung für offene Simulationsstandards“ zusammengefasst werden; eine ausführliche Beschreibung der Inhalte findet sich im nächsten Abschnitt.

Der SUC 1 wurde in enger Zusammenarbeit einer Vielzahl von Partnern aufgebaut. Dazu trug insbesondere die Verwendung von Arbeitsergebnissen aus vielen Teilprojekten innerhalb

SET Level bei. Im Rahmen des Teilprojekts 4 übernahm BMW u. a. die zentrale Koordination zur Befähigung und Durchführung der SUC 1 Simulation. Das DLR SE übernahm als Schwerpunkt im SUC 1 die Implementierung des Simulation Platform Managements, welche sowohl das Pre-Processing als auch das Post-Processing der Simulationsaufgabe übernahm.
Der SUC 1 entwickelte sich über den Projektverlauf weiter von einer einfachen Kreuzung ohne Verkehrsschilder mit lediglich zwei Verkehrsteilnehmern (ein Auto und ein Fußgänger) über eine weiterhin einfache Kreuzung mit Verkehrsschildern und deutlich mehr Verkehrsteilnehmern (mehrere Autos und mehrere Fußgänger) hin zu einer komplexen Kreuzung mit Ampelregelung und vielen Verkehrsteilnehmern (mehrere Autos und mehrere Fußgänger). Im Folgenden werden charakteristische Zwischenstände in der Evolution des SUC 1 ausführlich beschrieben.
In seiner ersten Version wurde der SUC 1 exemplarisch an einer einfachen X-Kreuzung beschrieben. Dabei bog ein Fahrzeug (Ego-Fahrzeug) an der Kreuzung rechts ab und musste dabei auf einen kreuzenden Fußgänger reagieren, der sich zunächst parallel zum Fahrzeug bewegte und beim Überqueren der Straße die Trajektorie des Fahrzeugs kreuzte (siehe Abbildung 149). Die Bewegungen des Fahrzeugs und des Fußgängers wurden dabei mittels eines Trajektorienfolgers umgesetzt.

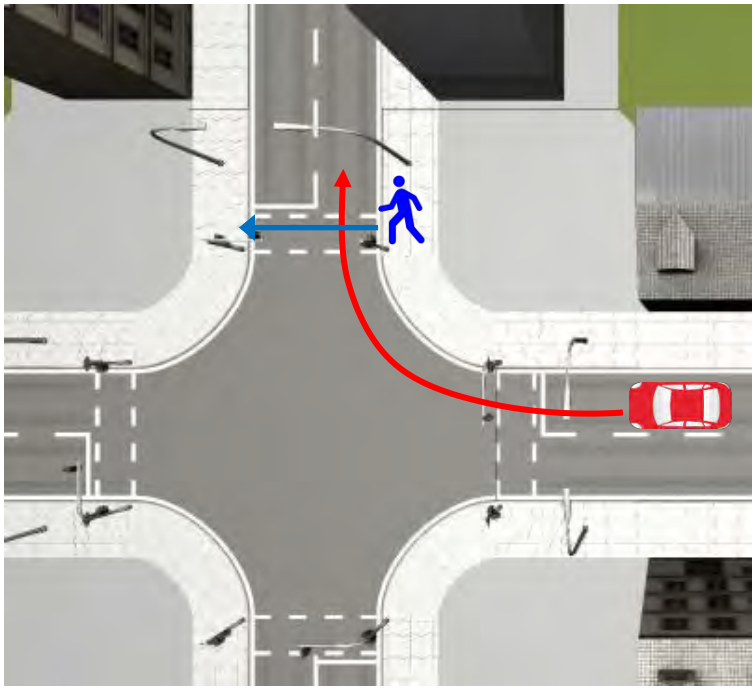


Abbildung 149: Systematische Darstellung SUC 1 – einfache Kreuzung ohne Verkehrsschilder

Die Beschreibung der Karte erfolgte gemäß dem Standard OpenDRIVE; openPASS musste dazu diese einfache Kreuzung laden können und die Agenten sowohl an der korrekten Stelle spawnen (initialisieren) als auch diese korrekt auf der Kreuzung mit den verschiedenen Spuren lokalisieren können. Das entsprechende Szenario war als OpenSCENARIO beschrieben, auch hier war eine korrekte Interpretation des Szenarios in openPASS erforderlich. Hierzu zählte auch das Einlesen und Interpretieren der Trajektorien aus dem TrajectoriesCatalog.
In einer zweiten Version war der SUC 1 nach wie vor über das Grundszenario „Rechtsabbieger“ an einer einfachen X-Kreuzung beschrieben worden (siehe Abbildung 150), allerdings in einer deutlich umfangreicheren Ausprägung. Ziel der Erweiterung war es, das Szenario realitätsnäher zu gestalten und Interaktionseffekte zwischen den Verkehrsteilnehmern zu berücksichtigen:

- Die Kreuzung verfügte diesmal über eine schilder-basierte Vorfahrtsregelung (Vorfahrtsstraße sowie Vorfahrt gewähren).
- Im Szenario befanden sich zusätzlich zum Ego-Fahrzeug weitere bis zu sechs Fahrzeuge, welche die Kreuzung überquerten oder auch parkten und somit direkt oder indirekt auf das Verkehrsgeschehen und damit auf das Szenario „Rechtsabbieger“ Einfluss nahmen. So fuhr z. B. ein Fahrzeug direkt vor dem Ego-Fahrzeug oder es stand ein Fahrzeug parkend auf dem Seitenstreifen zur Untersuchung von Sichtbehinderung.
- Darüber hinaus gab es weitere Fußgänger, welche den Gehsteig bevölkerten; ein weiterer Fußgänger griff dabei direkt auf das Szenario ein, indem er an der gleichen Stelle die Straße überquerte, aber diesmal von der anderen Seite, er kam also somit dem Ego-Fahrzeug entgegen. Das Ego-Fahrzeug musste mit zwei Fußgängern interagieren; insgesamt waren bis zu acht Fußgänger im Szenario verfügbar.
- Der Trajektorienfolger des Ego-Fahrzeugs war durch eine automatisierte Fahrfunktion ersetzt worden; der Umgebungsverkehr wurde durch ein einfaches Fahrermodell gesteuert.
- Die Fußgänger waren nun auch nicht mehr über einen Trajektorienfolger gesteuert, sondern über ein eigenes, im Projekt SET Level entwickeltes Fußgängermodell.

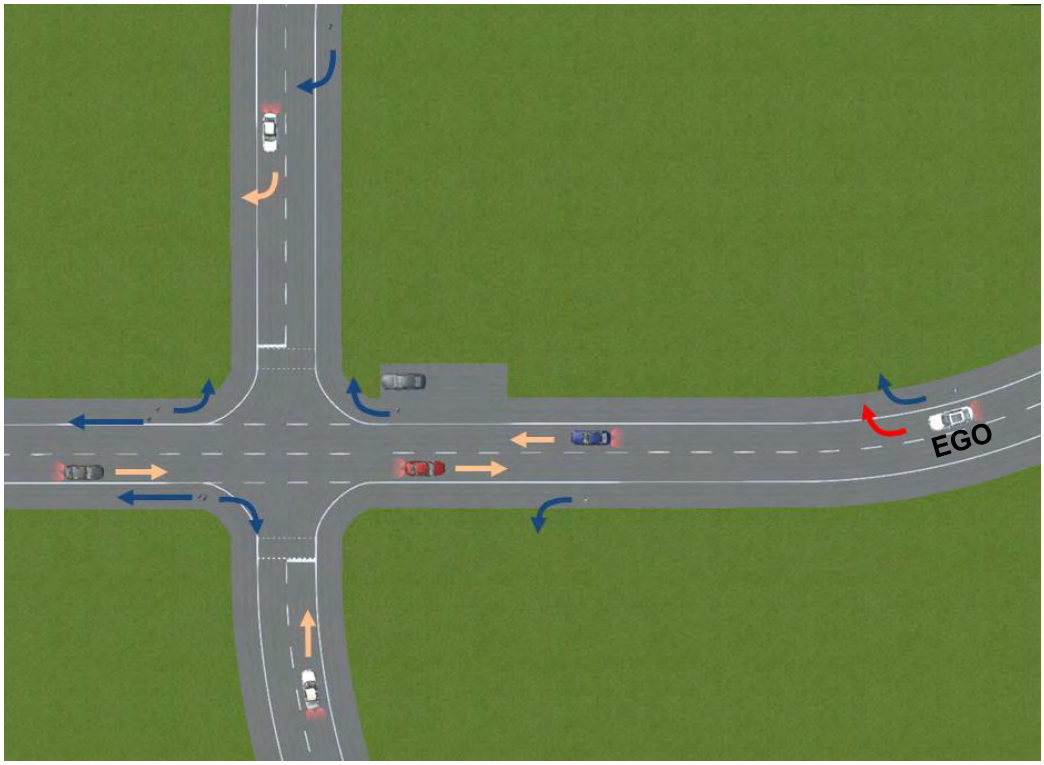


Abbildung 150: Systematische Darstellung SUC 1 – einfache Kreuzung mit Verkehrsschildern

Diese Ausbaustufe des SUC 1 erforderte die Umsetzung weiterer Features aus dem Standard OpenDRIVE. So war die Definition von urbanen Verkehrszeichen erforderlich („Vorfahrt“ und „Vorfahrt gewähren“) um das Szenario korrekt abbilden zu können. Für den Gehweg, auf dem sich die zahlreichen Fußgänger bewegten, wurde der zusätzliche LaneType „sidewalk“ eingeführt. Für die Untersuchung von Verdeckungsszenarien war die Platzierung eines zusätzlichen Fahrzeugs auf einem Parkplatz neben dem Gehweg notwendig; dafür wurde der zusätzliche LaneType „parking lot“ herangezogen. Das Szenario war ebenfalls wieder gemäß

dem Standard OpenSCENARIO definiert. Der Spawner musste erweitert werden, um Fahrzeuge und Fußgänger auf verschiedenen Straßen initialisieren zu können.
Um die Realitätstreue in einer dritten Version des SUC 1 weiter zu erhöhen, wurde die Komplexität des zu untersuchenden Szenarios erneut gesteigert (siehe Abbildung 151). Unter anderem wurde dabei die einfache X-Kreuzung durch eine komplexe X-Kreuzung ersetzt. Die folgenden wesentlichen Anpassungen am Szenario wurden durchgeführt, auch um zu zeigen, dass bei weiteren Komplexitätssteigerungen das Simulationssetup an diesen Stellen nicht limitiert war:

- Die Kreuzung im simulierten Szenario wurde von einer einfachen synthetischen Kreuzung ausgebaut auf eine synthetisch generierte Version der Forschungskreuzung in Braunschweig. Diese beinhaltete auf jedem Ast mehrere Spuren, die teilweise eine Richtung vorgaben, aber auch teilweise zwei Richtungen.
- Das zu simulierende Szenario sollte kein Rechtsabbieger-Szenario mehr sein, sondern ein Linksabbieger-Szenario. Dementsprechend kamen komplexere Vorfahrtsregeln zum Tragen.
- Die Fahrzeuge auf der Kreuzung sollten auch nicht mehr nur statische, schilder-basierte Vorfahrtsregeln berücksichtigen, sondern der Verkehrsfluss wurde vor allem über eine dynamische Ampelsteuerung orchestriert.
- Das Ego-Fahrzeug sollte weiterhin mit einer automatisierten Fahrfunktion unterwegs sein und dabei über eine Ampel nach links abbiegen. Dadurch musste das Ego-Fahrzeug auf den Gegenverkehr genauso achten wie auf querende Fußgänger.
- Die vier entgegenkommenden Fahrzeuge des Umgebungsverkehrs sollten Szenario-Fahrzeuge sein, die aktiv am Szenario beteiligt waren und teilweise geradeaus fuhren sowie teilweise nach rechts abbogen. Dadurch musste das Ego-Fahrzeug den Gegenverkehr passieren lassen und sich teilweise hinter den Rechtsabbieger einordnen.
- Insgesamt waren sieben Fußgänger im Szenario unterwegs, wovon zwei die Straße an der Kreuzung überquerten und es somit sowohl zur Interaktion Fußgänger / Ego-Fahrzeug als auch zur Interaktion Fußgänger / Szenario-Fahrzeug kommen konnte.
- Je nach gewürfeltem Szenario wurden dabei manche Szenario-Fahrzeuge oder auch Fußgänger verdeckt, so dass das Ego-Fahrzeug diese erst spät wahrnehmen konnte.
- Darüber hinaus überfuhren fünf weitere Fahrzeuge des Umgebungsverkehrs die Kreuzung, griffen aber auf Grund der Ampelschaltung nicht aktiv in das Szenario ein.

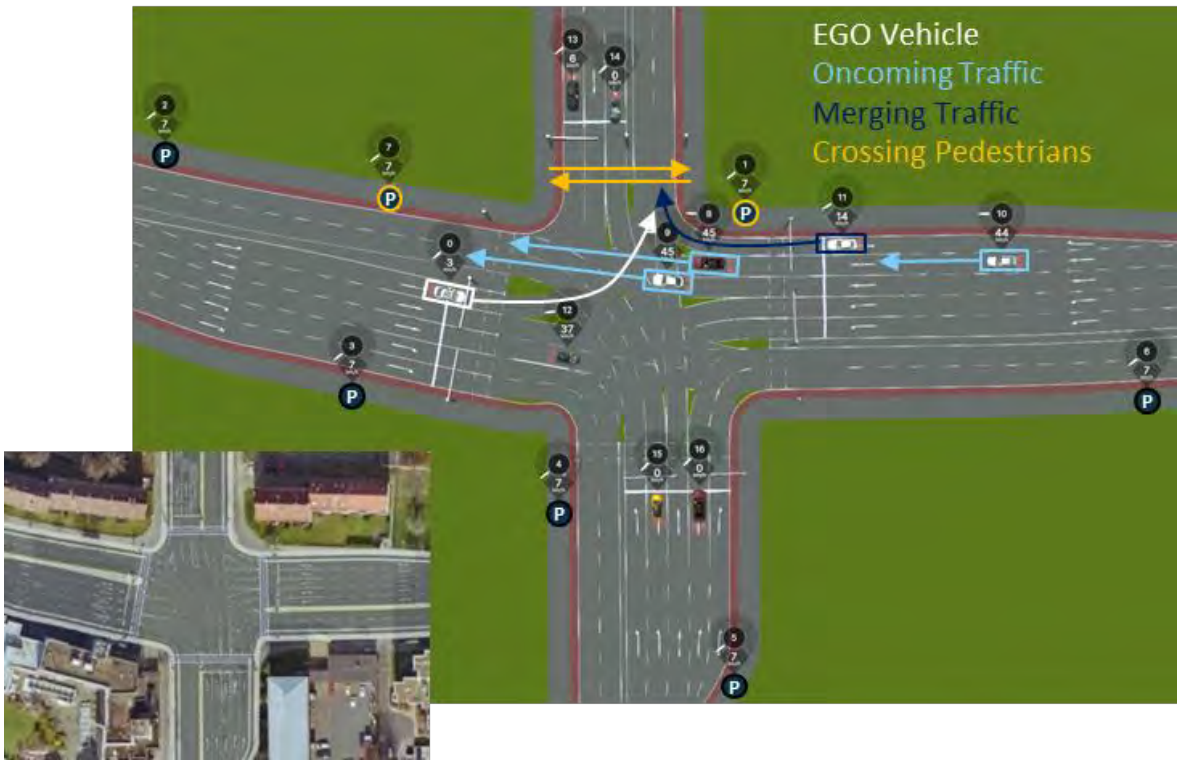


Abbildung 151: Systematische Darstellung SUC 1 – komplexe Kreuzung

Die finale Ausbaustufe des SUC 1 beinhaltete somit eine Vielzahl von unterschiedlichen Verkehrsteilnehmern, die in diversen Situationen aufeinander reagieren mussten (Fußgänger / Ego-Fahrzeug, Fußgänger / Szenario-Fahrzeug und Ego-Fahrzeug / Szenario-Fahrzeug). Zusätzlich konnten mit diesem Simulationssetup auch Performanceuntersuchungen durchgeführt werden, indem eine Vielzahl von unterschiedlichen Simulationsläufen analysiert worden sind. Darüber hinaus konnte damit auch sehr gut die Skalierbarkeit des Simulationsansatzes aufgezeigt werden.

2.3.3.2.2 Befähigung des Werkzeuges openPASS

Im Rahmen des Projekts stehen Ziele wie eine modulare Simulationsarchitektur, die einfache Integrier- und Austauschbarkeit von Modellen sowie die Skalierbarkeit der Simulationen im Zentrum. Die hierfür notwendigen Konzepte und technischen Umsetzungen wurden im Projekt entwickelt und im Rahmen der zuvor beschriebenen Szenarien erprobt. Die praxisnahen Beispiele stellen somit die Nutzbarkeit der konzeptionierten und realisierten Lösungen sicher. Die Zusammenarbeit vieler Partner, wie sie im Rahmen von Projekten für das automatisierte Fahren unumgänglich ist, erfordert ein einheitliches Verständnis der zugrundeliegenden Konzepte für die umzusetzenden Lösungen. Im Rahmen der Simulation betrifft dies neben der grundsätzlichen Architektur auch die Schnittstellen zwischen verschiedenen Tools und Komponenten. Darüber hinaus können zudem Open Source Lösungen einen weiteren Beitrag leisten, um die Kollaboration zwischen Partnern zu vereinfachen.
Für die beschriebenen Verkehrssimulationen wurde daher das Open Source verfügbare Simulationstool openPASS gewählt. Dieses wurde von Anfang an modular und mit offenen Schnittstellen entwickelt und stellte somit eine gute Ausgangsbasis für die Arbeiten im Rahmen des Projekts dar (siehe Abbildung 152). Dazu trug auch der bestehende Aufbau auf Simulationsstandards bei, deren Ausweitung und Nutzung im Folgenden beschrieben ist.

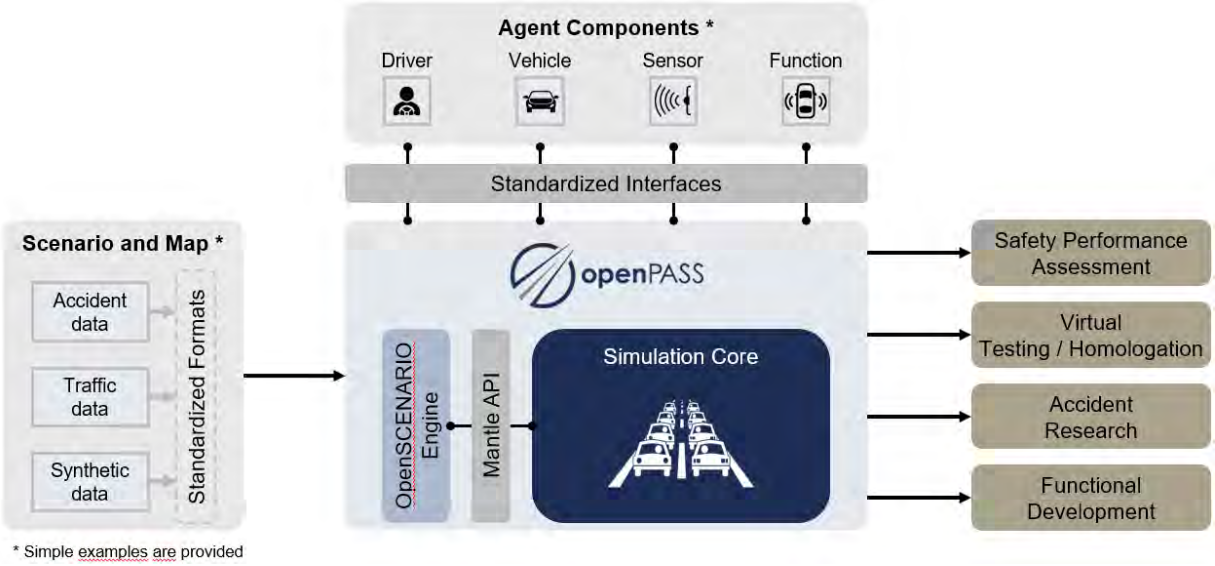


Abbildung 152: Modularer Architekturansatz von openPASS

Unterstützung für OpenDRIVE

Die Beschreibung der Karten erfolgte im Projekt gemäß dem OpenDRIVE Standard. Zu Beginn des Projektes basierten diese zunächst auf der Version 1.4. Auf dieser Basis wurde auch die einfache Kreuzung eingebunden. Da der Fokus in openPASS bisher vornehmlich auf Autobahnsimulationen lag und erst im Rahmen des Projekts auf den urbanen Raum gelenkt wurde, wurden u. a. neue LaneTypes für Parkplätze und Gehwege sowie Schilder für Vorfahrtsregelungen implementiert. Darüber hinaus wurde auch die Modellierung von Fußgängerüberwegen als Signale implementiert und weitere Features wie Stopplinien und erweiterte Fahrbahnmarkierungen befähigt.
Im weiteren Projektverlauf und im Rahmen des Umstiegs auf die komplexe Forschungskreuzung, wurde dann ein Upgrade der OpenDRIVE Version auf 1.6 umgesetzt. Durch die Rückwärtskompatibilität des Standards und der Implementierung in openPASS ist eine Simulation mit den vorhergehenden Versionen weiterhin möglich.
Im Zuge der Befähigung für die Forschungskreuzung wurde der Import von Ampeln (inkl. Fußgängerampeln) aus der Karte in openPASS implementiert.

Unterstützung für OpenSCENARIO

Die Modellierung der simulierten Szenarien erfolgte gemäß dem OpenSCENARIO Standard. Auf der Version 0.9 aufbauend wurden über die Projektlaufzeit auch die Versionen 1.0 und 1.1 vom ASAM inkl. der von SET Level beigesteuerten Inhalte veröffentlicht. Die Versionsupgrades wurden mit vielen zusätzlichen Features in openPASS befähigt.
Die Platzierung von Agenten (Spawning) in der Welt kann nun im OpenSCENARIO neben einer Position in Straßen-Koordinaten auch in absoluten Weltkoordinaten spezifiziert werden. Die in der ersten Ausbaustufe der Simulation genutzten Trajektorien wurden ebenfalls in das Szenario integriert. Das Einlesen der Trajektorien und Weiterleiten an die entsprechenden Agenten wurde implementiert. Dabei kann die Trajektorie sowohl im Szenario selbst definiert oder alternativ in einen TrajectoriesCatalog ausgelagert werden.
Im Zuge der Erweiterung auf die Version 1.1 des Standards wurde auch die Möglichkeit geschaffen, Parameter zu variieren und stochastische Verteilungen anzugeben. Ein entsprechender Ansatz aus PEGASUS wurde dazu zuvor in enger Abstimmung zwischen TP 4 und TP 2 überarbeitet, erweitert und anschließend über den ASAM im Standard veröffentlicht.

Hierbei wurden auch bisherige proprietäre Lösungen in openPASS berücksichtigt und abgelöst (z. B. Stochastische Variation der Spawngeschwindigkeit).
Das Laden und Interpretieren von Szenarien wurde im Rahmen des Projekts in eine eigenständige Library (OpenSCENARIO Engine) ausgelagert. Diese Implementierung erlaubt die Integration der OpenSCENARIO Engine in beliebigen Umgebungssimulatoren, analog zu openPASS. Die Anbindung erfolgt über eine zentrale Schnittstelle (Mantle API). Letztere erlaubt auch die Implementierung weiterer Szenario-Engines auf Basis alternativer Szenariobeschreibungsformate. Der modulare Ansatz erlaubt die weitreichende Nutzung und einfache Integrierbarkeit in beliebige Toolchains. Da die Umsetzung einem kontinuierlichen Prozess entspricht, werden die Aktivitäten auch nach der Projektlaufzeit weitergeführt und sind im Eclipse openPASS Projekt Open Source verfügbar.

Ertüchtigung Ground Truth-API (virtuelle Weltrepräsentation)
Erweiterung im Support für die offenen Simulationsstandards sowie die vielseitigen Agentenkonfigurationen machten auch diverse Erweiterungen in der Weltrepräsentation von openPASS erforderlich. Insbesondere ein einheitliches Vorgehen in der Vergabe von ID’s für Agenten, Objekte und jegliche Elemente aus der Karte wurde implementiert. Dies ermöglichte insbesondere eine bessere Durchgängigkeit zwischen den internen Datenstrukturen im Simulationskern und ausgeleiteten OSI-Nachrichten. Erweiterte und verbesserte Logausgaben aus der Simulation bringen einen direkten Mehrwert auch für die Nutzer von openPASS.

Erweiterung für automatisiertes Spawnen
Die Simulation auf einer Kreuzung mit unterschiedlichen Verkehrsteilnehmern machten auch einige Änderungen am Spawner erforderlich. Die vorhandenen Spawner (Szenario, PreRun und Runtime) mussten angepasst werden, damit nicht nur eine Straße bespawnt werden kann, sondern sämtliche Äste der Kreuzung. Dabei initialisiert der PreRun-Spawner die Agenten vor dem Simulationslauf randomisiert auf allen Ästen der Kreuzung; der Runtime-Spawner hingegen setzt die Agenten während der Simulation an vorher definierten Punkten (Spawnpunkte) in die Welt. Sollen die Fahrzeuge zu Beginn der Simulation an fest definierte Positionen gesetzt werden, kommt hierbei der Szenario-Spawner zum Einsatz. Zu beachten ist dabei bei allen Typen, dass die Agenten auf der passenden Seite in der korrekten Richtung gespawnt werden. Dabei dürfen keine Agenten auf dem Gehweg oder sonstigen nicht befahrbaren Spuren initialisiert werden. Im Gegensatz dazu muss der Spawner für die Fußgänger diese auf dem Gehweg platzieren, damit das Fußgängermodell sich auch korrekt in der Welt bewegen kann.

Unterstützung für FMI und OSI gemäß OSMP zur Modellanbindung
Die Austauschbarkeit und einfache Integration von Komponenten wird spätestens bei Simulationsmodellen zur Herausforderung, da diese regelmäßig von unterschiedlichsten Parteien entwickelt werden und gemeinsam zur Ausführung gebracht werden müssen.
Die Lösung bieten auch hier offene Standards. Dabei wurde eine Kombination zweier Standards – einer als Programming Interface, der andere für den Datenaustausch – herangezogen. FMI 2.0 definiert die Co-Simulations-API zum Initialisieren, Triggern und Beenden der Simulationsmodelle. Der Datenaustausch wurde auf Basis von Open Simulation Interface (OSI) realisiert. In OSI werden Messages befüllt und serialisiert. Die entsprechende Speicheradresse zur serialisierten Nachricht wird dann als Pointer über die FMI Schnittstelle an das Simulationsmodell übergeben, welches die Nachricht auslesen kann. Ebenso erfolgt eine Nachrichtenübermittlung vom Simulationsmodell zurück an openPASS. Dieses Vorgehen ist als Open Sensor Model Packaging (OSMP) Konvention bekannt.

Die im Zuge des Projekts für die Verkehrssimulation genutzten OSI-Nachrichten waren:

- OSI::SensorView
- OSI::SensorViewConfiguration
- OSI::SensorData
- OSI::GroundTruthInit
- OSI::TrafficCommand
- OSI::TrafficUpdate
- OSI::HostVehicleData
- SL45::MotionCommand
- SL45::DynamicsRequest
- SL45::VehicleCommunicationData

Nachrichten mit dem Präfix „SL45“ waren SET Level spezifische Nachrichten, die zwischenzeitlich genutzt wurden, aber zum Projektende durch standardisierte Nachrichten ersetzt werden konnten.
Zur Unterstützung der Nachrichten waren umfangreiche Erweiterungen in openPASS erforderlich. Diese umfassen insbesondere das Befüllen der verschiedenen Nachrichten und die Bereitstellung für FMUs. Dazu wurde neben der GroundTruth auch der FmuWrapper in openPASS ausgebaut. Die funktionalen Erweiterungen ergeben sich direkt aus den im Nachfolgenden beschriebenen Modellanbindungen und den genutzten Funktionalitäten.

Statisches Linken in Modellen
Die beschriebene Anbindung von Simulationsmodellen setzt das Linken gegen OSI und Protobuf in den Modellen voraus. In openPASS sind diese Abhängigkeiten ebenfalls gelinkt. Werden die Abhängigkeiten dynamisch gelinkt, kann es zur Runtime zu Fehlern beim Laden der Libraries kommen. Um dieses Problem zu umgehen und unterschiedliche Versionen der Abhängigkeiten in den verschiedenen Modellen zuzulassen, wurde das statische Linken für OSI und Protobuf aufgesetzt. Die aufgesetzte CMakeLists und das Vorgehen wurde den Modellentwicklern in TP 3 als Vorlage bereitgestellt. Durch dieses Vorgehen wurden auch Versionsabhängigkeiten unter den verschiedenen Modellen aufgelöst, da nun durch die Rückwärtskompatibilität von OSI und Protobuf auch unterschiedlichste Versionen in den Modellen und openPASS genutzt werden können.

Unterstützung für SSP
Wird ein Verkehrsteilnehmer nicht nur durch ein einzelnes Simulationsmodell modelliert, sondern kommt eine Verschaltung dieser zum Einsatz, so muss die Verschaltung konfiguriert werden. Bislang werden vor allem proprietäre Lösungen genutzt, welche eine Austauschbarkeit erschweren. Für die Befähigung in openPASS wurde daher der Standard System Structure and Parameterization (SSP) ausgewählt, um wiederum eine offene und erweiterbare Lösung zu schaffen.
Der SSP-Standard bietet eine Möglichkeit, mehrere FMUs (auch OSMP FMUs) zu verschalten und in die Simulation einzubinden. Neben dieser Möglichkeit zur Verschaltung mehrere FMUs bietet der Standard auch die Parametrierung der einzelnen Modelle an. Hierzu wurde ein Konzept erarbeitet, wie der SSP-Standard in der openPASS-Simulation angewendet werden kann. Es konnten wichtige Erkenntnisse hinsichtlich des Umgangs und der Austauschbarkeit mit Modellen und SSP-Spezifikationen gesammelt werden (z. B. Auslagerung von Parameterdefinitionen, Konvention zur Einbettung von OSI-Konnektoren). Im projektweiten Austausch zu SSP sind zudem Unklarheiten in der Interpretation ausdefiniert worden und eine

konsolidierte Vorgehensweise zum Aufbau und der Nutzung von SSP entwickelt. Die Implementierung des SSP-Wrappers in openPASS erfolgte im Anschluss.
Darüber hinaus wurde TP übergreifend eine Annotation zum SSP-Standard entwickelt, welche die Spezifikation der Ausführungsreihenfolge im SSP zulässt. Bisher war dies zufällig bzw. toolspezifisch gelöst. Die Umsetzung der Annotation in openPASS und Einführung in der Verkehrssimulation hat stattgefunden und wurde erfolgreich getestet. Die Anwendung des SSP-Standards ist im Rahmen des Ego-Fahrzeugs erfolgt und im späteren Verlauf des Berichts beschrieben.

Modellanbindung
Über die verschiedenen Szenarien hinweg wurden unterschiedlichste Modelle angebunden und simuliert. Das einfachste Modell zu einem frühen Zeitpunkt war ein Trajektorienfolger. Dieser war bereits als OSMP FMU implementiert und konnte so leicht angebunden werden. Die Trajektorie wurde aus dem Szenario eingelesen und mittels OSI::TrafficCommand an das Modell übermittelt. Das Modell führte eine Interpolation zwischen den Stützstellen durch und konnte so in beliebigen Schrittweiten simuliert werden. Durch zweifache Instanziierung des Modells wurden zwei Agenten auf der einfachen Kreuzung simuliert.
Um das Verhalten realistischer zu machen und Interaktionen zu ermöglichen, hat im folgenden Schritt die Integration des ika-Drivers stattgefunden. Dieser nimmt seine Umwelt über eine OSI::SensorView wahr und kann über OSI::TrafficCommand mit einer Zielposition, Bewegungsgeschwindigkeit und anderen Verhaltensanweisungen bedatet werden. Die entsprechenden Anweisungen werden von openPASS aus dem Szenario in den Traffic Command übersetzt. Der Output wird vom Modell als OSI::TrafficUpdate bereitgestellt und an openPASS übermittelt. Letztlich findet ein Update des Agenten im World State statt.
Analog zum ika-Driver wurde auch der OSI-Fußgänger eingebunden. Dieses verfügt über dieselben Schnittstellen, modelliert allerdings das Verhalten eines Fußgängers.
Beide Modelle wurden in TP 3 entwickelt und über die gesamte Projektlaufzeit erweitert. Die Anforderungen kamen dabei aus den zu simulierenden Szenarien. Kurze Feedbackzyklen und regelmäßige Updates der eingebundenen Versionen erlaubten eine agile Weiterentwicklung. Schlussendlich verfügten beide Modelle über Pfadplanungsalgorithmen, die Fähigkeit zu Kollisionsvermeidung sowie Beachtung von Verkehrsregeln und viele weitere Funktionalitäten. Auch die Interaktion untereinander wurden immer realistischer, sodass realitätsnahe Bewertungen auf Basis der Simulationsergebnisse möglich sind.
Um auch die Integration komplexerer Agenten zu erproben, wurde ein automatisiertes Fahrzeug bestehend aus vier Simulationsmodellen (Kamerasensor, HAD-Funktion, MotionControl, VehicleDynamics) zusammengesetzt. Jedes der Modelle wurde als OSMP FMU gebaut und integriert. Die Verschaltung erfolgte auf Basis der zuvor beschriebenen SSP-Standards. Die Kommunikation zwischen den Modellen erfolgte abermals über OSI-Messages.
Eine Besonderheit stellt die HAD-Funktion dar, welche als ROS-Knoten implementiert ist und in einem Docker-Container läuft. Die Bedatung erfolgt dabei über TCP/IP. Die Brücke zwischen openPASS und dem Docker Container übernahm ein als OSMP FMU implementiertes TCP/IP-Gateway.
Die Motion Control und Vehicle Dynamics Modelle wurden für eine detailliertere Abbildung eines realen Fahrzeugs benötigt und simulieren Regelverhalten sowie Fahrdynamik. Um die in den Modellierungsarbeitspaketen in der Systementwicklungsumgebung Simulink® erstellten Fahrzeugmodelle unterschiedlicher Detaillierungsstufen in openPASS nutzen zu können wurden diese in OSI-kompatible Schnittstellen gekapselt. Dazu wurde eine OSI-Schnittstelle für Simulink erweitert. Die Schnittstellen wurden an den Stand von OSI 3.3.3 und den Bedarf für die Modelle aus den AP 3.1 angepasst. Die Interfaces wurden als C++ Code implementiert

und in kompilierter Form als Simulink-Modellbibliothek (siehe Abbildung 153) bereitgestellt, was die Einbindung der Schnittstellen für den Modellentwickler erleichtert. Die Bibliothek enthält die folgenden Schnittstellen:

- *osi_write_host_vehicle_data*
- *osi_read_host_vehicle_data*
- *osi_write_traffic_update*
- *osi_read_traffic_update*
- *osi_read_ground_truth*
- *osi_write_ground_truth*
- *sl45_read_dynamics_request*
- *sl45_write_dynamics_request*
- *sl45_read_motion_command*
- *sl45_write_motion_command*

Zusätzlich enthält die Bibliothek die zwei weiteren Blöcke *osi_write_from_trace* und *osi_simulation_clock* als Hilfsfunktionen. Mit *osi_write_from_trace* können Open-loop Tests von Modellen mithilfe zuvor generierter OSI Trace Dateien ausgeführt werden. Die *osi_simulation_clock* stellt ein Zeitsignal im OSI-Format bestehend aus vollen Sekunden als int32 und Nanosekunden als uint32-Werte zur Verfügung. Diese werden standardmäßig als Zeitstempel in gesendeten OSI-Nachrichten verwendet. Zusätzlich werden die Inhalte der OSI-Versionsnachrichten in den Blockmasken gekapselt und damit die sichtbaren Teile der Schnittstellen auf die modellseitig relevanten Signalgrößen reduziert. Dies erhöht die Übersichtlichkeit der Modelle und erleichtert die Implementierung der Fahrzeugmodelle. Die Schnittstelle für Nachrichten des OSI-Typs MotionCommand wurde erweitert, um Trajektorien mit mehreren Punkten transportieren zu können. Dies ermöglicht eine asynchrone Ausführung der unterschiedlichen Ebenen der HAD-Funktion, indem die Funktionen der Bewegungssteuerung (Motion control) und der Aktorsteuerung (Actuator control) jeweils innerhalb der von der vorgelagerten Trajektorien Planung (Trajectory planning) generierten Trajektorie die für ihren Ausführungstakt erforderlichen Zielpunkte interpolieren kann:

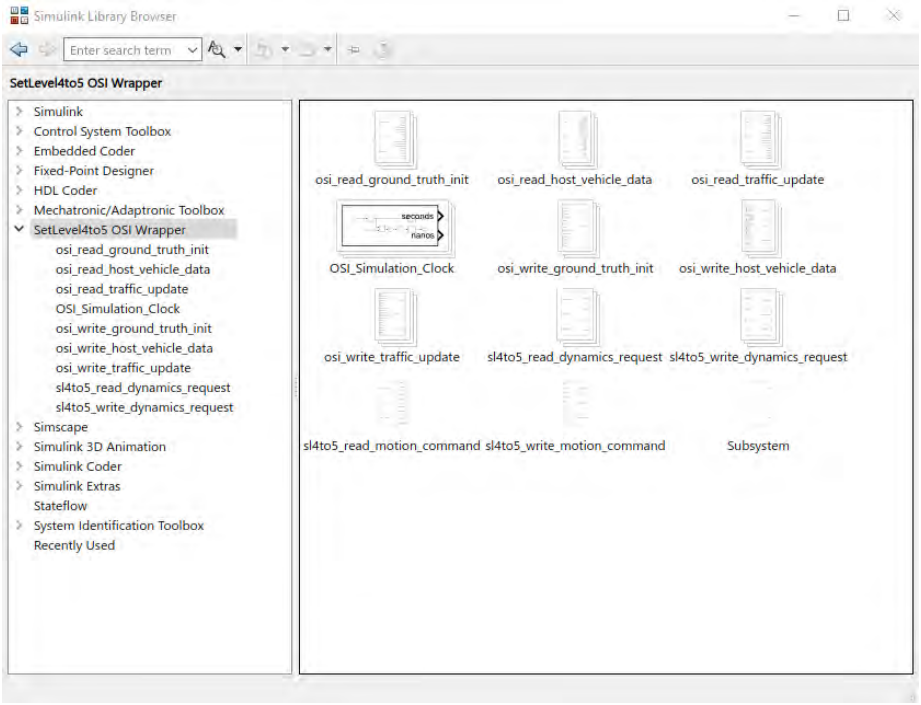


Abbildung 153: Simulink-Library mit Interfaceblöcken für Fahrzeugschnittstellen

Um die Modelle sowohl unter Linux als auch unter Windows zu kompilieren wurde der Build über die FMI Toolbox aufgesetzt. Die entstehenden Modelle konnten analog zu den sonstigen Simulationsmodellen eingebunden werden.
Im beschriebenen Aufbau war das automatisierte Fahrzeug in der Lage, Ampeln zu erkennen und die Vorfahrtsregeln beim Linksabbiegen einzuhalten.
Zusammenfassend zeigt sich, dass eine Vielzahl von verschiedenen Modellen an openPASS angebunden wurde. Das zugrundeliegende Prinzip hierfür lieferten in jedem Fall die Standards FMI und OSI gemäß OSMP Konvention. Obwohl sich die Modelle funktional stark unterscheiden – von Sensormodellen bis Agentenmodellen – lässt sich das Prinzip ausnahmslos anwenden. Unterschiede bestehen lediglich in den konkret verwendeten OSI-Nachrichten für den Informationsaustausch.

Implementierung von Observern

Um neben dem von openPASS ohnehin geschriebenen Simulationsoutput weitere Daten aus der Simulation auswerten und loggen zu können, wurden Observer für die Time-To-Collision (TTC) und die Post-Encroachment-Time (PET) implementiert. Alle Observer sind als C++ Modelle gemäß OSMP Konvention definiert und wurden analog zu den Simulationsmodellen in openPASS eingebunden. Auch hier wurde das statische Linken von OSI und Protobuf angewendet. Die Informationen werden per OSI::SensorView an die Observer gesendet.
Ein einfacher TTC-Observer berechnet die TTC auf Basis des longitudinalen Abstands sowie der Differenzgeschwindigkeit. Da in urbanen Szenarien mit Krümmungen und seitlichen Versätzen diese Größe nicht besonders aussagekräftig ist, wurde ein zweiter TTC-Observer mit Prädiktion implementiert. Dabei wird neben der Längsgeschwindigkeit auch die Quergeschwindigkeit, Beschleunigungen, Gierrate und Gierbeschleunigung für die vorausschauende Betrachtung berücksichtigt.
PET-Observer schreibt alle für die Berechnung der PET erforderlichen Größen direkt aus. Die PET wird dann im Post-Processing der Simulation berechnet, da diese nur rückwirkend bestimmt werden kann.
Alle Observer loggen die aggregierten Daten in Dateien.

Zusammenfassung
Die vollständige Verkehrssimulation zum Projektende lässt sich anhand der Architekturdarstellung in Abbildung 154 darstellen.

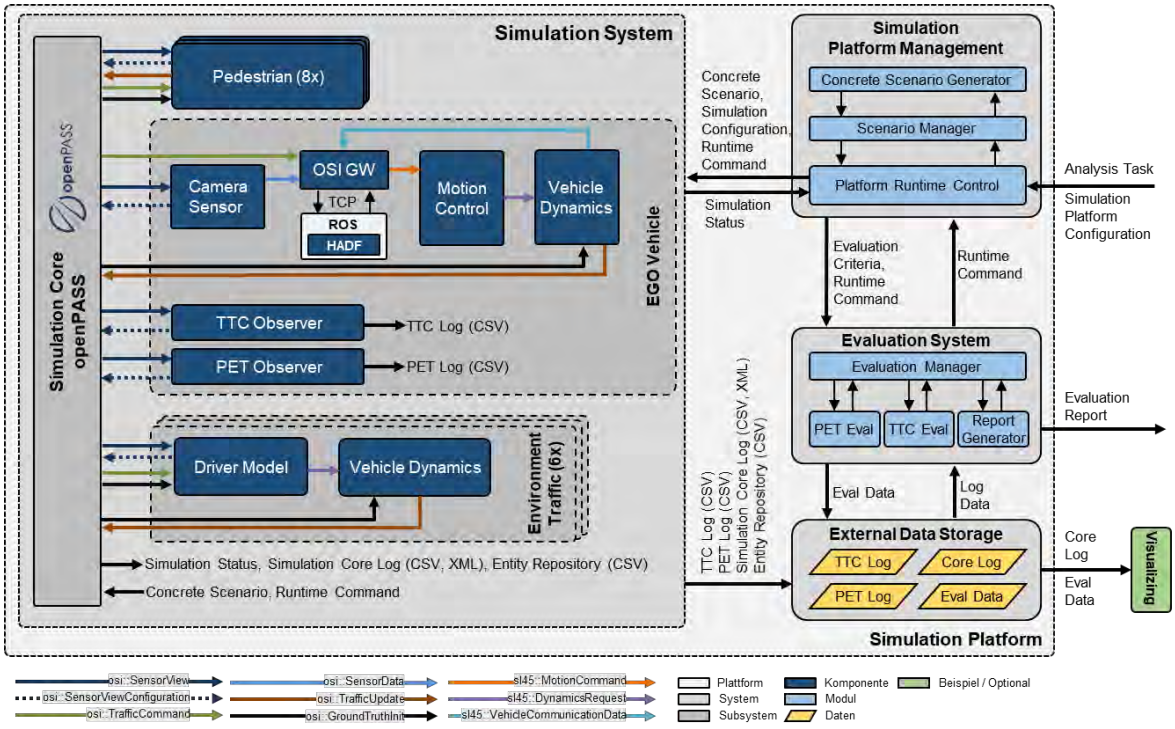


Abbildung 154: Architektur der finalen Ausbaustufe der Verkehrssimulation

Bei den schlussendlich genutzten Standards (siehe Abbildung 155) und den entsprechenden Versionen handelt es sich um:

- OpenSCENARIO 1.1
- OpenDRIVE 1.6
- OSI 3.3.3 (Open Simulation Interface)
- FMI 2.0 (Functional Mock-Up Interface)
- SSP 1.0 (System Structure and Parametrization)

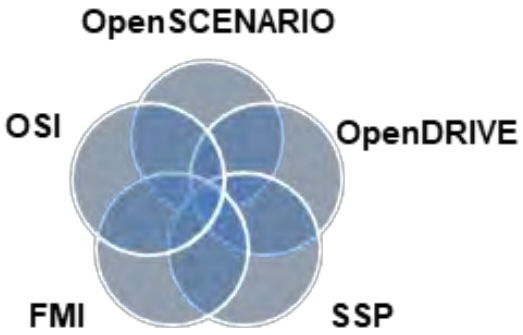


Abbildung 155: Genutzte Standards in der Verkehrssimulation

Mit dem beschriebenen Aufbau ist die Durchführung der obigen Szenarien möglich. Zur Durchführung einer Kritikalitätsanalyse ist darüber hinaus die Einbettung des Gesamtaufbaus

in eine Simulationsplattform zur Parametervariation sowie Simulationsauswertung notwendig. Dieser Schritt wird im folgenden Abschnitt beschrieben.

2.3.3.2.3 Aufbau und Integration Simulationsplattform

Aufgrund der fortschreitenden Entwicklung innerhalb des Projektes wurden mehrere Simulationsaufbauten im Rahmen des AP 4.2 umgesetzt und damit zur Demonstration der Anpassbarkeit der Plattform verschiedene Phänomene untersucht. Hierbei sind insbesondere die Simulationen einer einfachen Kreuzung mit einer Kritikalitätsanalyse rund um die statische Verdeckung und die Simulation einer komplexen Kreuzung mit der Analyse des Linksabbiegens mit Gegenverkehr zu nennen, die im Folgenden beschrieben und anschließend die jeweilige exemplarische Analyse vorgestellt werden soll.

Simulationsplattform für die Analyse der statischen Verdeckung
Die koordinierende Struktur der Simulationsplattform, die Komponente „Simulation Platform Management“, wurde für die Analyse der statischen Verdeckung in 3 Hauptkomponenten gegliedert:

- Platform Runtime Control
- Scenario Manager
- Concrete Scenario Generator

Ihre jeweilige Funktionsweise wird im Folgenden dargestellt.
Die Platform Runtime Control nimmt den Analysetask und die Plattformkonfiguration (jeweils als TOML File) entgegen und koordiniert auf Grundlage der enthaltenen Informationen die Ausführung der Simulation. Dabei wird zuerst der Scenario Manager getriggert, um die Szenariogenerierung durchzuführen. Der Scenario Manager erhält von der Platform Runtime Control die aus dem Analysis Task und der Simulation Platform Configuration extrahierten Informationen, um aus einem logischen Szenario einen Satz von konkreten Szenarien erstellen lassen zu können. Mit diesen Informationen steuert der Scenario Manager den Concrete Scenario Generator an, welcher basierend auf den definierten Variationsmechanismen und der Parameterverteilung dann letztendlich die konkreten Szenarien erzeugt. Im Anschluss werden die erstellten Szenarien nacheinander abgearbeitet. Dafür konfiguriert die Platform Runtime Control den Simulationskern openPASS und übergibt die zur Simulationsdurchführung notwendigen Daten. Nach jedem Simulationslauf werden die erzeugten Simulationsdaten entsprechend der Konfiguration im Analysis Task und der Simulation Platform Configuration im External Data Storage abgelegt. Nachdem alle Szenarien simuliert wurden, triggert die Platform Runtime Control den Evaluation Manager und startet somit die Evaluationsdurchführung. Nach Erfolgreicher Evaluation beendet die Platform Runtime Control die Ausführung.
Nach der Simulationsdurchführung erfolgt die Evaluation der Simulationsergebnisse durch das Evaluationssystem. Der Evaluation Manager steuert dabei die Evaluationsdurchführung und bekommt alle notwendigen Informationen aus dem Analysis Task und der Simulation Platform Configuration von der Platform Runtime Control übergeben. Dies sind u. a. die Evaluationskriterien und Reportinformationen, welche von den Evaluationsmodulen und dem Reportgenerator benötigt werden. Die PET Evaluation und die TTC Evaluation werden dann nacheinander mit den entsprechenden Evaluationskriterien aufgerufen.
Das PET Evaluationsmodul PET Eval berechnet auf Grundlage aller im Dateiordner gefundenen Logfiles die PET der Simulationsläufe. Des Weiteren berechnet das Modul verschiedene Aggregatfunktionen auf der Liste der errechneten PET-Werte. Hierfür werden dem PET Evaluationsmodul für die zwei zu berücksichtigenden Agenten jeweils die IDs und die Bounding Boxen sowie der Pfad zum Ordner übergeben, auf dem die Analyse durchgeführt werden soll.

Als Ausgabe erzeugt das PET Evaluationsmodul ein um die PET angereichertes CSV Logfile sowie einen Report im JSON Format, welches nach Hinzufügen des Headers durch den Evaluation Manager vom Report Generator verarbeitet werden kann.
Das TTC Evaluationsmodul TTC Eval wertet dagegen alle Simulationsergebnisse des TTC online Observers aus die im Dateiordner der Simulationsergebnisse gefunden werden. Des Weiteren werden verschiedene Aggregatfunktionen umgesetzt. Hierfür werden dem TTC Eval Modul für die zu berücksichtigenden Agenten jeweils die IDs sowie der Pfad zum Ordner übergeben, auf dem die Analyse durchgeführt werden soll. Als Ausgabe erzeugt das TTC Eval Modul ein um die TTC angereichertes CSV Logfile sowie ebenfalls einen Report im JSON Format.
Der Report Generator wird letztendlich vom Evaluation Manager aufgerufen und erhält von diesem einen aggregierten JSON String in dem die relevanten Informationen für den Evaluationsbericht (Evaluationsergebnisse, Statusinformationen) festgehalten sind. Anschließend wird der Report im angegebenen Format erstellt und im spezifizierten Ausgabeverzeichnis abgelegt.
Für diese Ausrichtung der Simulationsplattform wurde der External Data Storage durch einen Ordner im Dateisystem realisiert. Die Simulations- und Evaluationsergebnisse sowie die für eine Auswertung und spätere Reproduzierbarkeit benötigten Daten werden pro Analyse in Ordnern, die mit einem Timestamp versehen sind, abgelegt. Welche Daten abgelegt werden sollen kann im Analyse Task und in der Plattformkonfiguration konfiguriert werden.
Beispiele für den Analyse Task und die Plattformkonfiguration, sowie ihr Einfluss auf die Konfiguration der Simulationsplattform sind in Abbildung 156 dargestellt und wurde in beiden hier behandelten Analysen über TOML-Dateien realisiert. Die Konfigurationsmöglichkeiten umfassen hierbei unter anderem die genutzten Evaluationsmodule, Speicherorte für die relevanten Szenarien, das Ausgabeformat des Analyseberichts sowie die Art, den Analysebericht abzulegen (z. B. je Metrik ein Bericht, je Paar von Akteuren ein Bericht).

Simulationsplattform für die Analyse des Linksabbiegens mit Gegenverkehr
Die bereits zuvor beschriebene Simulation Platform Management, bestehend aus den drei Komponenten Platform Runtime Control, Scenario Manager und Concrete Szenario Generator, wurde für diese im Projektverlauf nachgelagerte Analyse durch die Erkenntnisse aus der Analyse der statischen Verdeckung modifizierten Anforderungen angepasst.
Die Platform Runtime Control nimmt weiterhin den Analysetask und die Plattformkonfiguration (jeweils als TOML File) entgegen und koordiniert auf Grundlage der erhaltenen Informationen die Ausführung der Simulation. Neu ist hierbei als Erkenntnis aus der vorangegangenen Analyse die Verschiebung der Parameter und der Anzahl der durchzuführenden Simulationsläufe aus dem Analysetask direkt in die OpenSCENARIO-Datei, um die Konformität zum kurz zuvor veröffentlichten OpenSCENARIO 1.1 Standard sicherzustellen.
Durch die Platform Runtime Control wird zuerst der Scenario Manager getriggert, um die Szenariogenerierung durchzuführen. Der Scenario Manager erhält von der Platform Runtime Control die aus dem Analysis Task und der Simulation Platform Configuration extrahierten Informationen, um aus einem logischen Szenario einen Satz von konkreten Szenarien erstellen lassen zu können.
Mit diesen Informationen steuert der Scenario Manager den Concrete Scenario Generator an, welcher basierend auf den definierten Variationsmechanismen und der Parameterverteilung dann letztendlich die konkreten Szenarien erzeugt. An dieser Stelle war es für die angepasste Simulationsplattform notwendig, eine neue Version des Parametervariationsskripts von dSPACE einzubinden, um auch an dieser Stelle die Konformität zum OpenSCENARIO 1.1 Standard zu gewährleisten.

Anschließend steuert die Platform Runtime Control die Durchführung der Simulation der generierten konkreten Szenarien. Die dabei größte Erweiterung zur Analyse des Linksabbiegens bei Gegenverkehr ist hierbei die Befähigung, die Simulationsläufe nicht nur sequenziell, sondern ebenfalls parallel ausführen zu können. Dies war bei der zuvor durchgeführten Analyse der statischen Verdeckung als einschränkend aufgefallen, da eine repräsentative Analyse aller kritischen Szenarioklassen eine große Anzahl an simulierten Szenariorealisierungen voraussetzte. Eine detailliertere Ausführung hierzu findet sich im nachfolgenden Abschnitt zur Durchführung der Analyse.
Für jeden einzelnen Lauf konfiguriert die Platform Runtime Control den Simulationskern openPASS und übergibt die zur Simulationsdurchführung notwendigen Daten. Falls die parallele Ausführung genutzt wird, werden an dieser Stelle Docker-Container für die openPASS Instanzen erzeugt und Docker-Netzwerke aufgesetzt, über die die Zuordnung von openPASS und HAD-Funktion stattfindet. Weiterhin wird in den Docker-Containern ein Python-Skript abgelegt, dass den jeweiligen Simulationslauf im Detail koordiniert. Nach jedem Simulationslauf werden die erzeugten Simulationsdaten aus dem Docker-Container extrahiert und entsprechend der Konfiguration im Analysis Task und der Simulation Platform Configuration im External Data Storage abgelegt. Nachdem alle Szenarien simuliert wurden, triggert die Platform Runtime Control den Evaluation Manager und startet somit die Evaluationsdurchführung. Nach erfolgreicher Evaluation beendet die Platform Runtime Control die Ausführung:

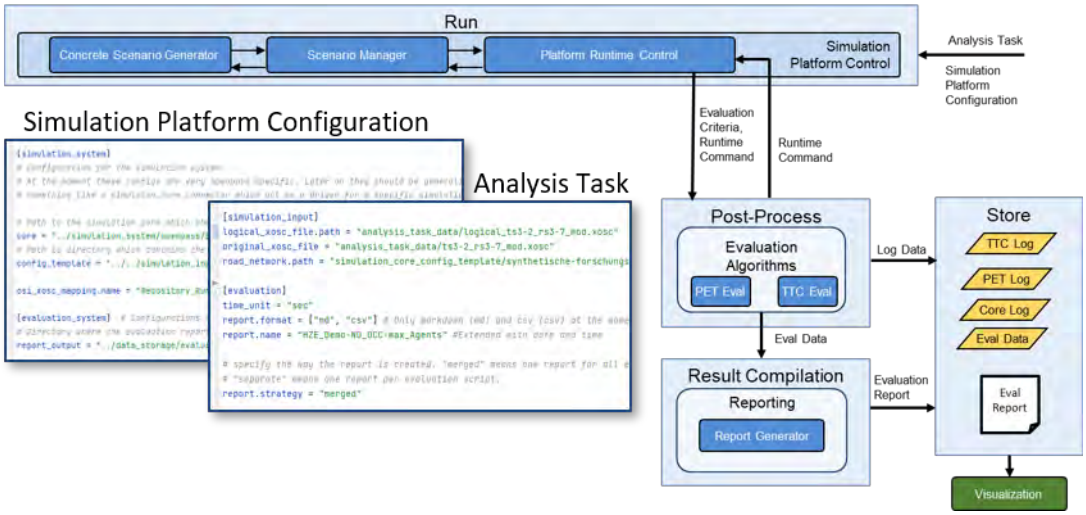


Abbildung 156: Teilstück der modularen Simulationsplattformarchitektur zur Darstellung der Konfiguration durch Simulation Platform Configuration und Analysis Task

Aufgrund der Modularität der Simulationsplattform musste für die Integration und Veränderung der übrigen Komponenten zwischen den Simulationen zur Kritikalitätsanalyse keine Anpassung am Evaluationssystem durchgeführt werden. Jedoch haben einige inhaltliche Anpassungen stattgefunden, um unter anderem die Präzision mit denen die Kritikalitätsmetriken die tatsächliche Kritikalität schätzen zu verbessern.
Daher erfolgt nach der Simulationsdurchführung wie bei der Simulation der einfachen Kreuzung die Evaluation der Simulationsergebnisse durch das Evaluationssystem. Der Evaluation Manager steuert dabei die Evaluationsdurchführung und bekommt alle notwendigen Informationen aus dem Analysis Task und der Simulation Platform Configuration von der Platform Runtime Control übergeben. Dies sind u. a. die Evaluationskriterien und Reportinformationen, welche von den Evaluationsmodulen und dem Reportgenerator benötigt werden. Die PET Evaluation und die TTC Evaluation werden dann nacheinander mit den entsprechenden Evaluationskriterien aufgerufen.

Das PET Evaluationsmodul PET Eval berechnet auf Grundlage aller im Dateiordner gefundenen Logfiles die PET der Simulationsläufe. Des Weiteren berechnet das Modul verschiedene Aggregatfunktionen auf der Liste der errechneten PET-Werte. Hierfür werden dem PET Evaluationsmodul für die zwei zu berücksichtigenden Agenten jeweils die IDs und die Bounding Boxen sowie der Pfad zum Ordner übergeben, auf dem die Analyse durchgeführt werden soll. Als Ausgabe erzeugt das PET Evaluationsmodul ein um die PET angereichertes CSV Logfile sowie einen Report im JSON Format, welches nach Hinzufügen des Headers durch den Evaluation Manager vom Report Generator verarbeitet werden kann.
Das TTC Evaluationsmodul TTC Eval wertet dagegen alle Simulationsergebnisse des bereits während der Simulation mitlaufenden und berechnenden TTC Observers aus, die im Dateiordner der Simulationsergebnisse gefunden werden. Des Weiteren werden verschiedene Aggregatfunktionen umgesetzt. Hierfür werden dem TTC Evaluationsmodul für die zu berücksichtigenden Agenten jeweils die IDs sowie der Pfad zum Ordner übergeben, auf dem die Analyse durchgeführt werden soll. Als Ausgabe erzeugt das TTC Evaluationsmodul ein um die TTC angereichertes CSV Logfile sowie ebenfalls einen Report im JSON Format.
Der Report Generator wird letztendlich vom Evaluation Manager aufgerufen und erhält von diesem einen aggregierten JSON String in dem die relevanten Informationen für den Evaluationsbericht (Evaluationsergebnisse, Statusinformationen) festgehalten sind. Anschließend wird der Report im angegebenen Format erstellt und im spezifizierten Ausgabeverzeichnis abgelegt.

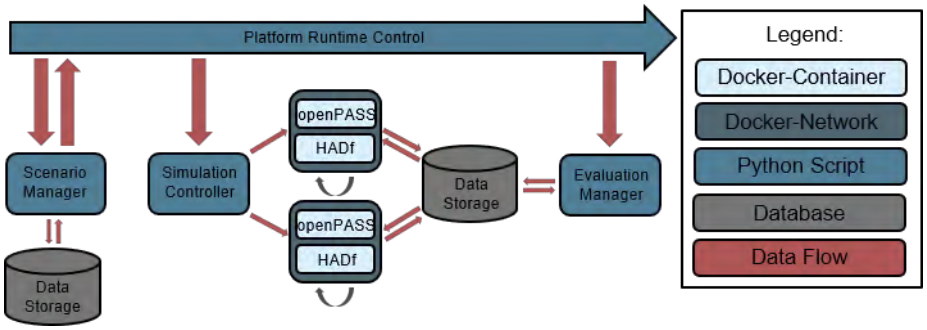


Abbildung 157: Ausschnitt aus der Anstoßreihenfolge der Platform Runtime Control

Wie bereits bei der Beschreibung der Platform Runtime Control erwähnt, wurde die sequenzielle Ausführung der einzelnen Simulationsläufe aufgrund der erkannten hohen Anzahl notwendiger Simulationsläufe parallelisiert, wodurch die bisher bereits sehr effizient ausgelegte Simulation die vollständigen Ressourcen der Simulationsrechner ausnutzen kann.
In Abbildung 157 ist ein Ausschnitt der Ausführungsreihenfolge der Simulationsplattform dargestellt, um die Steuerung der für die Simulation genutzten Docker-Container zu veranschaulichen. Der hierbei dargestellte Simulation Controller ist eine Komponente der Platform Runtime Control und kein eigenes Modul der Simulationsplattform, wurde jedoch zur Verdeutlichung der Ablaufreihenfolge und der Ansteuerung der Docker-Netzwerke ausgegliedert. Jedes Docker-Netzwerk besteht hierbei jeweils aus einer openPASS-Instanz und einer HAD-Funktionsinstanz, die sich über TCP verbinden.

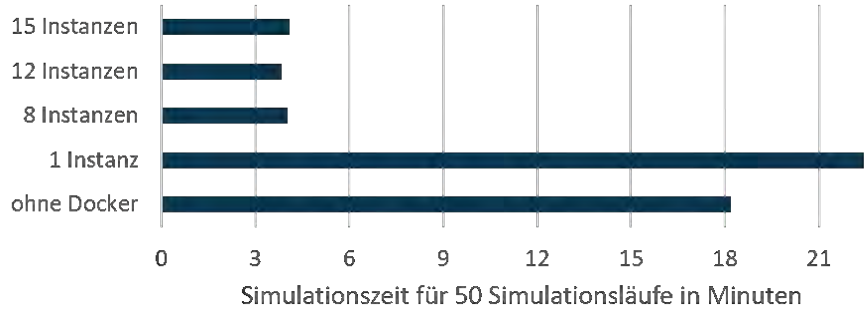


Abbildung 158: Darstellung der Performanzverbesserung der Gesamtsimulation durch Parallelisierung der Simulations-instanzen innerhalb der Simulationsplattform

Zum Vergleich der Ausführungszeiten (siehe Abbildung 158) zwischen sequenzieller und paralleler Ausführung wurden mehrere Experimente ohne Nutzung von Docker und mit je 1, 8, 12 und 15 parallelen Simulationsläufen unter Nutzung von Docker durchgeführt. Hierbei zeigt sich, dass die exemplarischen 50 Simulationsläufe aufgrund der zusätzlichen Aufbauzeit der Docker-Container unter Verwendung eines einzelnen Containers zunächst ca. 3 Minuten langsamer durchgeführt werden als ohne Docker-Container. Durch die Parallelisierung und die gleichzeitige Ausführung von 12 Docker-Netzwerken, wird allerdings eine Simulationszeit für 50 Läufe von unter 4 Minuten, statt den über 18 Minuten bei der sequenziellen Ausführung, realisiert. Der Flaschenhals ist hierbei die Nutzung von ca. 3 GB Arbeitsspeicher pro Kombination von openPASS und HAD-Funktion bei einem Gesamtarbeitsspeicher von 32 GB des verwendeten Testsystems. Dies erklärt ebenfalls den erneuten Anstieg der Simulationszeit unter Durchführung von mehr als 12 gleichzeitigen Simulationsläufen.

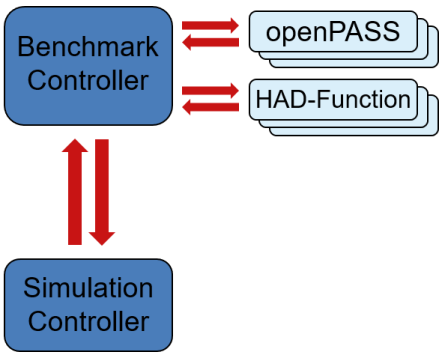


Abbildung 159: Kommunikationsweise der Heuristik

Um eine möglichst schnelle Abarbeitung der durchzuführenden Simulationsaufgabe zu erhalten und gleichzeitig zu vermeiden, dass Docker-Container aufgrund einer zu hohen Auslastung der Systemressourcen gestoppt werden, wurde für die Parallelisierung der Simulationsdurchführung eine Heuristik-Komponente eingeführt (siehe Abbildung 159), die vor den ersten Simulationsläufen zunächst eine möglichst optimale Anzahl an parallelen Simulationsinstanzen schätzt. Hierfür wird vor den ersten effektiv genutzten Simulationsläufen jeder Container des Simulationskerns (in diesem Fall der Container der openPASS enthält und der Container in dem sich die HAD-Funktion befindet) 10 mal gestartet und nach wenigen Sekunden gestoppt um die maximale durch den jeweiligen Container herbeigeführte Systemressourcenbelastung zu messen. Anschließend wird pro Container das Maximum über die 10 Starts gebildet und der erhaltene Wert über die Container summiert. Anschließend werden die verfügbaren Systemressourcen (abzüglich einer Fehlergrenze) durch die berechneten benötigten Systemressourcen geteilt und die daraus erhaltene maximale Anzahl an parallelen Simulationsinstanzen an den Simulation Controller zurückgegeben. Diese Heuristik ist

aufgrund der Systemunabhängigkeit eher konservativ gehalten und könnte für ein festes bekanntes System weiter optimiert werden. Da die Simulationsplattform jedoch so systemunabhängig wie möglich gehalten werden sollte, wurden diese Schritte an dieser Stelle nicht durchgeführt.

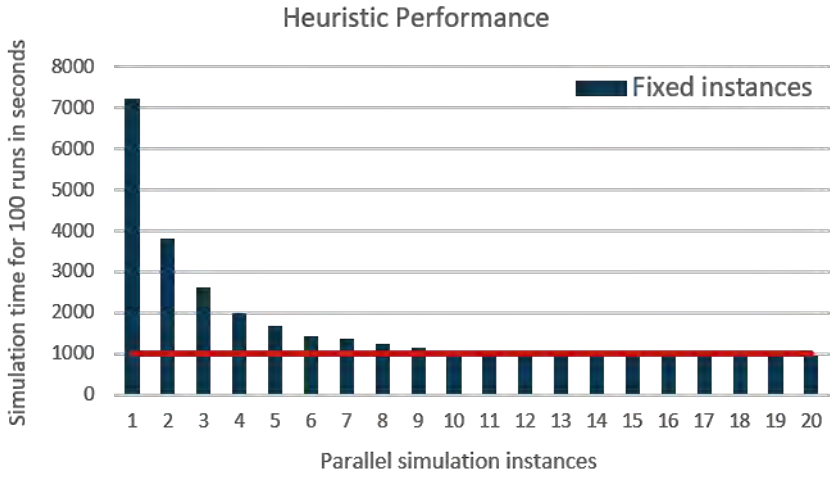


Abbildung 160: Untersuchung der Performanz der Heuristik

Die Abbildung 160 zeigt dabei zum einen die Laufzeit der Gesamtsimulation bei festgesetzter Anzahl von parallelen Simulationsinstanzen (1-20, blau) für 100 Simulationsläufe, sowie die Gesamtsimulationszeit mit der Anzahl von parallelen Simulationsinstanzen, die die vorgeschlagene Heuristik für diesen Fall errechnet hat (13 Simulationsinstanzen, rot).

2.3.3.2.4 Durchführung und Auswertung Kritikalitätsanalyse

Im Laufe der Entwicklung wurde zunächst eine einfache Kreuzung mit einem rechtsabbiegenden Ego-Fahrzeug und anschließend eine komplexere Kreuzung mit einem linksabbiegenden Ego-Fahrzeug in der Simulation realisiert und anschließend analysiert. Um die inhaltlichen Entwicklungen aus dem assoziierten Projekt VVMethoden und der dort entwickelten Kritikalitätsanalysemethode angemessen zu repräsentieren, wurde die durchgeführte Analyse dahingehend abgestimmt, sodass insbesondere die folgenden zwei Analysearten näher beleuchtet wurden:

- Explorative Analyse: Das Ziel der explorativen Analyse ist die möglichst vollständige Erfassung aller kritischen Situationen in einem gegebenen Szenarioraum (der Menge der unter einer Parametervariation möglichen Szenarien).
- Phänomenologische Analyse: Ziel der phänomenologischen Analyse ist hingegen die assoziative Analyse der Relevanz eines gegebenen Faktors innerhalb des Szenarios für eine erhöhte gemessene Kritikalität.

Simulation zur Analyse des Kritikalitätsphänomens *statischen Verdeckung*

Für die Evaluation der Architektur, der Schnittstellen und der aufgebauten Werkzeugkette sowie zur Durchführung einer exemplarischen Kritikalitätsanalyse wurden insgesamt 10.000 Simulationsläufe mit den o.g. Agenten im erläuterten Szenario durchgeführt. Von den Simulationsläufen enthielten 5000 (50%) ein parkendes Fahrzeug am Straßenrand und somit mögliche Effekte durch Verdeckung. In den Szenarien wurden des Weiteren die Startposition und die Zielgeschwindigkeit der zwei Fußgänger und des Ego-Fahrzeugs variiert. Die Konfiguration der Simulationsplattform und die Ausführung der Simulationsaufgabe wurden durch den Analysetask und die Plattformkonfiguration realisiert. Nach dem Einlesen der zwei Dateien

wurden aus einem logischen Szenario jeweils 5000 konkrete Szenarien mit bzw. ohne geparktes Fahrzeug im Kreuzungsbereich erzeugt. Diese wurden anschließend nacheinander simuliert und die Simulationslogs sowie die Observer-Outputs im spezifizierten External Data Storage abgelegt. Nach erfolgreicher Simulationsausführung erfolgte eine Evaluation basierend auf den im Analyse Task spezifizierten Evaluationskriterien. Der Evaluationsreport wurde zum Abschluss erzeugt und an der spezifizierten Stelle im Dateisystem abgelegt.
Auf Grund einer notwendigen Interaktion, musste für die Durchführung initial der Docker Container mit der hochautomatisierten Fahrfunktion einmal manuell gestartet werden. Alle anderen beschriebenen Arbeitsschritte erfolgten automatisiert.

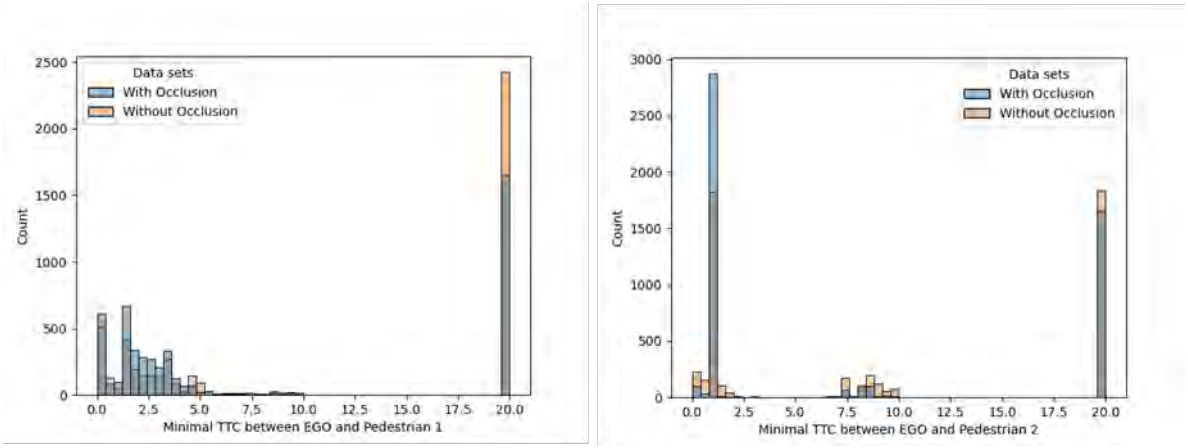


Abbildung 161: Histogramme über den Wert der minimalen TTC in jeweils 5000 Simulationsläufen

Die Werkzeugkette lief stabil und die Simulationsläufe konnten abgearbeitet werden. Lediglich im Fehlerfall (z. B. Spawnposition eines Agenten befindet sich abseits der Straße) kam es anfangs zu Problemen beim Ablauf der Werkzeugkette. Diese konnten jedoch durch Anpassungen bei der Auswertung der Logfiles behoben werden. Zum Zeitpunkt dieser Analyse fehlte hier noch eine entsprechende API des Simulationskerns. Des Weiteren wäre es wünschenswert gewesen den Docker Container mit der Fahrfunktion ebenfalls automatisiert zu starten.
In Abbildung 161 ist die minimale TTC zwischen Ego-Fahrzeug und den querenden Fußgängern (Fußgänger 1 links, Fußgänger 2 rechts) für alle Simulationsläufe dargestellt. Fußgänger 1 überquert die Straße im Kreuzungsbereich, Fußgänger 2 den nördlichen Kreuzungsarm erst etwas entfernt von der Kreuzung.
In der Mehrzahl der Simulationsläufe mit Fußgänger 1 (siehe Abbildung 161, links) lag die TTC über 20 Sekunden. Es trat somit keine kritische Verkehrssituation auf. Erkennbar ist weiterhin eine Häufung von TTCs zwischen Null und fünf Sekunden mit zwei Peaks bei 0 und 1,5 Sekunden. Weiterhin ist zu erkennen, dass es mehr Fahrten mit einer niedrigeren TTC gab, wenn die Fußgänger möglicherweise durch das parkende Fahrzeug verdeckt wurden.
In den Simulationsläufen mit Fußgänger 2 (siehe Abbildung 161, rechts) gibt es auch recht viele Simulationsläufe mit einer TTC über 20 Sekunden. Weitaus mehr Simulationsläufe haben jedoch eine TTC von genau einer Sekunde. Eine Analyse dieser Simulationsläufe hat ergeben, dass es sich dabei allerdings nicht um kritische Verkehrssituationen handelt, sondern der Peak in der TTC aus dem Verhalten des Fußgängeragenten nach Erreichen der Zielposition resultiert. Der Fußgänger dreht sich an der Zielposition auf der Stelle im Kreis, wodurch vom TTC Observer zeitweise eine Kollision vorhergesagt wird. Ohne Berücksichtigung dieser Spitze sind zwei Cluster erkennbar. Das eine Cluster spiegelt die Verkehrssituation wider, in der sich das Ego-Fahrzeug im Kreuzungsbereich befindet und der zweite Fußgänger bereits die Straße überquert. Das zweite Cluster spiegelt eher die Verkehrssituationen wider, in der Fußgänger 2 versucht die Straße zu überqueren, während das Ego-Fahrzeug den

Kreuzungsbereich verlassen hat und bereits einen Teil des Weges zum Fußgänger 2 zurückgelegt hat.

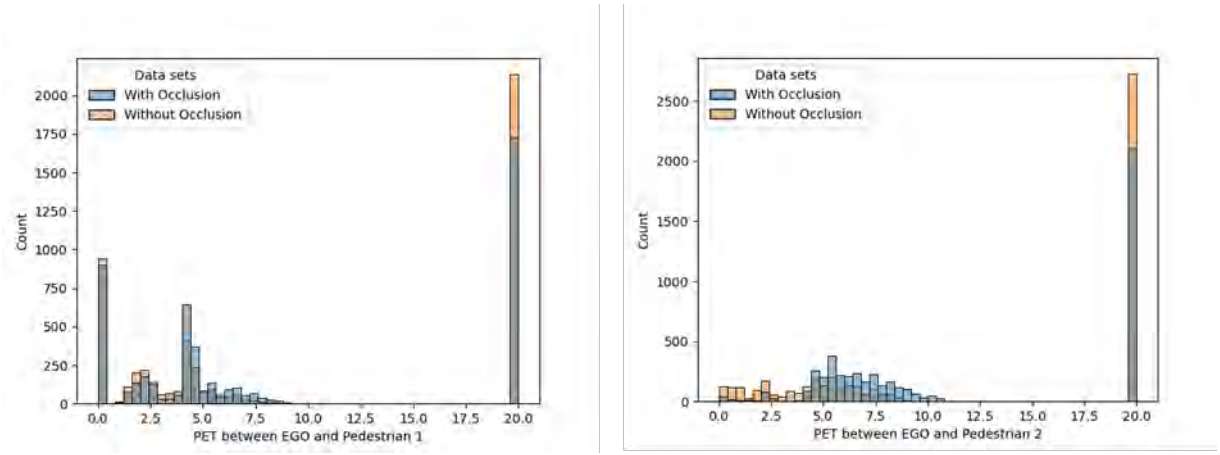


Abbildung 162: Histogramme über den Wert der berechneten PET in jeweils 5000 Simulationsläufen

In Abbildung 162 ist ein Histogramm der PET zwischen dem Ego-Fahrzeug und den beiden Fußgängern dargestellt. Die beiden Spitzen bei jeweils 20 Sekunden (dies entspricht dem Fall, dass keine PET berechnet werden konnte) sind das Ergebnis einer sehr defensiven Fahrweise der Fahrfunktion, die in verschiedenen Verkehrssituationen zu Deadlocks führen, bei denen das Ego-Fahrzeug aufgrund eines nahen Fußgängers (oder eines anderen Fahrzeugs) anhält und nicht wieder losfährt. Eine PET von 0 Sekunden ist oftmals auch mit Deadlocks verbunden. In diesem Fall haben beide Agenten den Konfliktbereich betreten und nie verlassen. Neben diesen Deadlock-Situationen ist auch wieder der Effekt der Verdeckung durch das parkende Fahrzeug in den Ergebnissen sichtbar.
Im Verhältnis ist die PET zwischen dem Ego-Fahrzeug und dem Fußgänger 2 in den Simulationsläufen mit dem parkenden Fahrzeug größer als in den Simulationsläufen ohne. Dies ist darauf zurückzuführen, dass in den Simulationsläufen mit Verdeckung das Ego-Fahrzeug oftmals länger im Kreuzungsbereich verweilt und der Fußgänger somit bereits frühzeitig die Straße überquert hat.

Simulation zur Analyse des Kritikalitätsphänomens *Linksabbiegen bei Gegenverkehr*
Für die simulative Analyse des Kritikalitätsphänomens *Linksabbiegen bei Gegenverkehr* wurden ebenfalls eine explorative und eine phänomenologische Analyseaufgabe definiert, die die exemplarische Anwendbarkeit zeigen. Folgende Parameter wurden für diese Analysen basierend auf dem in Abbildung 163 dargestellten Szenario variiert:

- Startpositionen der Fahrzeuge des entgegenkommenden Verkehrs *„Oncoming Traffic“*
- Startposition des Fahrzeugs des einscherenden Verkehrs *„Merging Traffic“*
- Startposition des Ego-Fahrzeugs *„EGO Vehicle“*
- Startpositionen und Startgeschwindigkeiten der kreuzenden Fußgänger *„Crossing Pedestrians“*

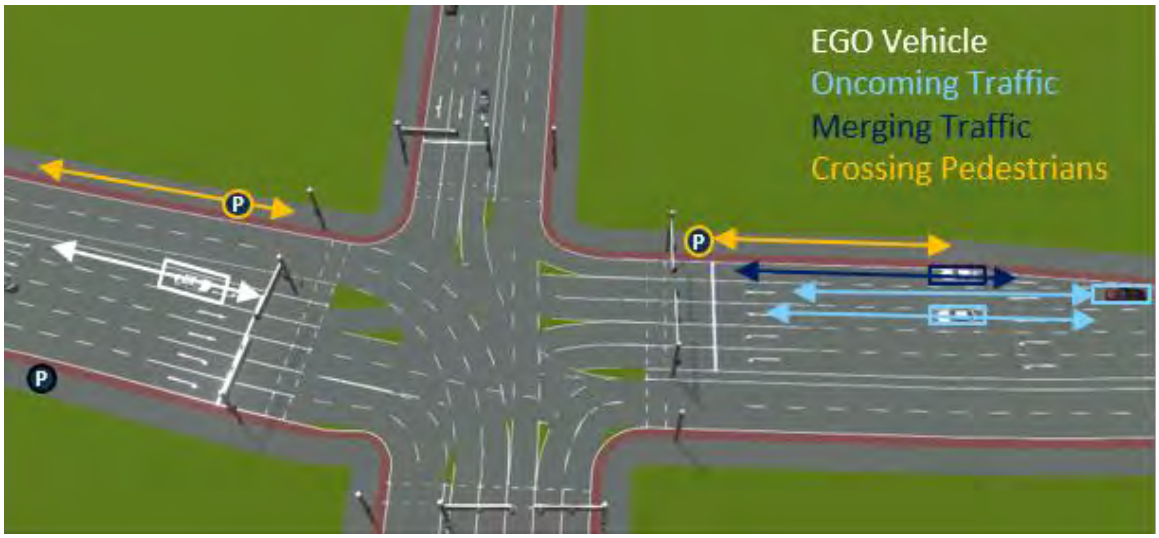


Abbildung 163: Darstellung der Parametervariation für die Untersuchung des Kritikalitätsphänomens Linksabbiegen bei Gegenverkehr

Für die explorative Analyse, also die Erkennung möglichst aller kritischen Szenarien in einem hochdimensionalen Parameterraum, wurden zunächst ca. 40.000 Simulationsläufe durchgeführt. Dies wurde insbesondere durch die Parallelisierung der Simulationsdurchführungen und die dadurch verbesserte Auslastung der verfügbaren Systemressourcen erreicht. Bei der explorativen Simulation fanden sich mehrere Anhäufungen von kritischen Szenarien, die sich jeweils innerhalb der Anhäufungen in der Charakteristik des Auftreffens der Verkehrsteilnehmer an der Ampel am nördlichen Kreuzungsarm ähnelten. Dabei war insbesondere eine zeitliche Nähe des Erreichens der genannten Ampel zwischen dem Ego-Fahrzeug und dem Fahrzeug des ebenfalls einscherenden Gegenverkehrs für eine resultierende erhöhte Kritikalität erkennbar. Aus diesem Grund wurde eine weitere Analyse durchgeführt, um zu untersuchen, ob der einscherende Verkehr die Kritikalität der Gesamtsituation erhöht oder durch vorsichtigeres Anfahren des Ego-Fahrzeugs an den Ampelbereich aufgrund des einscherenden Verkehrs die über die Verkehrsteilnehmer aggregierte Kritikalität verringert ist.
Für die Durchführung der phänomenologischen Analyse, also der Untersuchung, ob das Kritikalitätsphänomen *Linksabbiegen mit einscherendem Gegenverkehr* eine valide Konkretisierung des Kritikalitätsphänomens *Linksabbiegen* darstellt, wurde zu den zuvor variierten Parametern noch die Existenz des einscherenden Verkehrs hinzugenommen. Die Analyse fand hierbei wiederum unter Beachtung der beiden Kritikalitätsmetriken TTC und PET statt.

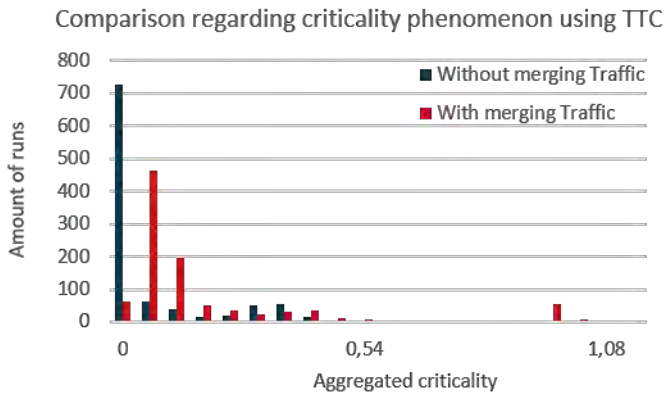


Abbildung 164: Vergleich der Kritikalität mit und ohne Präsenz des konkretisierten Kritikalitätsphänomens gemessen mit TTC

Abbildung 164 stellt hierbei die über die Läufe aggregierte Kritikalität basierend auf der Kritikalitätsmetrik TTC dar. Weiterhin wurde die Kritikalität – die als Gesamtrisiko der involvierten Akteure der Situation definiert ist – als Aggregation der TTC des Ego-Fahrzeugs mit den relevanten Akteuren der Situationen aufgefasst, wobei die Formel

$$C_{TTC} = \sum_{A,B \in Actors} e^{-TTC(A,B)}$$

angewendet wurde um eine Aggregation zu ermöglichen.
Besonders interessant ist hierbei der deutliche Unterschied beim Ausschlag bei Werten in der Nähe von 1, da aufgrund der Reskalierung der Metrik hier vornehmlich die Unfälle wiederzufinden sind. Es zeigt sich daher schon an dieser Grafik unter Benutzung der TTC, dass sich die Präsenz des einscherenden Gegenverkehrs in einer erhöhten Kritikalität der Situation zeigt. Nach der Abbildung war die Kritikalität ohne einscherenden Verkehr größtenteils sehr gering und eine deutliche Trennung der Situationen nach vorhanden sein des einscherenden Verkehrs ist erkennbar.

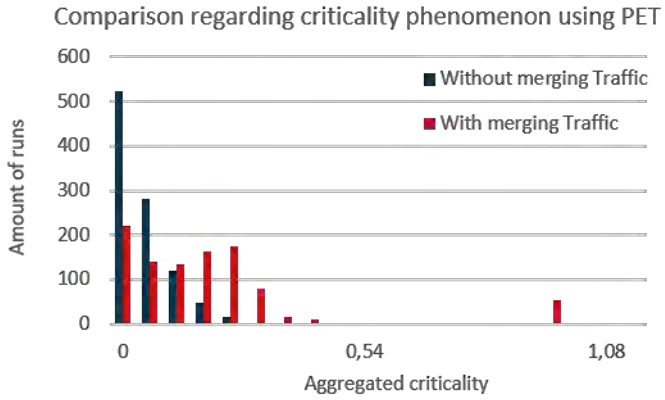


Abbildung 165: Vergleich der Kritikalität mit und ohne Präsenz des konkretisierten Kritikalitätsphänomens gemessen mit PET

Die Abbildung 165 wurde ebenfalls exemplarisch unter Benutzung der für die PET angepassten Aggregationsformel erstellt:

$$C_{PET} = \sum_{A,B \in Actors} e^{-PET(A,B)}$$

Auch in dieser erkennt man eine deutliche Erhöhung der Anzahl der Szenarien in denen eine Kritikalität nahe 1 gemessen wurde. Weiterhin lässt sich hier jedoch ebenfalls erkennen, dass die Kritikalitätseinschätzung durch die aggregierte Kritikalitätsmetrik sowohl ohne als auch mit einscherendem Verkehr deutlich breiter gefächert ist. Daraus folgt, dass auch bereits knappere Szenarien ohne Unfall kritischer dargestellt werden und damit klarer erkennbar sind. Es sollte an dieser Stelle allerdings erwähnt werden, dass die Untersuchung unter Nutzung dieser Aggregation exemplarisch ist und nähere Experimente bezüglich der optimalen Wahl einer Aggregationsmethode und Kritikalitätsmetrik notwendig sind.
Insgesamt lässt sich Abbildung 164 und Abbildung 165 erkennen, dass die Existenz von einscherendem Gegenverkehr beim Linksabbiegen eine deutliche Kritikalitätserhöhung mit sich bringt und damit das Kritikalitätsphänomen *Linksabbiegen mit einscherendem Gegenverkehr* eine valide Konkretisierung des Kritikalitätsphänomens *Linksabbiegen* darstellt.

2.3.3.2.5 Begleitende Aktivitäten

Neben dem Aufbau und der Durchführung der Simulationen in TP 4 wurden im Rahmen des SUC 1 auch weitere Aktivitäten im Projekt unterstützt. Diese sind im Folgenden zusammengefasst.
Alle Aktivitäten im Hinblick auf den SUC 1 wurden unter dem Gesichtspunkt der Traceability vollständig dokumentiert. Dazu wurde das CSP-Framework aus TP 3 herangezogen und durch den SUC 1 direkt in der Anwendung erprobt. Neben der entstandenen Dokumentation konnte durch einige Feedbackschleifen auch der Prozess verbessert werden.
Die Simulationen im SUC 1 wurden nicht nur im TP 4 selbst durchgeführt, sondern auch bei weiteren Partnern aufgebaut und erprobt. Diese Partner wurden kontinuierlich in ihren Tests unterstützt und Feedback eingeholt. Das Feedback wurde in weiteren Ausbauten des SUC 1 eingearbeitet. Analog wurden Teile des SUC 1 oder modifizierte Aufbauten im Projekt verwendet, welche analog aus TP 4 heraus unterstützt wurden. Beispielhaft ist die Performanceevaluierung von Modellen in der Anbindung per OSMP, als auch die angestrebte Kopplung von openPASS mit SUMO zu nennen.
Außerhalb des SET Level Projekts hat eine enge Kooperation mit VVMethoden stattgefunden. Dabei wurden sowohl methodische Ansätze ausgetauscht und angewendet als auch die entwickelten Umfänge und Integrationsergebnisse zwischen den Projekten ausgetauscht.
Die verwendeten ASAM-Standards, in die zwischenzeitlich auch Weiterentwicklungen aus SET Level eingeflossen sind, finden sich auf der offiziellen ASAM Homepage:

- OpenDRIVE: https://www.asam.net/standards/detail/opendrive/
- OpenSCENARIO: https://www.asam.net/standards/detail/openscenario/
- OSI: https://www.asam.net/standards/detail/osi

Die verwendeten MODELICA-Standards finden sich hier:

- FMI: https://fmi-standard.org/
- SSP: https://ssp-standard.org/

Die Erweiterungen an openPASS sind im openPASS open source Projekt veröffentlicht und dort in das Release v0.7 bis v0.10 eingeflossen:

- Homepage: https://openpass.eclipse.org/
- Repository: https://gitlab.eclipse.org/eclipse/openpass

Die finale Version der Simulation wurde u. a. auf der EclipseCon 2022 beim Community Day am 24.10.2022 (https://wiki.eclipse.org/EclipseCon_2022_Automotive_Community_Day) und in der openPASS Working Group präsentiert. Viele der beschriebenen Aktivitäten werden zukünftig im open source Eclipse openPASS Projekt nahtlos fortgeführt.

2.3.3.3 Closed-loop Integrationstests

Closed-loop Simulationen werden vor allem im rechten Ast des V-Prozesses verwendet, also im Bereich der Integration und des Tests von Subsystemen oder des Fahrzeuggesamtsystems, der Validierung des Gesamtsystems sowie der abschließenden Abnahme- oder Akzeptanztests (siehe Abbildung 13). Die damit verbundenen Simulationsaufgaben bringen eine Vielzahl von Herausforderungen mit sich, welche im Rahmen des Projektes SET Level im Rahmen des Simulation Use Case 2 (SUC 2) eingehend untersucht wurden. Das Vorgehen und die wesentlichen Erkenntnisse sowie die angewendeten und weiterentwickelten Werkzeugketten werden in diesem Abschnitt näher erläutert.

2.3.3.3.1 Closed-loop Simulation für Validierung und Integrationstests

Der im Projekt als SUC 2 bezeichnete Anwendungsfall umfasst den Einsatz einer Closed-loop Simulation zur Verifikation oder Validierung einer Fahrzeugautomation bzw. eines wesentlichen Subsystems, das die Steuerungsfunktion einschließt.
Dieser Use Case kann in verschiedenen Phasen der Entwicklung einer Automation angesiedelt sein. Entsprechend den V&V-Aufgaben der Phase werden sich diese Ausprägungen des Use-Cases unterscheiden. Mögliche Ausprägungen sind zum Beispiel:

a. Test einer Entwicklungsversion einer HAD-Funktion (Testobjekt: HAD-Funktion)
b. (Beitrag zu einer) Validierung einer HAD-Funktion (Testobjekt: HAD-Funktion)
c. (Beitrag zu einer) Validierung eines Automationssystems (Testobjekt: Modell des Automationssystems mit allen Komponenten)

Während der Projektlaufzeit wurden dabei unterschiedliche Szenarien betrachtet. anhand derer Werkzeugketten für die Durchführung einer Simulation im Sinne von SUC 2 weiterentwikkelt wurden. Die gewählten Testfälle gingen allesamt von einer zu testenden Entwicklungsversion einer HAD-Funktion als zentralem Testobjekt aus.
Dabei galt es, unterschiedliche Aspekte zu analysieren, die im Projekt betrachtet und erarbeitet wurden. Dies umfasste u. a.

- Anwendbarkeit von aktuellen Standards
- Nutzung von geeigneten Architekturen und Schnittstellen
- Ausarbeitung beispielhafter Metriken
- Anwendung des Credible Simulation Prozesses

Eine sehr zentrale Rolle spielten dabei im gesamten Projektverlauf die Verwendung von Simulationsstandards, wobei vor allem fünf zentrale Standards im Fokus standen: OpenSCENARIO, OpenDRIVE, SSP, OSI und FMI (siehe Abbildung 166).

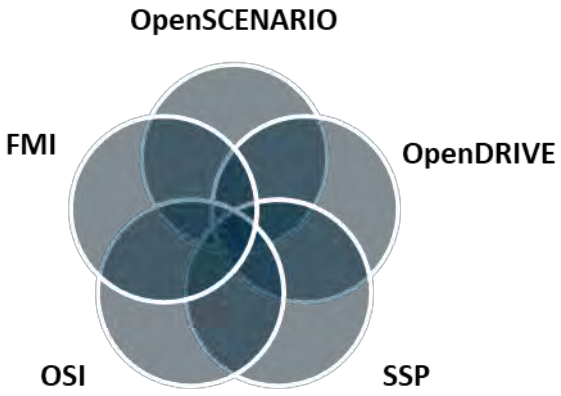


Abbildung 166: Für SUC 2 zentral relevante Standards

In nahezu allen Standards wurden dabei die aktuellen Entwicklungen begleitet, so dass es im laufenden Projekt zu Wechseln auf neuere Standardversion kam, die auch z.T. (vor allem beim OSI Standard) schon auf zurückgespiegelten Ergebnissen aus der Anwendung der Standards auch im SUC 2 kam. Im Einzelnen wurden dabei folgende Versionen der Standards verwendet:

- OpenSCENARIO 0.9.1, 1.0 & 1.1
- OpenDRIVE 1.4 & 1.6
- SSP 1.0
- OSI 3.2.0 & 3.3.1
- FMI 2.0

Insbesondere von Interesse waren dabei die Weiterentwicklung von OpenSCENARIO und OSI.
In Zusammenhang mit OpenSCENARIO wurde ausgehend von einzelnen konkreten Szenarios (0.9.1) über die Verwendung von proprietären Skripten zur Umwandlung von logischen Szenarien in konkrete Szenarien (1.0) hin zu einer integrierten Beschreibung von logischen Szenarien inklusive der relevanten Parameterräume (1.1) eine intensive Weiterentwicklung des Standards mitvollzogen. Auch die praktische Anwendung des OSI Standards resultierte in regen Diskussionen und Weiterentwicklungen, die sich schließlich in der zuletzt im SUC 2 verwendeten Version (3.3.1) wiederfanden
Im Mittelpunkt der Simulationen im SUC 2 stand immer die HAD-Funktion als System under Test. Dabei wurden insgesamt drei unterschiedliche Szenarien betrachtet, bei denen es jeweils um Abbiegemanöver an Kreuzungen ging, mit dem Ziel, keine anderen Verkehrsteilnehmer zu gefährden. Während zunächst ein einfaches Abbiegemanöver mit einem kreuzenden Fußgänger auf der zu überquerenden Fußgängerfurt betrachtet wurde, kamen im nächsten Schritt weitere Verkehrsteilnehmer hinzu, die zum einen die Interaktion erhöhten, und zum anderen für eine abschnittsweise Verdeckung des Fußgängers sorgten (LS 2-5 in Abbildung 167). Im letzten Abschnitt des Projekts wurde dann ein Szenario betrachtet, in dem ein Fehler im Lenk-Aktuator zu einer reduzierten maximalen Verstellgeschwindigkeit des Radeinschlagwinkels führte, worauf die HAD-Funktion zu reagieren hatte (LS 2-6 in Abbildung 167).

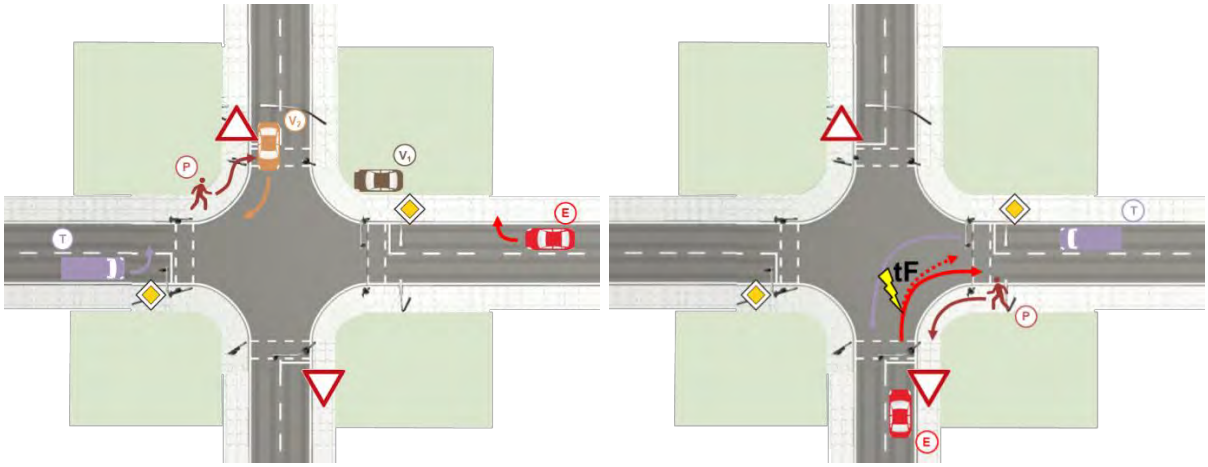


Abbildung 167: Kreuzungsszenario LS 2-5 (links) und LS 2-6 (rechts) auf dem Verkehrsraum TS 2-4

Auch hinsichtlich der benötigten Simulationsmodelle im SUC 2 zeigte sich eine fortlaufende Erweiterung der relevanten und zu integrierenden Modelle. Bei der finalen Umsetzung des Szenarios LS 2-6 wurde schließlich folgende Modell-Konstellation realisiert:

- Fahrzeugsystem
    - HAD Funktion
        - Fahrzeugautomationslogik + Sensorfusion (FZI)
        - Motion Control (LBF)
    - Fahrdynamikmodell LoD 2 (LBF, ZF)
    - 1x Kamera (Bosch)
    - 1x Lidar, objektbasiert (FZD)
    - 1x Radar, objektbasiert (FZD)
- Agentenmodelle

- Fahrzeuge
  - LKW Modell LoD 1 (MAN)
  - Driver Modell (IKA)
- Fußgänger (BMW)

Lediglich die HAD-Funktion wurde dabei fortlaufend weiterentwickelt und in unterschiedlichen Versionen eingesetzt, welche auf die jeweiligen Szenarien angepasst und im Funktionsumfang bzw. der Parametrierung geändert wurden. Das Fahrdynamikmodell kam zunächst im Detaillierungsgrad LoD 1 zum Einsatz (LS 2-5) und wurde dann aufgrund der für den Test der HADfunktion bei degradierter Performanz des Lenkaktuators benötigten, erweiterten und detaillierter modellierten Eigenschaften des LoD 2 Modells durch dieses ersetzt (LS 2-6).
Eine besondere Herausforderung, die sich speziell anhand des SUC 2 zeigte, war die geschickte Umwandlung von Simulink-Modellen in FMUs und deren Integration in die Gesamtsimulation.
Mit der beständigen Erweiterung durch mehr und mehr an der Simulation beteiligte Modelle, musste auch die Simulationsarchitektur nach und nach erweitert werden. Mit der oben aufgelisteten Modellauswahl ergab sich dabei folgendes Architekturbild für LS 2-6:

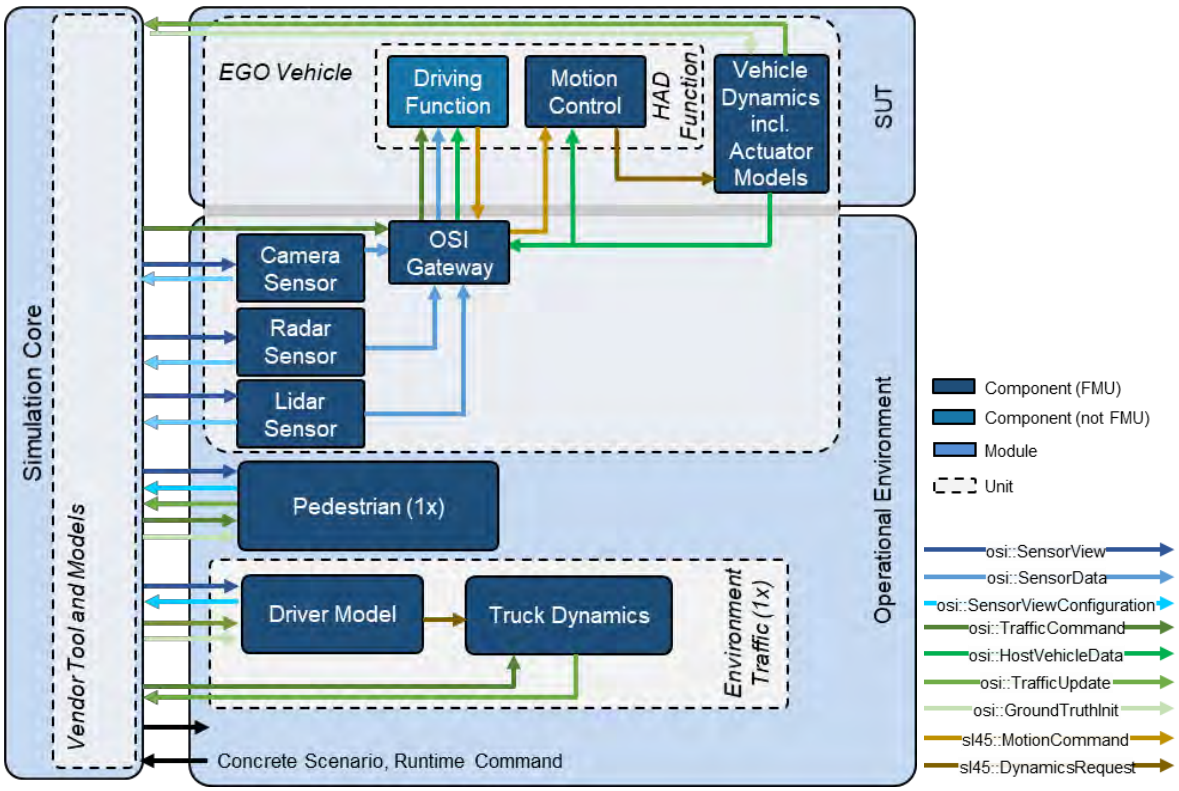


Abbildung 168: Simulationsarchitektur für den SUC 2 zur Umsetzung des Szenarios LS 2-6

Die Architektur richtet sich hierbei sowohl an den erarbeiteten Grundlagen für die Definition einer flexiblen und modularen Simulationsarchitektur aus, als auch an dem Ziel, Werkzeugketten unabhängige Schnittstellen zu verwenden. Dies gelang vor allem durch die konsequente Verwendung des OSI- und des FMI-Standards (vergleiche auch SUC 1 im Abschnitt 2.3.3.2 Closed-loop Verkehrssimulation).

Als Metriken zur Bewertung der Ergebnisse wurden im Laufe des Projektes unterschiedliche Messgrößen verwendet. Dazu gehören:

- Die Time-to-Brake (TTB)
- Die Post Encroachment Time (PET)
- Der minimale laterale Abstand zu den anderen Verkehrsteilnehmern
- Der longitudinale Abstand zu den anderen Verkehrsteilnehmern
- Die Geschwindigkeit des Ego-Fahrzeugs

Die PET beschreibt die Zeitlücke zwischen dem Zeitpunkt, bei dem ein Verkehrsteilnehmer (hier der Fußgänger) die Interferenzzone mit einem anderen Verkehrsteilnehmer (hier das Ego-Fahrzeug) verlässt, und dem Zeitpunkt, bei dem der zweite Verkehrsteilnehmer diese erreicht. Eine PET von 0 bedeutet, dass es einen Zusammenstoß gab, in diesem Fall also der Test der HAD Funktion nicht erfolgreich bestanden wurde.
Die TTB ist dagegen eine Metrik, welche auch zur Bewertung der Kritikalität einer Situation herangezogen werden kann. Sie beschreibt die verblebende Zeit, um mit einer bestimmten Verzögerung noch vor einem Objekt oder einer Interferenzzone bremsen zu können.
Die jeweils relevanten Metriken wurden beispielhaft ausgewertet. Der Fokus beim SUC 2 lag jedoch auf dem Prozess der Modell-Integration bis hin zur Simulationsdurchführung mit unterschiedlichen Werkzeugketten und nicht auf detaillierten Testergebnissen. Daher wird an dieser Stelle nicht näher darauf eingegangen.
Wie geschildert, war u. a. der Nachweis der flexiblen Realisierung der SUCs in Werkzeugketten ein Kernpunkt der Arbeiten. Die im Projekt erzielten Ergebnisse und Entwicklungen der Werkzeugketten des DLR, von dSPACE und IPG Automotive werden daher in den folgenden Abschnitten ausführlich dargestellt.

2.3.3.3.2 Werkzeug Forschungsimplementierung

Motivation und Ziele

Das Projekt SET Level adressiert das simulationsbasierte Entwickeln und Testen von automatisierten Fahrzeugen für urbane Räume. Mit der Weiterentwicklung numerischer Simulationsmethoden, der Konzeptionierung und Umsetzung neuer Mechanismen zur Modellkopplung sowie der punktuellen Neuentwicklung von Modellen, die neben dem jeweils korrekten Verhalten auch ein zu erwartendes nicht-normatives Verhalten im Systemumfeld hinreichend abbilden, werden wichtige methoden- und werkzeugorientierte Bausteine zu effizient gestalteten simulationsbasierten Test- und Freigabeprozeduren geliefert. Besondere Herausforderungen liegen hierbei darin, Anforderungen an die Genauigkeit von Modellen und deren Integrationsfähigkeit mit anderen Modellen oder realen Systembausteinen (z. B. Prüfständen oder Fahrsimulationen) in Balance zu bringen und diese auch formal fassbar zu machen sowie zu quantifizieren. Hierdurch soll u. a. auch eine barrierefreie und insbesondere werkzeugunterstützte Kopplung von Einzelmodellen ermöglicht werden. Die Validität des entstehenden Gesamtmodells ist eine stets zentrale Anforderung. Ein erster Schritt dazu ist die Überprüfung der Kompatibilität von verwendeten Einzelmodell mit den verwendeten Standards (OSI, FMI, OpenX), sowie die Überprüfung des korrekten Zusammenwirkens von Einzelmodellen in einer Gesamtsimulation.
Ziel bei der Umsetzung der Forschungsimplementierung war es daher, eine generische und frei verfügbare Simulationsplattform zu entwickeln, die zur Handhabung höherer Automationsstufen und bei höherer Komplexität in urbanen Umgebungen im Hinblick auf eine Gesamtfahrzeugbeurteilung verwendbar ist und auch zur Ergebnisdarstellung und -demonstration genutzt werden kann. Die Forschungsimplementierung umfasst im Wesentlichen zwei Hauptfunktionen: Den Integrationstest von Simulationsmodulen sowie die Verwendung zur Closed-Loop Simulation von Gesamtfahrzeugsystemen zur Evaluation von Automationsfunktionen

u. a. auch durch Szenarienexplorationen. Für die Integrationsbewertung werden die verwendeten Simulationsmodule einem Integrationstest unterzogen, der prüft, ob sie kompatibel zu den Vorgaben der Integrationsarchitektur sind und somit Basisfunktionalitäten im Zusammenspiel eines vollständigen Simulationssystems erfüllen können. Die Szenarioexploration ermöglicht es logische Szenarien und deren Parameterräume systematisch auszuführen und die Anzahl der auszuführenden Szenarien durch intelligente Algorithmen, anhand von definierten Kriterien, ergebnisorientiert zu optimieren.
Ein wichtiger Aspekt für die Umsetzung der Forschungsimplementierung ist auch die Berücksichtigung der im Projekt erarbeiteten Strukturierung einer generischen Architektur (siehe Abschnitt 2.1.3.3 „Simulations- und Integrationsarchitektur“), die es ermöglichen soll, szenarienbasiertes Testen automatisiert und skalierbar zu betreiben.
Im Projekt diente die Forschungsimplementierung darüber hinaus zur Demonstration einer Closed-loop Simulation unter Verwendung von Modellen verschiedener Partner.

Umsetzung

Die Forschungsimplementierung wurde über den gesamten Zeitraum des Projektes zunächst konzipiert und dann modulweise nach und nach implementiert und zum Projektende hin anhand des SUC 2 demonstriert.
Die im Projekt erarbeitete Plattformarchitektur der Forschungsimplementierung (siehe Abbildung 169) soll es ermöglichen szenarienbasiertes Testen automatisiert und skalierbar zu betreiben.

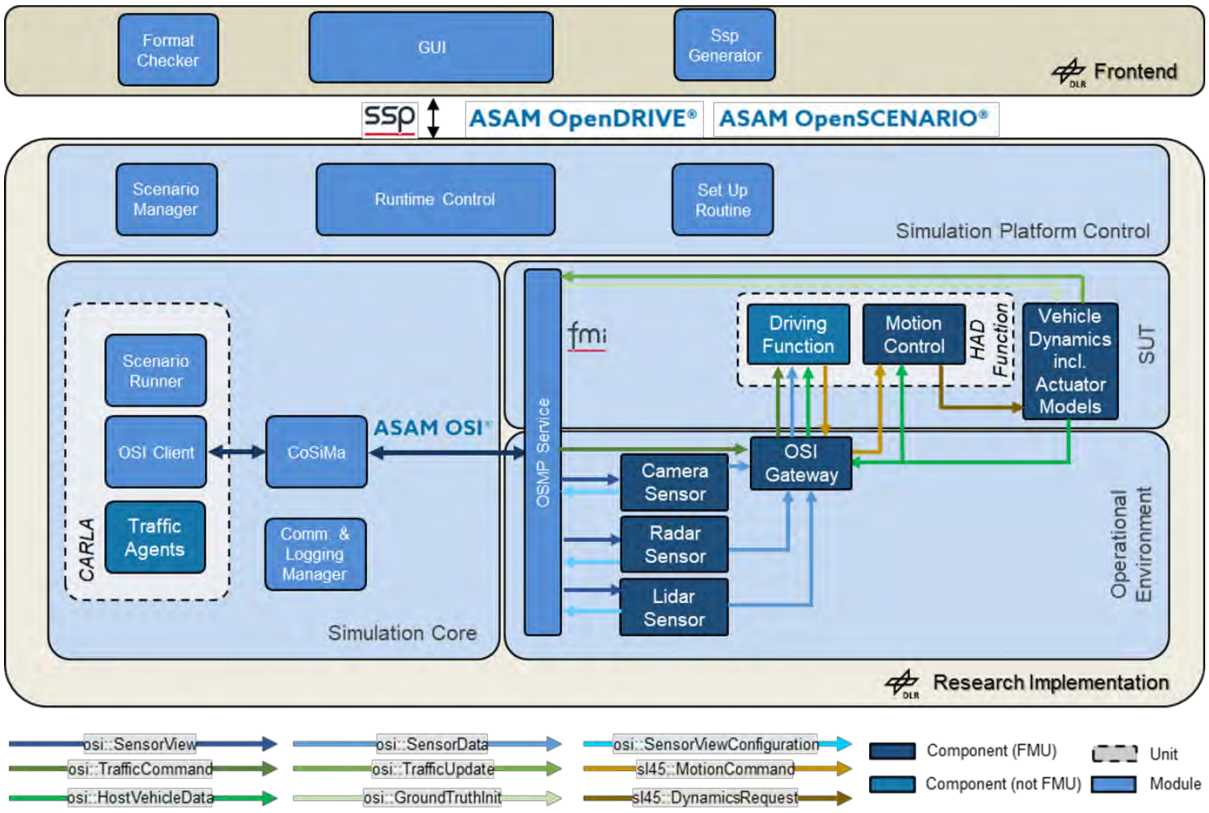


Abbildung 169: Architekturdiagramm der Forschungsimplementierung

Die Grundannahme während des Entwurfs war es die Simulationen auf einem Rechencluster durchzuführen, während die Koordination der zu simulierenden Szenarien von einer eigenen Komponente (Szenario Manager, siehe Abbildung 169) übernommen wird. Des Weiteren wurden die einzelnen Schritte, die zur Durchführung einer (Co-)Simulation notwendig sind,

auch auf eigene Softwarekomponenten aufgeteilt. Eine zentrale Rolle kommt dabei dem Co-Simulationsmanager (CoSiMa, siehe Abbildung 169) zu. Die vorgenommene Trennung zwischen Frontend, Simulation Platform Control und Simulation Core hat zum Vorteil, dass, je nach Ausgestaltung dieser Komponenten, ein Simulation-as-a-Service Betriebsmodell implementiert werden kann. Die Modularität der Plattform hat neben der Skalierbarkeit noch den Vorteil einer offenen Plattform. Es ist möglich einzelne Bestandteile dieser Plattform auszutauschen, um so auch eine langfristige Weiterentwicklung und Nutzung dieser Plattform zu gewährleisten. Damit diese Modularität flexibel ausgenutzt werden kann, müssen diese Module auf standardisierte Schnittstellen zurückgreifen, damit eine Anbindung problemlos möglich ist. Diese Schnittstellen sind innerhalb von SET Level insbesondere FMI, OSI und SSP. Auch die von der Forschungsimplementierung verwendeten Karten und Szenariobeschreibungen verwenden die standardisierten und im Projekt SET Level üblichen Formate OpenDRIVE und OpenSCENARIO. Als zentrale Kernkomponente wird in der derzeitigen Umsetzung die Open Source Simulationssoftware CARLA (http://carla.org/) verwendet.
Die einzelnen Softwaremodule der Forschungsimplementierung sind im Folgenden eingehender beschrieben.
Das *Frontend-GUI* (siehe Abbildung 170) ermöglicht die Konfiguration der Simulation, also die Auswahl der zu verwendenden Karte, der Szenarien sowie der Modelle. Dabei ist es sowohl möglich auf die im Projekt entstandenen Open Source Modelle zurückzugreifen, als auch eigene Modelle, Karten oder Szenarien der Bibliothek hinzuzufügen und zu nutzen. Hier werden abschließend auch die fertig konfigurierten Simulationsjobs gestartet.

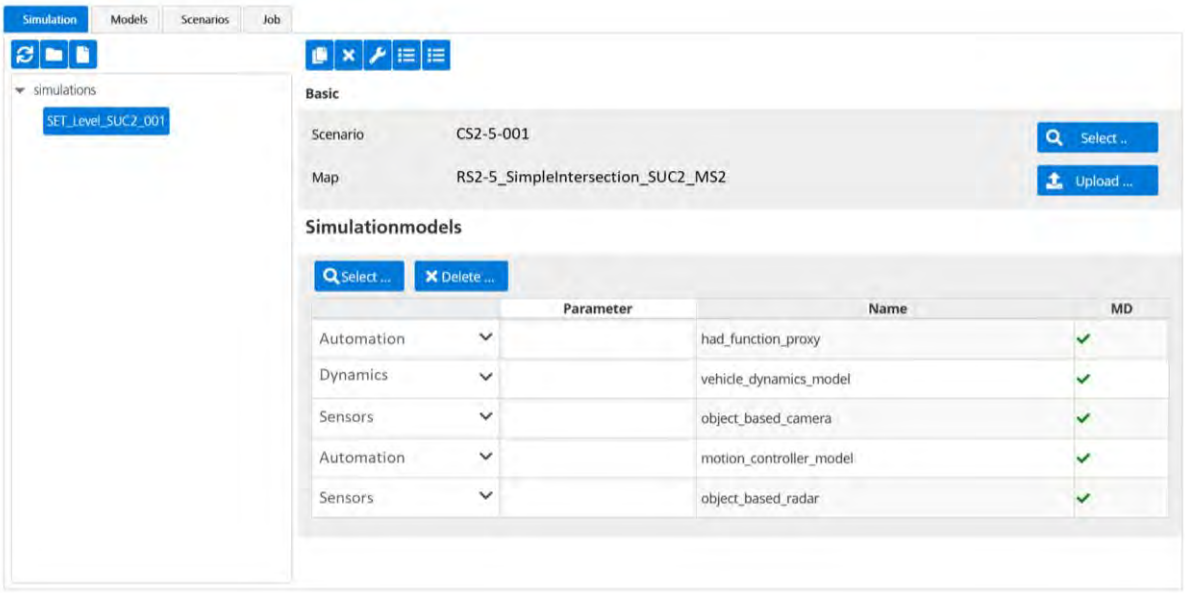


Abbildung 170: Frontend GUI der Forschungsimplementierung

Der *Format Checker* wird aktiv, sobald neue Modelle, Karten oder Szenarien der Bibliothek hinzugefügt werden. Ausgeführt werden hier

- FMI Compliance Checker (siehe https://github.com/modelica-tools/FMUCompliance-Checker)
- Python Script zur XML Schema Validierung relevant für
    - Überprüfung der OpenSCENARIO 2.0 Artefakte
    - Überprüfung der OpenDRIVE 1.4 - 1.7 Artefakte
    - Überprüfung der SSP 1.0 Artefakte
- OSI-Validator (siehe https://github.com/OpenSimulationInterface/osi-validation)

Das Ergebnis der Prüfung wird dem Nutzer in einem Textausgabefenster angezeigt. Der Format Checker kann auch unabhängig vom Frontend ausgeführt werden. Aktuell sind diese Checks als Continuous Integration Pipeline der die Bibliothek verwaltenden GitLab-Instanz eingebunden.
Der *SSP-Generator* befüllt eine Konfigurationsdatei konform zur aktuellen SSP-Definition. Dieser Schritt ist notwendig, da die eigentliche Forschungsimplementierung auch ohne das Frontend voll nutzbar sein soll und daher eine entsprechend standardisierte SSP-Datei als Input erwartet.
Die eigentliche Forschungsimplementierung ist in die Bereiche *Simulation Platform Control*, *Simulation Core*, *SuT* und das *Operational Environment* unterteilt.
Der Bereich *Simulation Platform Control* umfasst die allgemeinen Funktionen, die auf oberster Steuerungsebene innerhalb der Plattform vorhanden sein müssen.
Dazu gehören zunächst die *Set Up Routine*, welche die Konfigurationsdatei auswertet und alle notwendigen vorbereitenden Schritte ausführt. Zunächst wird dabei das SSP Paket entpackt und die darin enthaltenen Informationen an die einzelnen Softwaremodule verteilt. Die verwendeten Szenariodateien werden an den *Scenario Manager* zur Verwaltung gegeben. Die Information welche FMUs an der Simulation beteiligt sind gehen an die *OSMP Service* Instanzen, um das Triggern der Modelle zur Laufzeit zu ermöglichen und den *Communication & Logging Manager*, der diese Information für den Report verwendet. Des Weiteren werde diese Information ausgewertet, um die benötigten Modelle in einer passenden Laufzeitumgebung zu instanziieren. D.h. jede FMU wird automatisch in einem Dockercontainer durch den *OSMP Service* initialisiert und im Rahmen eines Kubernetesclusters weiter verwaltet.
Der *Scenario Manager* ist die Komponente, die die durchzuführenden Szenarien verwaltet. Er hält die Warteschlange an Szenarien und interagiert mit der Scenario Engine des Core Simulators. Sollte es sich bei einer ausgewählten Szenariodatei um ein logisches Szenario handeln, so wird an dieser Stelle die Parameterraumdatei ausgewertet und die korrespondierenden konkreten Szenarien generiert. Im Falle einer Exploration kann die erzeugte Reihenfolge der auszuführenden konkreten Szenarien zwischen den Simulationsdurchläufen entsprechend der gewählten Explorationsstrategie angepasst werden.
Die *Runtime Control* schließlich verwaltet das Zusammenspiel der einzelnen Komponenten. Sie ist verantwortlich dafür, dass die notwendigen übergeordneten Systeme (Simulation Platform Controller, Core Simulator, Operational Environment und SuT) korrekt miteinander interagieren.
Der *Core Simulator* ist die Ausführungsebene, in der die eigentliche Simulationssoftware läuft. Für die Forschungsimplementierung wurde hier die Open Source Simulationssoftware CARLA (siehe http://carla.org/) als Kernkomponente gewählt. Diese wurde im Projekt um einige Features erweitert, welche sich nun als Teil der Software wiederfinden (siehe https://github.com/carla-simulator/carla). Die wichtigste Erweiterung stellt dabei die Integration eines *OSI Clients* dar, welcher mittels OSI Nachrichten die Kommunikation zu als FMUs verfügbaren Modellen ermöglicht.
Der Co-Simulationsmanager *CoSiMa* ist ein vom DLR entwickeltes Softwaremodul, welches die Schnittstelle zwischen CARLA und den Modellen des *Operational Environment* sowie des *SuT* darstellt. *CoSiMa* ermöglicht dabei sowohl die Einbindung der FMUs als auch die Kommunikation zwischen diesen und CARLA zur Laufzeit.
Der *Communication & Logging Manager* verwaltet die Simulationsdaten und loggt die für die Auswertung notwendigen und in der Konfiguration festgelegten Signale.
Die Bereiche des *Operational Environment* sowie des *SuT* umfassen die für die Simulation notwendigen und ausgewählten Modelle. Diese werden über einen *OSMP-Service* über *CoSiMa* in den Simulation Core eingebunden.

Nutzungsmöglichkeit und Ausblick

Das Ergebnis des im Rahmen des Projektes entstandenen Simulationsframeworks ist Open Source. Einzelne Elemente sind wie folgt dokumentiert:

- Beschreibung der Quickstart Demo:
  - https://github.com/DLR-TS/OSTAR-Quickstart
- Tiefergehende Beschreibung vom CoSiMa (Carla/OSMP-Service):
  - https://github.com/DLR-TS/CoSiMa
- Tiefergehende technische Beschreibung der CoSiMa Konfiguration:
  - https://github.com/DLR-TS/CoSiMa/blob/master/Configuration.md

Das Framework wird unter dem Namen OSTAR (Open Simulation Toolchain for Automotive Research) vom DLR auch nach Projektende fortgeführt und findet bereits weitere Anwendung u. a. in den Projekten VVMethoden und GAIA-X-4PLC.
Dabei kommen sowohl die Forschungsimplementierung als Simulationsframework zum Einsatz, als auch die Modellbibliothek aus SET Level (siehe Abbildung 171).

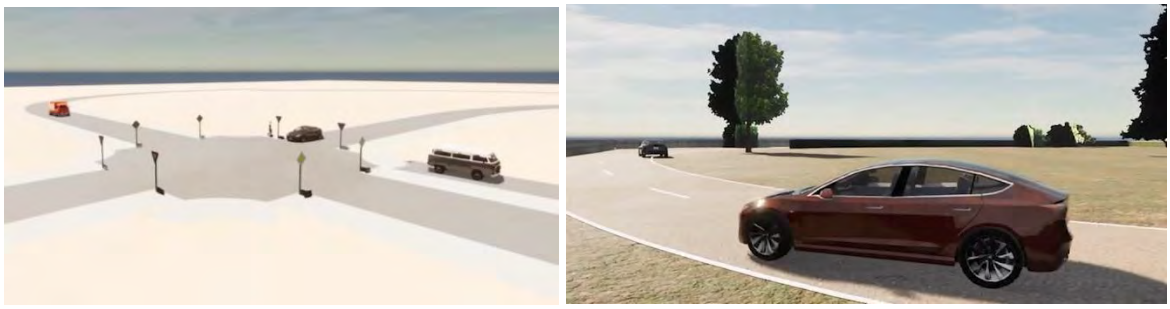


Abbildung 171: Umsetzung des SET Level SUC 2-MS2 Szenarios mit der Forschungsimplementierung (links) und Automationsvalidierung im Rahmen von VVMethoden (rechts)

Neben der Anbindung der Open Source Automationsfunktion ADORe (siehe https://projects.eclipse.org/projects/automotive.adore) und der Simulationsvalidierung im Rahmen einer Studie im Projekt VV-Methoden, sind die Integration eines OpenDRIVE Validators sowie die Erweiterung um ein Modul zur Überprüfung der resultierenden Daten auf Plausibilität (Data consistency check) in Planung.

2.3.3.3.3 Umsetzung in den Werkzeugen von dSPACE

Während der Projektlaufzeit ergab sich eine fortlaufende Modellintegration der von den Projektpartnern bereitgestellten Modelle (FMUs). Diese wurden von dSPACE in mehreren Iterationen in ihrer Umgebungssimulation eingebunden und damit auf ihre Funktionsfähigkeit getestet.
Den Kern der Modellierung bilden die dSPACE Automotive Simulation Models (ASM) als Toolsuite für modellbasierten Test und Entwicklung von Steuergeräten. Die ASM-Modelle basieren auf MATLAB/Simulink. Für dieses Projekt wurde das Simulationsmodell ASM Traffic genutzt, welches neben der Simulation eines Personenwagens auch seine Interaktion mit der Umgebung und umliegendem Verkehr modelliert (Scenario Engine).
Die externen Modelle der Projektpartner sind als FMUs in das Simulationsmodell eingebunden. Dabei wird der für SET Level definierten Modellarchitektur gefolgt. Die vom Simulationsmodell bereitgestellten OSI-Nachrichten werden an die entsprechenden Eingänge der externen Modelle weitergeleitet. Die Kommunikation über OSI-Nachrichten zwischen einzelnen

Komponenten-FMUs wird durch eine direkte Verschaltung der Ausgangsports mit den entsprechenden Eingangsports erreicht.
dSPACE hat zwei verschiedene HAD Funktionen in eine Closed-Loop Simulation überführt. Zum einen die HAD Funktion des FZI und zum anderen einen Motion Planner von MAN:

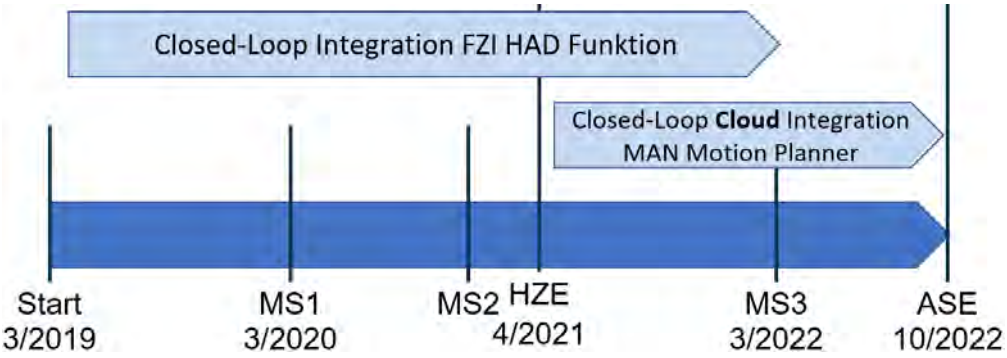


Abbildung 172: Zeitstrahl der dSPACE Closed-Loop HAD Integrationen

Die FZI HAD Funktion als Simulation auf lokaler PC-Hardware und den MAN Motion Planner in der Cloud mit dSPACE SIMPHERA.
Die FZI HAD Integration wurde auf dem Halbzeitevent (HZE) und die MAN Motion Planner Integration wurde auf dem Abschlussevent (ASE) präsentiert. Nachfolgend werden beide Varianten vorgestellt.

dSPACE Integration FZI HAD Funktion
Die HAD-Funktion wird als Docker-Container integriert, der als separates Linux-Ubuntu-System läuft. Die direkte Kommunikation mit der HAD-Funktion (siehe Abbildung 173 „Driving Function“ und „Motion Control“) übernimmt eine in das Simulationsmodell integrierte Komponente, die als TCP-Client agiert. In Abbildung 173 wird dies „OSI GW“ (OSI Gateway) genannt. Durch diese Konfiguration ist es möglich verschiedene Betriebssysteme für die Simulation von Operational Environment und SuT (System under Test) verwenden zu können.
Der Rückgabewert der HAD-Funktion wird über dieselbe Komponente in das Simulationsmodell zurückgeleitet. Da die Komponente blockierend auf den Rückgabewert der HAD-Funktion wartet, ist eine Synchronisation des entkoppelten Docker-Containers mit der Gesamtsimulation gewährleistet.
Zur Datenübertragung wird die C++-Implementierung des Open Simulation Interfaces verwendet. Die Übertragung von OSI-Nachrichten zu FMI-2.x-Modellen folgt der Spezifikation von OSI Sensor Model Packaging (OSMP). Für die Übertragung von OSI-Nachrichten mittels eines TCP-Gateways wird eine Beispiel OSMP-FMU aus dem OSMP-GitHub-Repository als Basis verwendet.

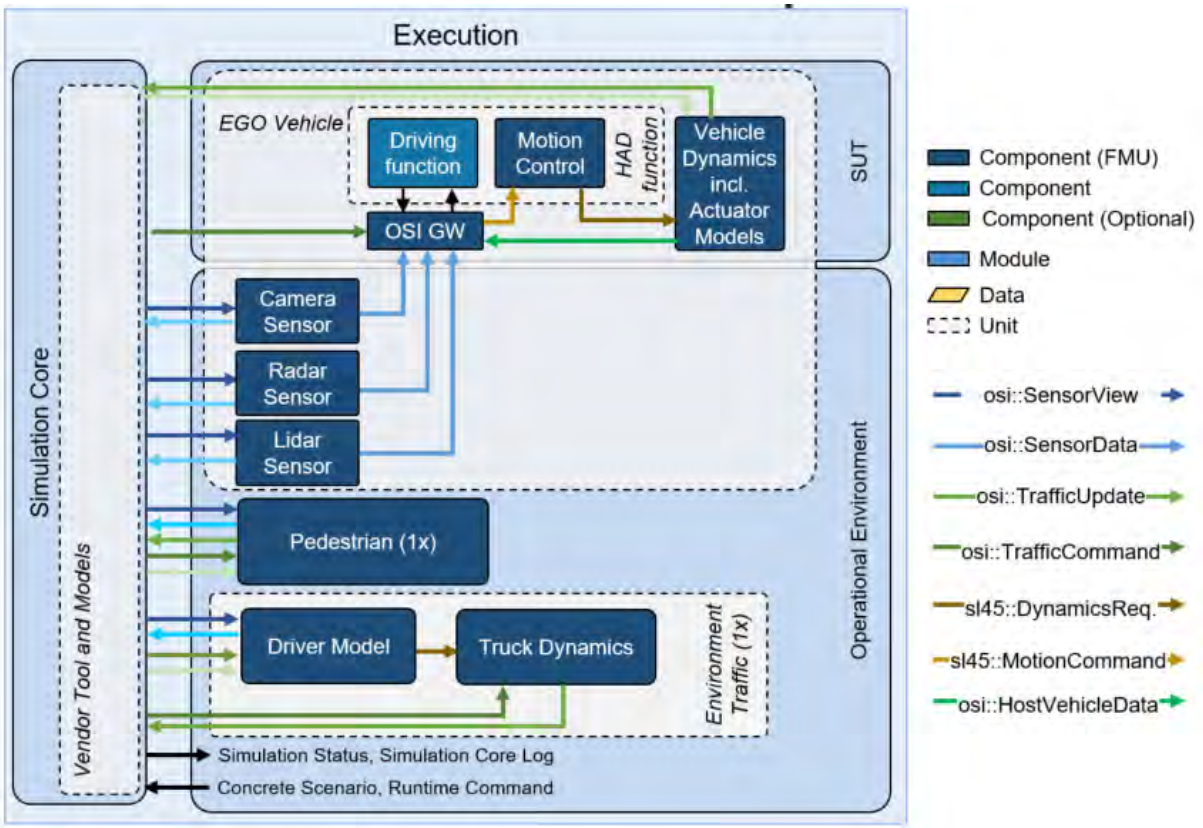


Abbildung 173: dSPACE Simulationsarchitektur für den SUC 2 mit FZI HAD Funktion

Die in Abbildung 173 dargestellte Simulationsarchitektur zeigt den modularen Aufbau der Co-Simulation die aus bis zu 9 Co-Simulation FMUs (FMI 2.0) besteht, die ausschließlich über OSI-Nachrichten kommunizieren. Dies zeichnet auch die Neuartigkeit dieses Ansatzes aus. Da sämtliche FMUs auch im Rahmen des Projekts von verschiedenen Partnern entwickelt wurden, kam es zu zahlreichen Iterationen und Inbetriebnahmen.
Die Anwendung wurde in der dSPACE-Toolkette Release 2020-A auf einem Windows 10-System umgesetzt. Den Kern dieser Umsetzung bilden die Automotive Simulation Models (ASM) als Toolsuite für modellbasierten Test und Entwicklung von Steuergeräten. Die ASM-Modelle basieren auf MATLAB/Simulink. Für dieses Projekt wird das Simulationsmodell ASM Traffic genutzt, welches neben der Simulation eines Personenwagens auch seine Interaktion mit der Umgebung und umliegendem Verkehr modelliert.
Zur Konfiguration und Parametrierung des ASM-Traffic-Modells kommt die Anwendung dSPACE ModelDesk zum Einsatz. In ModelDesk werden insbesondere das verwendete Straßennetzwerk und die Szenariobeschreibung für das Ego-Fahrzeug und die übrigen Verkehrsteilnehmer konfiguriert. Außerdem wird die Simulation über ModelDesk auf die Zielplattform heruntergeladen und gesteuert. Um den Ablauf des simulierten Szenarios nachvollziehen zu können, wird die Simulation während der Laufzeit mit dSPACE MotionDesk in einer 3D-Welt visualisiert.
Um die externen Modelle in die Simulation einzubinden, hat dSPACE das mit ihrem Produkt ausgelieferte Simulationsmodel ASM Traffic in Simulink um einen OSI-Adapter ergänzt, der aus den OSI-*GroundTruth*-Daten, die im Standardproduktumfang enthalten sind, für die Sensoren nutzbare OSI-*SensorView*-Nachrichten erzeugt. Außerdem wurde ASM Traffic angepasst, um den Bewegungsbefehl an die HAD-Funktion bereitzustellen. Zusätzlich hat dSPACE eine Komponente entwickelt, um die aktuellen Positions- und Bewegungsdaten des externen Fahrzeugmodells in das Simulationsmodell und den OSI-Adapter einzubinden und somit eine „Closed-Loop-Simulation“ zu erreichen:

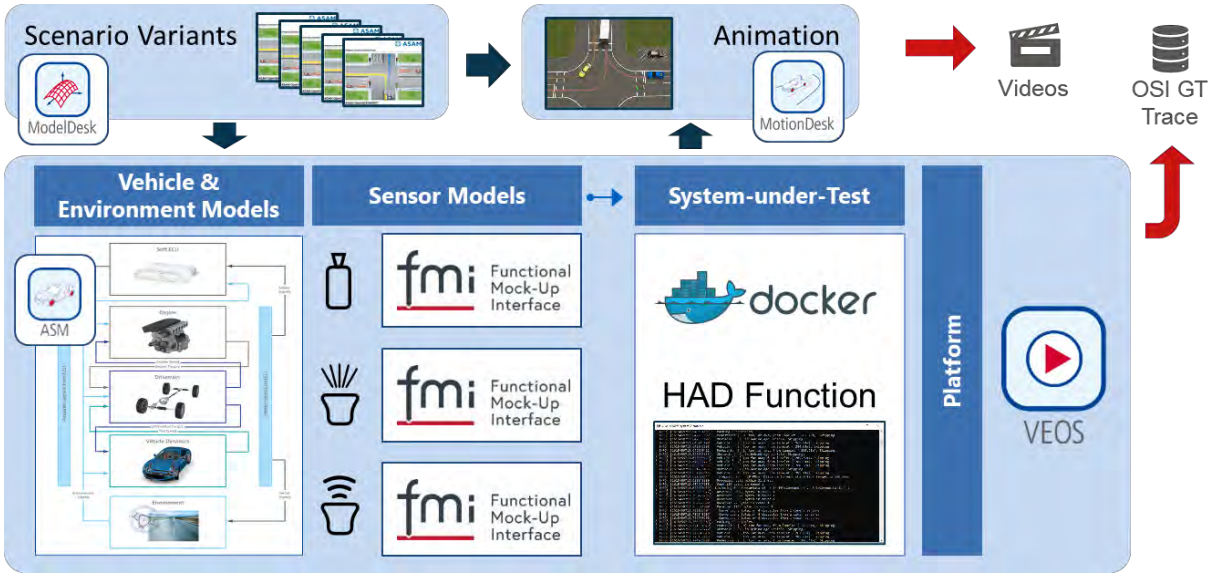


Abbildung 174: dSPACE Toolkette für SUC 2 mit FZI HAD Funktion

Die HAD-Funktion wird als Docker-Container integriert, der als Linux-Ubuntu-System läuft. Als Laufzeitumgebung unter Windows dient Docker for Desktop.
Abbildung 175 zeigt ein Bild der Tooldemo auf dem Halbzeitevent. Auf der linken Seite die Konsolenausgabe des FZI HAD Dockers und auf der rechten Seite die 3D Szene in MotionDesk.

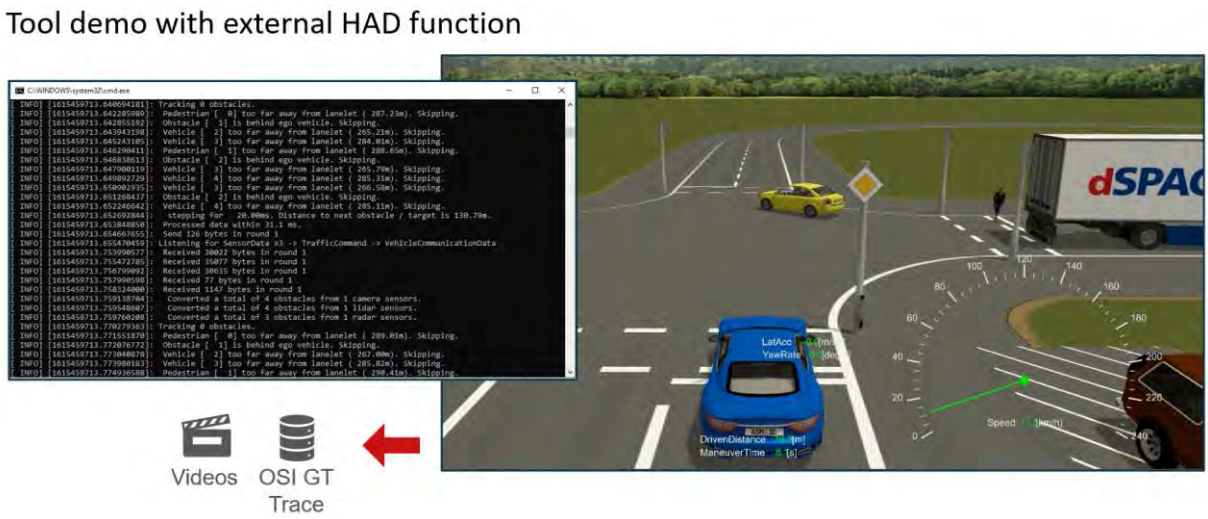


Abbildung 175: dSPACE Simulation für SUC 2 mit FZI HAD Funktion

Zur nachträglichen Auswertung eines Simulationslauf wird laufend die OSI-*SensorView*-Nachricht eines beliebigen Sensors als OSI Trace File auf der Festplatte abgespeichert. Dieses Trace File enthält den kompletten Nachrichteninhalt der einzelnen Simulationsschritte. Im Post-Processing werden aus diesen umfassenden Daten die Zeitreihen extrahiert, die für die Berechnung der Metriken zur Auswertung benötigt sind. Diese Zeitreihen werden in Tabellenform als CSV-Datei abgespeichert. Die CSV-Tabellen werden als Eingabe für die Python-Skripte genutzt, um die simulierten Szenarien auszuwerten (Metriken, Test Verdicts).

dSPACE Integration MAN Motion Planner

Neben der Integration der FZI HAD Funktion hat dSPACE intensiv mit dem Projektpartner MAN zusammengearbeitet, um eine alternative Integration für den Nutzfahrzeugbereich zu erproben. Dazu werden die Simulationsarchitektur und Modellschnittstellen auf eine von MAN entwickelte GPU-basierte Fahrfunktion (MAN Motion Planner) übertragen, die mit der dSPACE Umgebungssimulation gekoppelt werden soll. Die dabei verwendete Simulationsarchitektur (vgl. Abbildung 176) ist dabei sehr ähnlich wie die für die FZI HAD Funktion. Allerdings wird nur ein idealer Sensor verwendet und der Testgegenstand (SuT) ist anders geschnitten. SuT ist der Motion Planner Algorithmus und nicht das gesamt automatisierte Fahrzeug (Sensor, Vehicle Dynamics, Actuator, HAD).

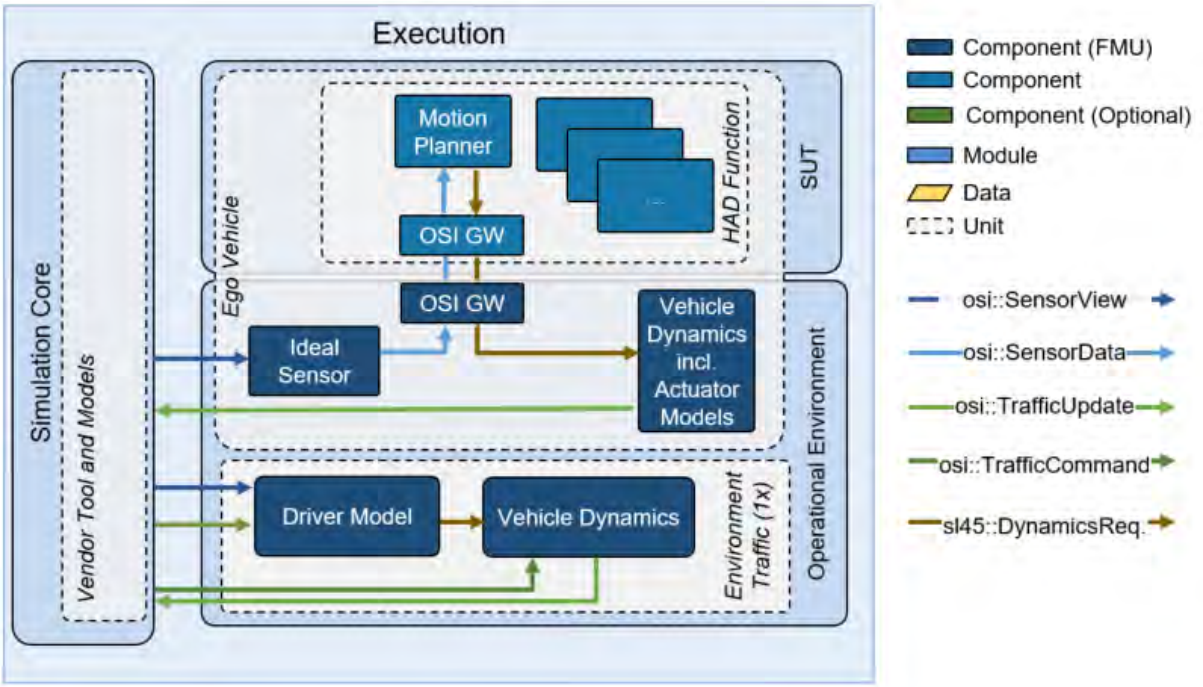


Abbildung 176: dSPACE Simulationsarchitektur für den SUC 2 mit MAN Motion Planner

Weiterer Unterschied zur FZI HAD Integration ist die durchgeführte Integration in dSPACE SIMPHERA. SIMPHERA ist eine web-basierte Cloud-Lösung für hochskalierbare Simulation und Validierung von kritischen Systemen. Diese Einbindung ist in Abbildung 177 schematisch dargestellt. Die Integration selbst baut auf denselben Technologien auf, die auch in der klassischen dSPACE-Toolkette zum Einsatz kommen und oben bereits beschrieben sind. Jedoch wird nun auch die Umgebungssimulation (Operational Environment) als Docker bereitgestellt. Der Operational Environment Docker (VEOS Simulation) und der SuT Docker (MAN Motion Planner) werden dann in der Azure Cloud ausgeführt und kommunizieren über den OSI Gateway (OSI GW) genau wie in der FZI Integration. Hervorzuheben ist, dass sowohl SuT als auch Operational Environment (VEOS, ASM) unter dem Linux-Betriebssystem laufen.

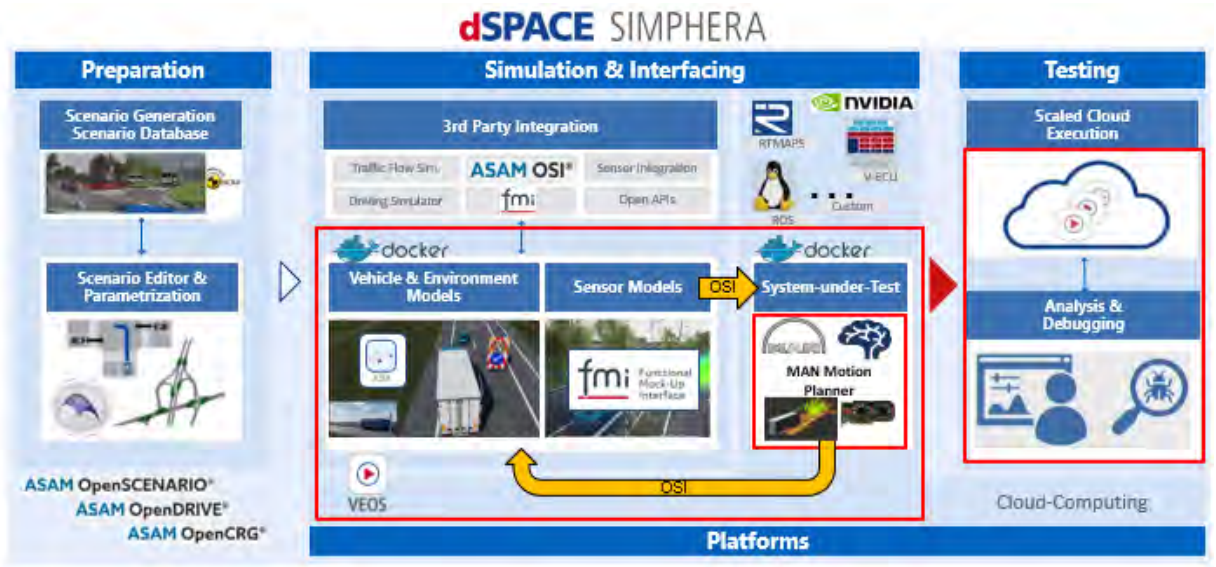


Abbildung 177: dSPACE SIMPHERA für den SUC 2 mit MAN Motion Planner

Zur Simulation wurde die Karte (OpenDRIVE) der MAN Teststrecke in Karlsfeld verwendet. Als Szenario wurde ein Ausweichmanöver eines Trucks (EGO Fahrzeug mit Motion Planner) mit gleichzeitig überholendem PKW definiert (siehe Abbildung 178).

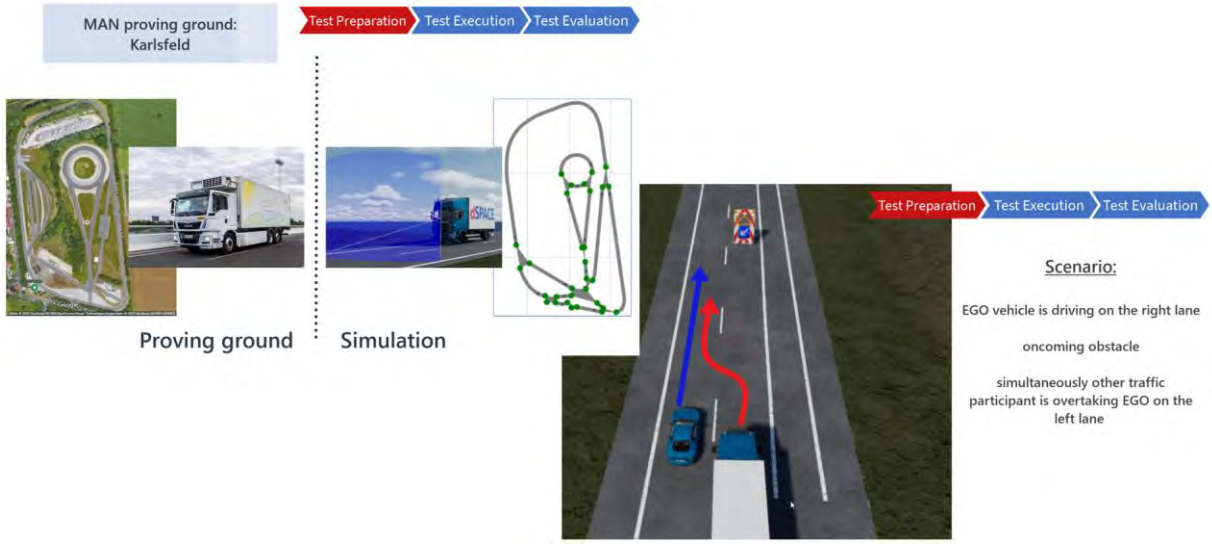


Abbildung 178: Map und Szenario für den SUC 2 mit MAN Motion Planner

Die Anfangsposition des Trucks (Ego), die Anfangsgeschwindigkeit des Trucks und die Geschwindigkeit des überholenden PKW (Fellow) wurden innerhalb definierter Intervalle mit einer „Cross Variation“ variiert. Bei der Cross Variation werden die Parameter einzeln variiert, während die anderen Parameter konstant gehalten werden. Dadurch ergeben sich hier mit der Vorgabe der Schrittweite 141 Variationen (Jobs) zur Simulation (siehe Abbildung 179).

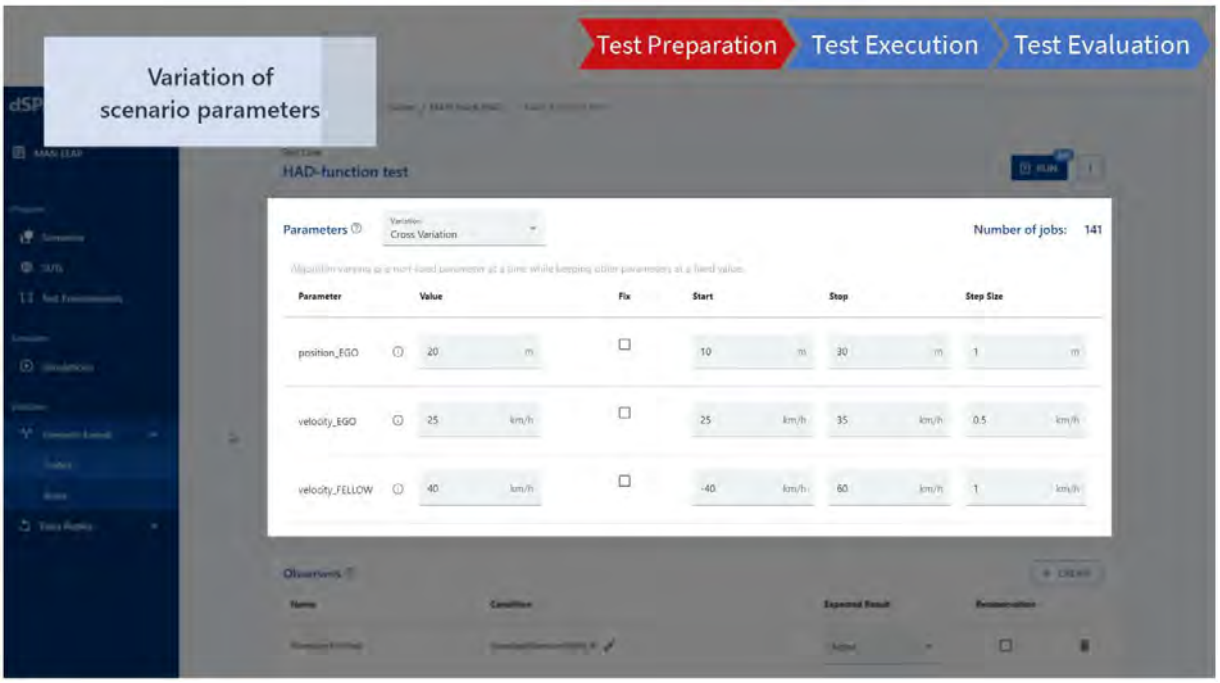


Abbildung 179: dSPACE SIMPHERA Parametervariation für den SUC 2 mit MAN Motion Planner

Als Metrik zur Bewertung wurde der TTC Wert (Time-to-Collision) zwischen Truck und PKW betrachtet, Situationen unter einem definierbaren Schwellwert führen zu negativen Testergebnissen. In Abbildung 180 sind alle Simulation in der Übersicht dargestellt. Negative Tests sind im 3D-Parameterraum (rechts) mit roten Punkten markiert. Für diese Fälle kann nun ein Motion Planner Entwickler nachfolgend prüfen, ob hier der Motion Planner Algorithmus weiter verbessert werden muss.

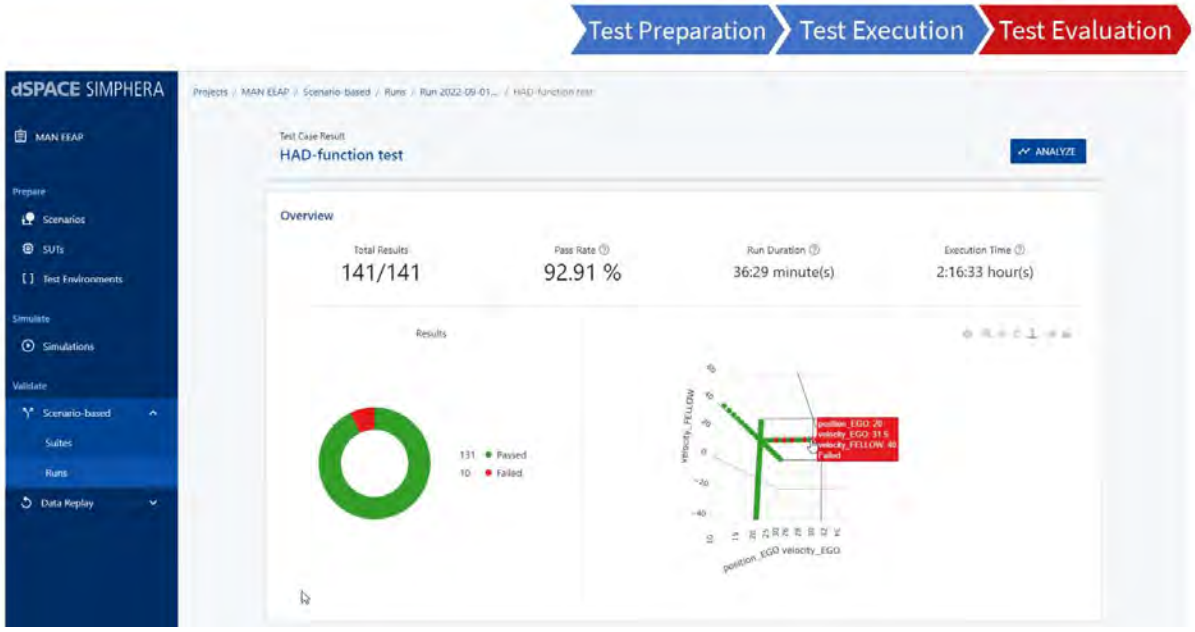


Abbildung 180: dSPACE SIMPHERA Test Ergebnisse für den SUC 2 mit MAN Motion Planner

Die beschriebene Integration wurde auf dem Abschlussevent in Form einer Tooldemo von dSPACE vorgeführt (https://setlevel.de/en/final-presentation/booths/suc-2-test-of-man-motion-planner-in-the-cloud-with-simphera). Insgesamt konnte gezeigt werden, dass der SET Level Ansatz (Kombination OpenX, OSI, FMI, SET Level Simulationsarchitektur) auch in einer Cloud-Umgebung und damit hochskalierbar einsetzbar ist. Zusätzlich wurde exemplarisch

gezeigt, dass das SET Level Konzept auch in der Entwicklung von Nutzfahrzeugen anwendbar ist.

2.3.3.3.4 Umsetzung in den Werkzeugen von IPG

IPG Automotive unterstütze über die Projektlaufzeit die Integration der von den Projektpartnern bereitgestellten Modelle in die Simulationsplattform CarMaker. Diese bietet eine vollständige und umfassende Modellierungs- und Integrationsumgebung, die den virtuellen Fahrversuch mit den Methoden Model-, Software-, Hardware- und Vehicle-in-the-Loop unterstützt (siehe Abbildung 181). Zur Integration der Modelle der Projektpartner wurden folglich überwiegend existierende Modelle in CarMaker ersetzt und die im Konsortium abgestimmten Schnittstellen entwickelt und bereitgestellt.

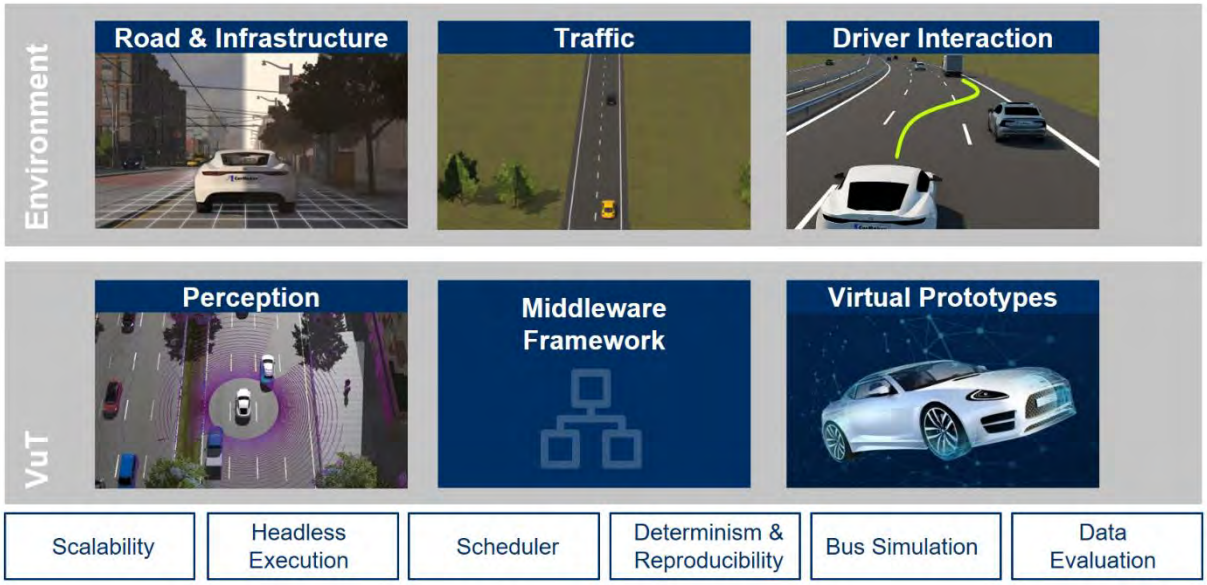


Abbildung 181: Hauptfunktionalitäten der Simulationsplattform CarMaker

Die finale Ausbaustufe der Simulationsarchitektur ist in Abbildung 182 dargestellt. Szenarienbeschreibungen werden in den Standardformaten OpenSCENARIO und OpenDRIVE bereitgestellt, die dann in einem automatisierten Prozess in die proprietären Formate TestRun und Road5 überführt und im Projektverzeichnis abgelegt werden. Diese Szenarien können dann entweder im CarMaker User Interface manuell ausgewählt werden, oder im Rahmen von Testautomatisierungsumgebungen (z. B. auf lokalen Rechenclustern oder in der Cloud) skriptbasiert gestartet werden. Der Simulationskern enthält alle nötigen internen Modelle und die Schnittstellen zu externen Modellen wie im Projektkontext verwendete OSI-FMUs für Sensormodelle, Bewegungsregler und Fahrdynamikmodelle oder die über einen OSI Network Proxy angebundene HAD-Funktion des FZI. Simulationsergebnisse werden nach der Simulation als CSV-Dateien abgelegt, um die Metriken zur Bewertung ableiten zu können.
Während oder nach einer Simulation kann der zeitliche Verlauf des Testszenarios in der 3D-Visualisierung IPGMovie betrachtet werden oder die Entwicklung einzelner Signale in IPGControl verfolgt werden.

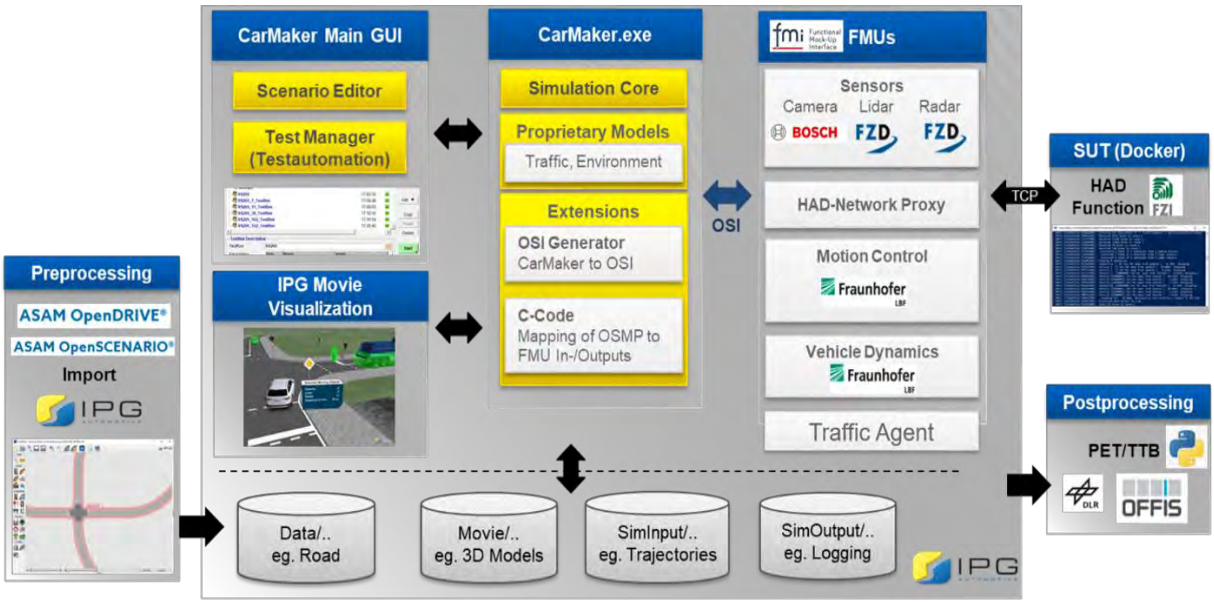


Abbildung 182: Werkzeugkette von IPG Automotive zur Unterstützung der SUCs im Projekt SET Level

Die im Vorgängerprojekt PEGASUS entwickelte OSI-Erweiterung von CarMaker, die die GroundTruth für objektbasierte Sensormodelle bereitstellte, wurde im Projekt SET Level auf veränderte Spezifikationen angepasst sowie um weitere OSI-Nachrichten erweitert. So werden nun auch die Nachrichtentypen OSI::SensorView, OSI::TrafficCommand und OSI::GroundTruthInit von CarMaker bereitgestellt. Außerdem werden die Nachrichten OSI::SensorViewConfiguration und OSI::TrafficUpdate von CarMaker verarbeitet. SET Level-interne Erweiterungen von OSI wurden nur von externen Modellen untereinander ausgetauscht, weshalb CarMaker an dieser Stelle die Orchestrierung des Nachrichtenaustausches als Integrationsplattform übernimmt:

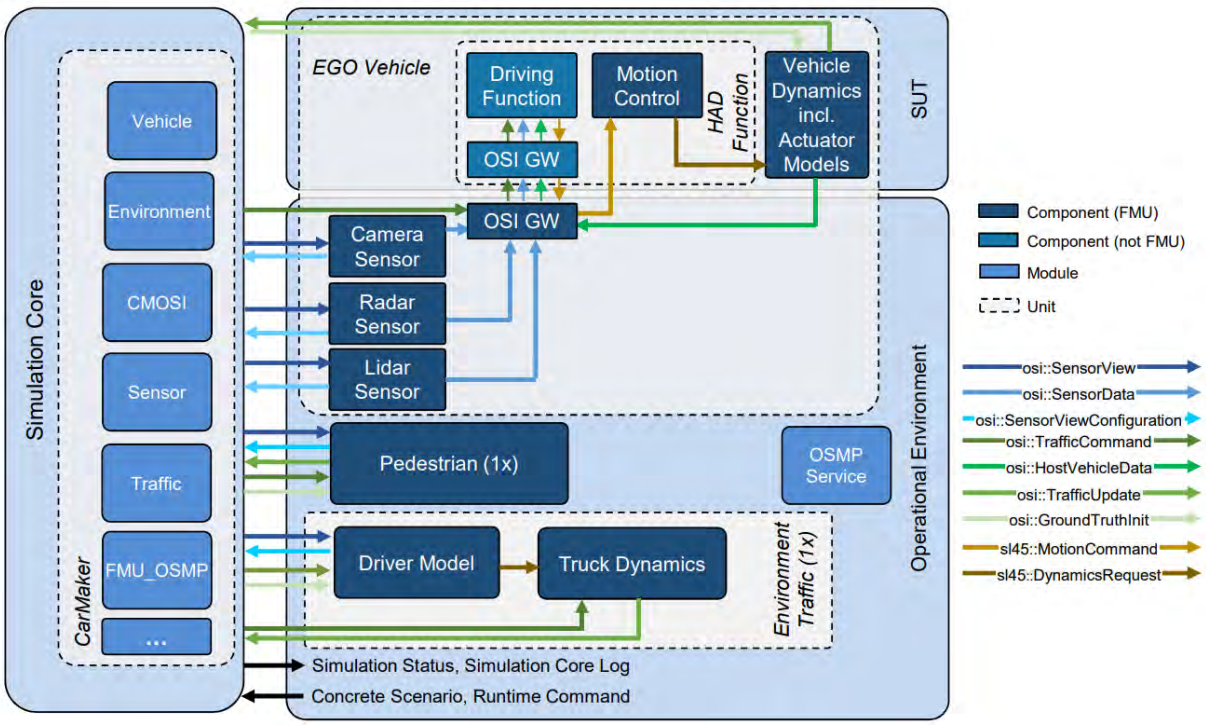


Abbildung 183: CarMaker-Simulationsarchitektur für den SUC 2 mit FZI HAD Funktion

Vor der Inbetriebnahme der Simulation sind die einzelnen Modelle in den CarMaker-Simulationszyklus zu integrieren. Dies ist ein Prozess, der für eine gegebene Konfiguration nur einmal durchzuführen ist – danach kann die Plattform für eine beliebige Anzahl von Tests verwendet werden. Abbildung 184 zeigt die Integration der FMUs in der Benutzeroberfläche. An dieser Stelle sind die Inputs und Outputs der FMUs (Pointer zu den Speicheradressen und Angaben zur Größe des Nachrichtenbuffers) mit der CarMaker-Simulationsumgebung zu verbinden.
Die Konfiguration der zu übermittelnden Botschaften in diesen Speicherbereichen wird auf C-Code-Ebene durchgeführt. An dieser Stelle kann auf die von IPG Automotive bereitgestellte OSI-Bibliothek für CarMaker zurückgegriffen werden, die die OSI-Standardnachrichten zusammenstellt und serialisiert, oder auch eine individuelle Nachricht formuliert werden. Die OSI-Bibliothek für CarMaker ist neben dem automatischen Import von OpenDRIVE und OpenSCENARIO eines der Hauptergebnisse des Projektes, da sie aufgrund der Standardkonformität künftig auch außerhalb des Projektes eingesetzt werden kann.

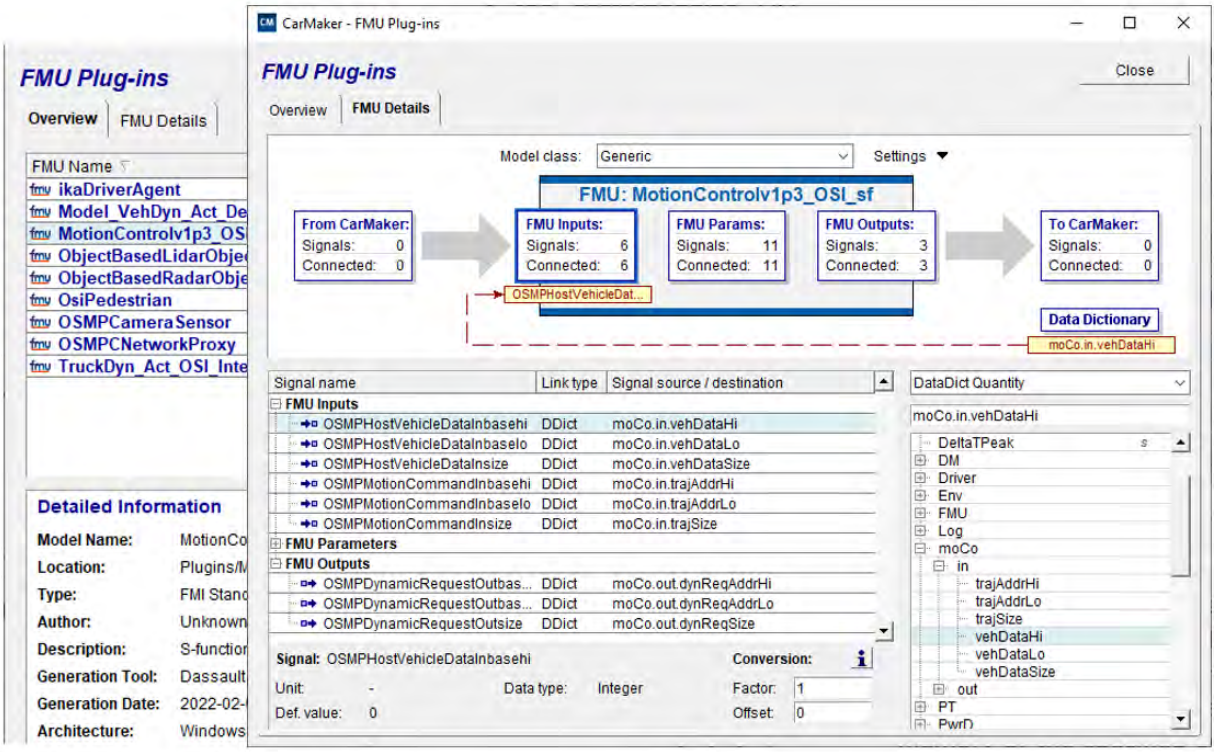


Abbildung 184: Integration der OSI-FMUs in CarMaker

Die erfolgreiche Umsetzung der SUC 2-Architektur konnte in exemplarischen Szenarien dargestellt werden (siehe Abbildung 185). Wie vorgesehen, wurde das Vehicle-under-Test von der FZI-HAD-Funktion geführt, die auf OSI-Sensordaten zurückgegriffen hat. Während des Szenarios traf es auf andere Verkehrsteilnehmer (Fußgänger und LKW), die ebenfalls durch OSI-FMUs modelliert wurden und TrafficCommands von CarMaker entsprechend der Manöverpläne im Szenario erhalten haben. Somit konnte die technische Umsetzbarkeit und praktische Anwendbarkeit der OSI-Schnittstelle und deren Weiterentwicklungen und Erweiterungen bewiesen werden.

Abbildung 185: Ausführung des SUC 2 Szenarios in CarMaker

2.3.3.3.5 Bewertung der Ergebnisse

Der Simulation Use Case 2 diente, ebenso wie die anderen beiden SUCs, im Laufe des Projektes als methodische Leitplanke für die praktische Erprobung einer Vielzahl von Projektaktivitäten, welche einleitend bereits in Abschnitt 2.3.3.3.1 benannt wurden. Insofern ist der Kompetenzaufbau aller am SUC 2 Beteiligten ein nicht gering zu bewertendes Ergebnis dieser Aktivitäten. Unter anderem konnten die Erstellung und Einbindung von FMUs und die dafür notwendigen Abstimmungsprozesse zwischen Tool-Vendoren und Modelllieferanten erprobt und verbessert werden. Eine der größten Herausforderungen war hierbei die Erstellung von FMUs aus Simulink-Modellen (siehe Abschnitt 2.2.3.4.2 Simulink OSI Wrapper).
Auch die Funktion der zumeist als Open Source Varianten umgesetzten Modelle, die im Projekt (weiter-) entwickelt wurden, konnte anhand der gewählten Szenarien getestet werden und den Entwicklern direkte Informationen für die weiteren Implementierungen geben.
Die während dieses Prozesses gewonnen Erkenntnisse im Bereich der Standards FMI und OSI konnten z. T. direkt in die Standardisierungsaktivitäten eingebracht werden und zu erweiterten neuen Version führen. Aber auch die Diskussionen rund um den relativ neuen Standard OpenSCENARIO sind durch die praktische Anwendung positiv vorangebracht worden, so dass u. a. die anfangs noch proprietär gelöste Generierung von konkreten Szenarien aus logischen Szenarien im späteren Verlauf des Projektes als Teil des weiterentwickelten Standards verwendet werden konnte.
Nicht zuletzt war die Weiterentwicklung von unterschiedlichen Werkzeugketten anhand des SUC 2 ein zentraler Aspekt, welcher für die drei in den vorangehenden Abschnitten ausführlich beschriebenen Werkzeugketten erfolgreich umgesetzt werden konnte.
Zusammenfassend kann gesagt werden, dass sich die Einführung der Simulation Use Cases im Allgemeinen, und des SUC 2 im Speziellen für die Erprobung und Demonstration von Closed-loop Integrationstests, als ein sinnvolles Element im Gesamtgefüge des Projektes erwiesen hat.

2.3.3.4 Open-Loop Komponententests

Das übergreifende Entwicklungsziel des SUC 3 (Open-Loop Komponenten-Tests) ist, dass die zu entwickelnde Sensor Komponente ihren Anteil der Umweltwahrnehmung leistet. Diese Leistungsfähigkeit wird während des Entwicklungsprozesses kontinuierlich getestet und ist letztendlich ein Teil der Absicherung.

Wie man in Abbildung 186 sehen kann, wird der SUC 3 für Optimierung und Test im Bereich Komponenten und Verkehrsknoten eingesetzt.

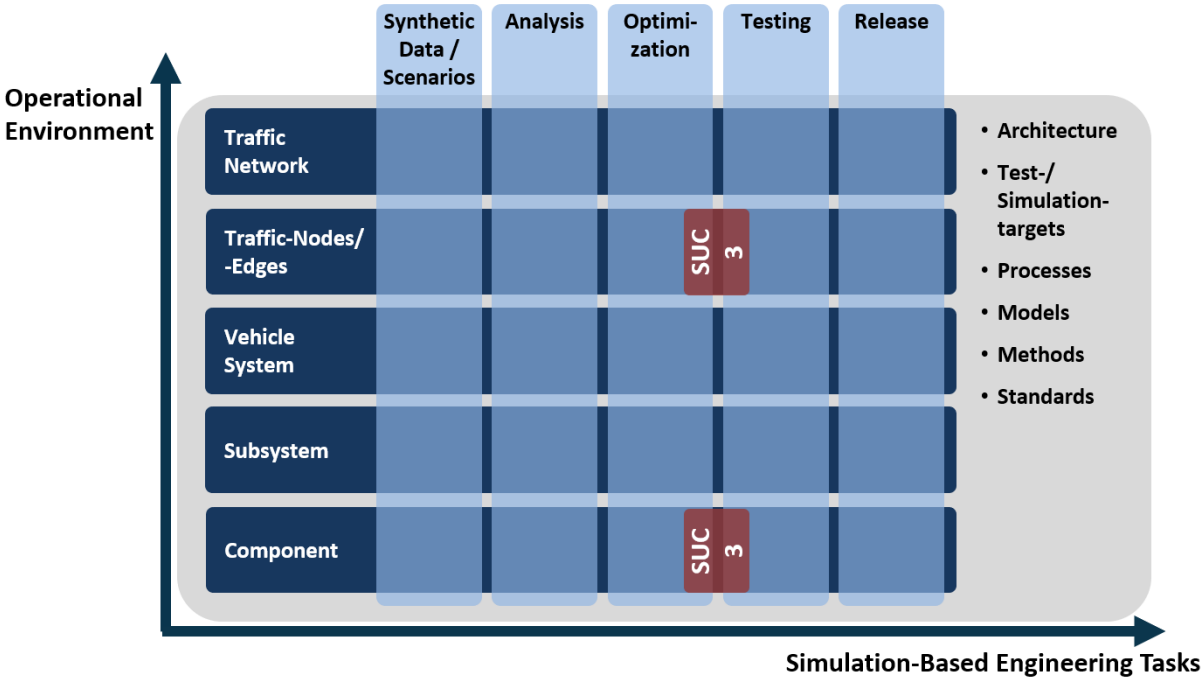


Abbildung 186: Die Verortung des SUC 3 im der SET Level Matrix

Die Abbildung 187 zeigt die Verortung des SUC 3 im V-Modell. Der SUC 3 wird im V-Modell überwiegend in den Bereichen der Implementierung der Sensor-Komponenten und dem Testen von Sensor-Komponenten eingesetzt.

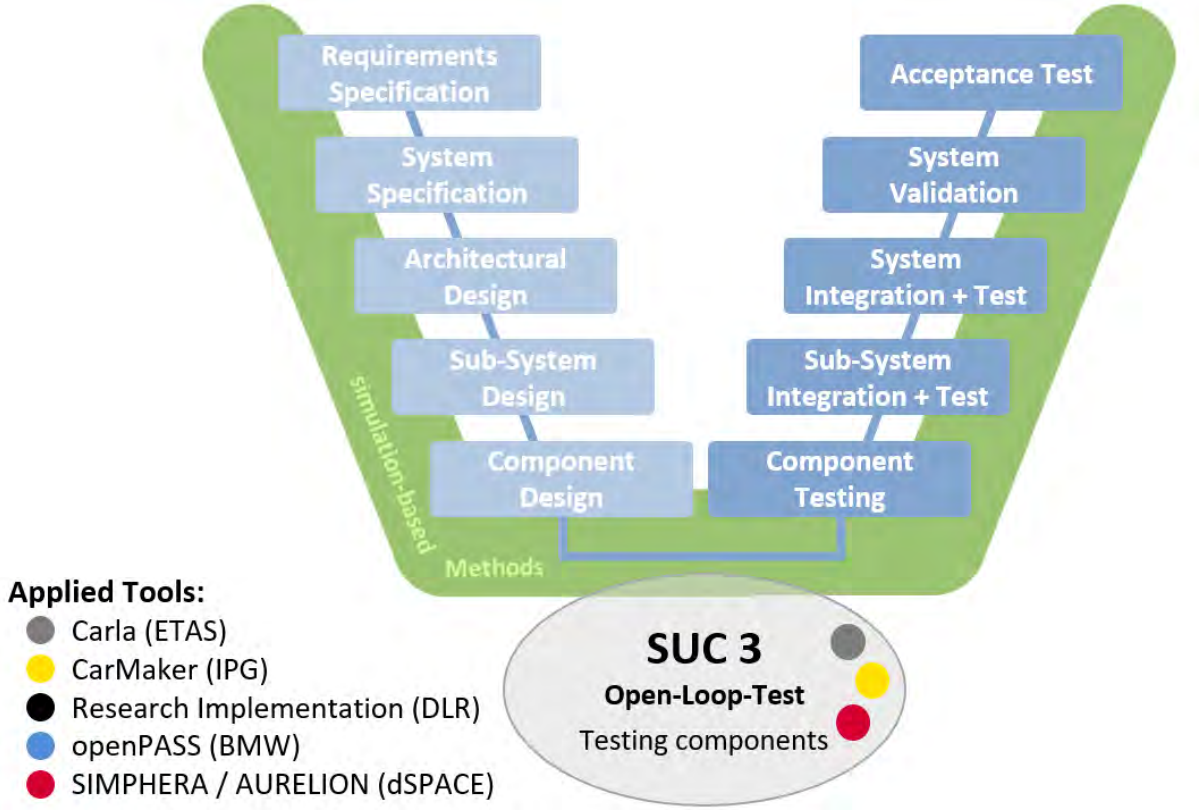


Abbildung 187: Die Verortung des SUC 3 im V-Modell

Es wurden in SUC 3 verschiedenste Werkzeuge eingesetzt. Aus Abbildung 188 geht hervor, welche der im Projekt SET Level verwendeten Artefakte in SUC 3 eingesetzt werden. Im SUC 3 werden nur Komponenten-Modelle, Basis-Artefakte sowie Basis-Prozesse eingesetzt.

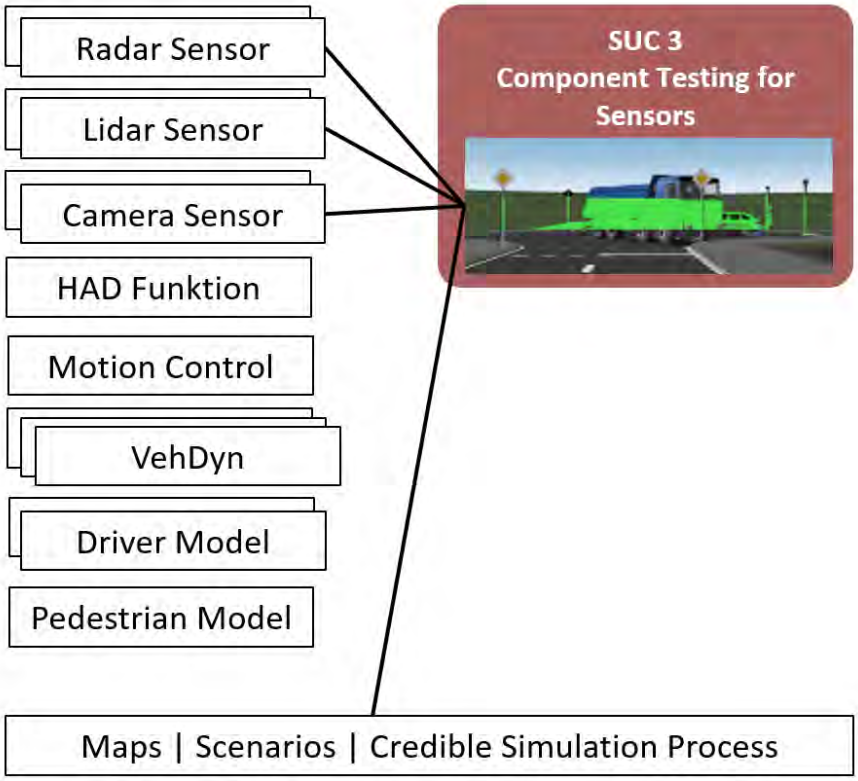


Abbildung 188: Übersicht der Auswahl der Modelle, Artefakte und Prozesse für SUC 3

Die im Projekt SET Level entwickelte und definierte generische Plattformarchitektur, siehe Abbildung 189 diente als Vorlage oder Basis zur Definition der speziellen „Open-loop“ Architektur des SUC 3, siehe Abbildung 190.

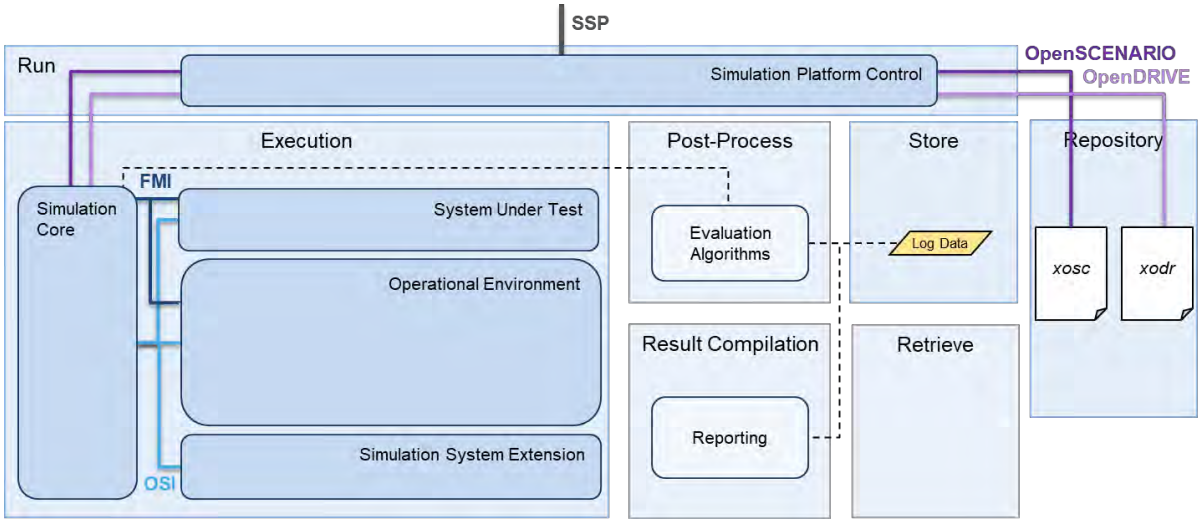


Abbildung 189: Generische Plattformarchitektur

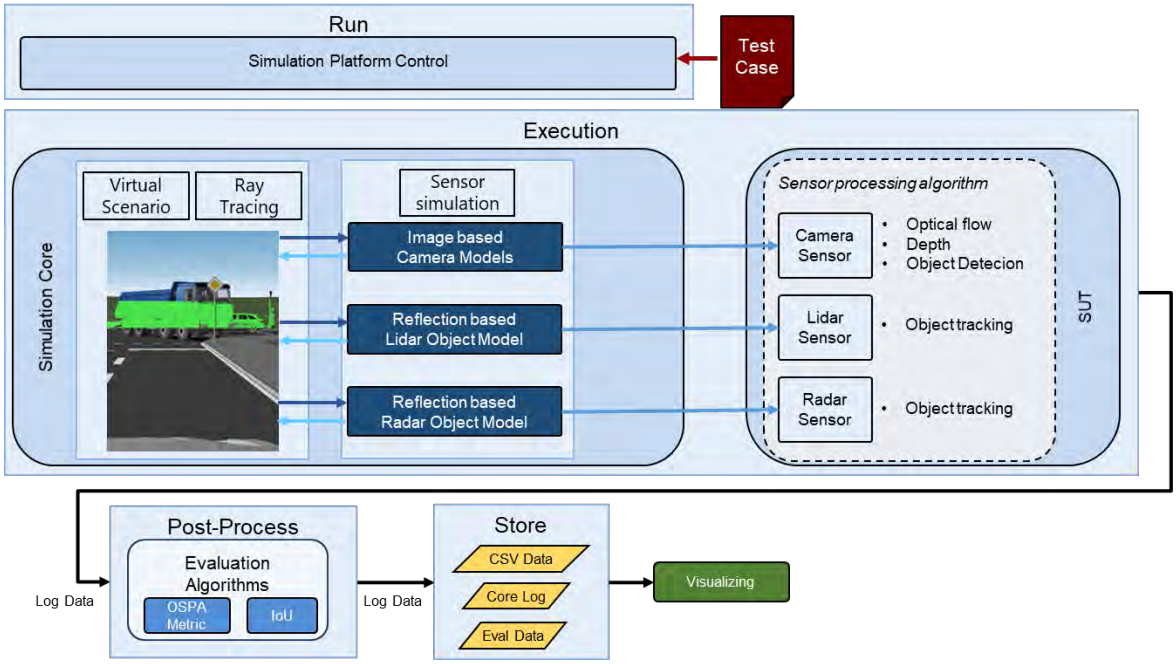


Abbildung 190: Open-Loop Architektur (SUC 3-Architektur)

Die generische Plattformarchitektur von SET Level hat verschiedene Vorteile:

1. Gemeinsames projekt-übergreifendes Verständnis des Simulationsaufbaus
2. Definition logischer Architekturelemente und entsprechender Interaktionen
3. Ermöglicht es spezifische Simulationsaufgaben in SUC 3 detailliert anzupassen

SUC 3 wurde auf Basis des CSP entwickelt und umgesetzt. Es erfolgten konkrete Implementierungen und praktische Anwendungen unterschiedlichster Projektergebnisse im Rahmen mehrerer Implementierungen in unterschiedlichen Werkzeugketten (dSPACE, IPG Automotive, ETAS, Open Source CARLA-basiert).
SUC 3 wurde im Halbzeitevent und im Abschlussevent der Öffentlichkeit präsentiert. Durch SUC 3 erfolgte eine erfolgreiche interne Erprobung von anderen Projektaktivitäten wie dem „Credible Simulation Process (CSP)“, Standardisierungen, Anwendung desselben Szenarios für unterschiedliche Sensortechnologien, unterschiedliche Sensorauswertung auf gleichem Rohdatensatz und der Modellintegration.

2.3.3.4.1 Validierung einer isolierten Komponente ohne Wechselwirkung

Thema und Problemstellung
In AP 4.4 erfolgte die Umsetzung des Simulation Use Case 3 (SUC 3). Die Demonstrationsziele der Open-Loop Simulationen des SUC 3 wurden analysiert, ausgeplant und in der Projektlaufzeit umgesetzt. Die Werkzeugtoolketten für die drei physikalischen Sensorsimulationen (Kamera, LIDAR, Radar) wurde während der Projektlaufzeit in Verbindung mit den geeigneten physikalischen Modellen eingesetzt und weiterentwickelt. Die Aufgaben, die beteiligten SET Level-Partner sowie die Vorgehensweise sind hierbei gleichgeblieben. Die geplanten Aufgaben für MS1, MS 2 und MS 3 konnten bis zum Projektende abgeschlossen werden. Aus personellen und technischen Gründen wurden im Bereich Radar einzelne geplante Ziele neu priorisiert und umgeplant.
Bekanntermaßen geht es im SUC 3 darum, ein Ego-Fahrzeug, bestückt mit Kamera-, LiDAR- und Radar-Sensoren, durch ein bestimmtes Szenario, zukünftig auch durch ein beliebiges durch die Verwendung von Standards definiertes Szenario, fahren zu lassen. Die mit den Sensoren erkannten Objekte werden über sensor-spezifische Metriken mit der absoluten

(korrekten) Objektliste verglichen und bewertet. Dies erlaubt es, Verbesserungen sowie auch Verschlechterungen zu erkennen und somit simulationsbasiert entwickeln und testen zu können. Dies ist das gesetzte Ziel von SET Level.
Im Folgenden wird im Detail auf die 3 verschiedenen Sensortechnologien sowie auf OpenMATERIAL eingegangen.
Die Verwendung der beiden Sensortechnologien Kamera und LiDAR konnte durch gut funktionierende Modelle und die dSPACE- und IPG Automotive-Toolketten wie geplant umgesetzt werden.
Die Modell-Implementierung der Radar-Sensortechnologie erfolgte durch zwei miteinander gekoppelte Modelle. Wegen der Verwendung des zukünftig noch zu etablierenden „Standards“ OpenMATERIAL wurde der Raytracer, anstatt in der Werkzeugkette, im Modell implementiert.
Diese beiden zusätzlichen Innovationen (OpenMATERIAL und zwei getrennte Modelle mit internem Raytracer) stellten sehr hohe Anforderungen an die Flexibilität des Prozesses, der Werkzeuge, der Simulations-Architektur und der beteiligten Mitarbeiter. Aus diesen Gründen mussten die angestrebten Ziele mehrfach angepasst und priorisiert werden. Beispielweise wurde die Machbarkeit der Integration durch eine prototypische Implementierung demonstriert. Auf eine umfangreiche Materialdatenbank sowie validierte Materialparameter wurde verzichtet. Trotzdem, oder vor allem deshalb, konnten die Machbarkeit der angestrebten Konzepte gezeigt werden.

2.3.3.4.2 Allgemeines zur Umsetzung der Kamera-Simulation

Im Rahmen des Abschlussevents wurden die User Stories innerhalb des SUC 3 auf die drei Toolvendoren aufgeteilt. So werden für den Zuschauer thematische Dopplungen vermieden und trotzdem jedem Tool-Hersteller und Sensormodellentwickler die Möglichkeit gegeben, seine Arbeit zu präsentieren. Die Umsetzung der Bosch Kamera User Story erfolgte hierbei in den dSPACE Tools.
Anhand des für das Abschlussevent erzeugten Materials wird nachfolgend eine Übersicht über die Umsetzung der Kamera-Simulation gegeben.

SUC3 – Open-Loop Sensor Simulation for Component Test

Camera-based Tracking

Verification of Sensor Functions

The goal of this Simulation Use Case was to research the possibilities of integrating representative elements of a **video camera sensor system** (optics, imager, **image processing, sensor function**). It was achieved with the help of **standard interfaces** in an open loop simulation chain with synthetic camera data in a high definition environment created with dSPACE Simulation Tools.

Image 1: (dSPACE visualization) Optical flow shown as an overlay, derived from ground truth data. Velocity and direction are encoded in HSV color space.

Image 2: (Bosch visualization) Optical flow overlay from the camera sensor model, encoded in the same color space.

Sensor-Realistic Simulation

For development and testing future automotive systems, **highly realistic sensor simulation** is key. To test autonomous vehicles up to SAE Level 5, it is mandatory to be able to provide sensor data for **every conceivable scenario**. That includes vehicle dynamics, traffic, and sensor-realistic data based on environmental influences.

Image 3: Sensor-realistic simulation for all levels of automation.

Implementation: dSPACE Toolchain

The **dSPACE Automotive Simulation Models (ASM)** implement the concepts of SET Level, which are consumed by **AURELION** to generate sensor-realistic camera data and ground truth information. The resulting data is injected into **Bosch's system-under-test (SUT)** via **Open Simulation Interface (OSI)** Sensor View. To evaluate the function, the output of the SUT is compared to ground truth information.

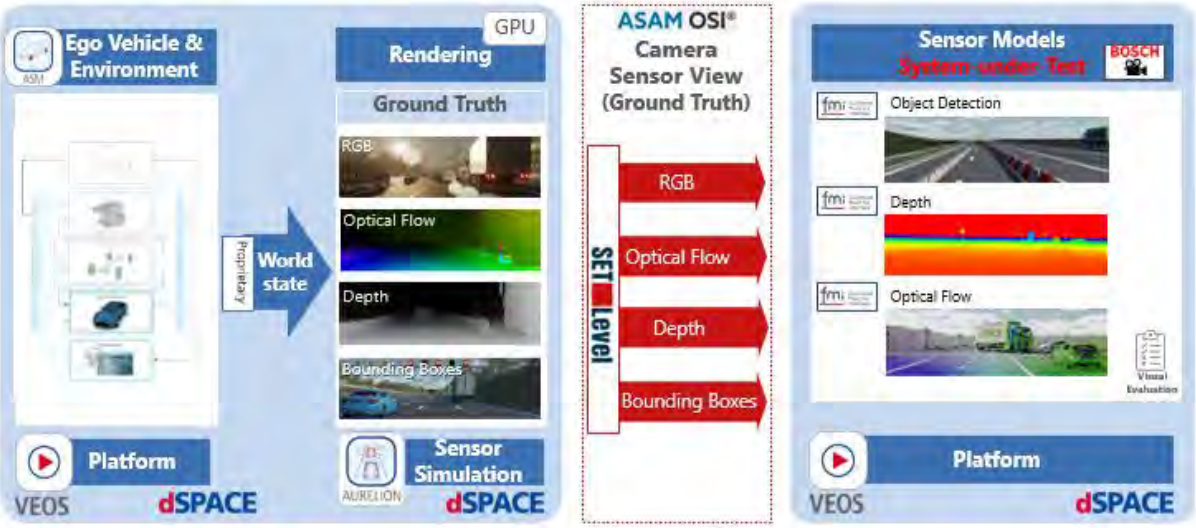


Abbildung 191: Übersichtsbild (Poster) der modularen Werkzeugkette zur Kamera Simulation innerhalb SUC 3

Das obige Übersichtsbild (siehe Abbildung 191) für den Bereich Kamera Simulation gliedert sich in vier Bereiche:

1) Verifikation von Sensorfunktionen
2) Sensorrealistische Simulation
3) Implementierung: dSPACE Werkzeugkette
4) Blockschaltbild

Diese werden im Folgenden kurz beschrieben.

Verifikation von Sensorfunktionen
Ziel dieses Simulation Use Case war es, die Möglichkeiten der Integration repräsentativer Elemente eines Videokamera-Sensorsystems (Optik, Bildverarbeitung, Sensorfunktion) zu erforschen. Dies wurde mit Hilfe von Standardschnittstellen in einer Open-Loop-Simulationskette mit synthetischen Kameradaten in einer hochauflösenden Umgebung erreicht, die mit dSPACE Simulation Tools erstellt wurde.
Abbildung 191 Bild 1: (dSPACE-Visualisierung) Optischer Fluss als Overlay, abgeleitet von Daten aus der GroundTruth. Geschwindigkeit und Richtung werden im HSV-Farbraum kodiert.
Abbildung 191 Bild 2: (Bosch-Visualisierung) Optisches Fluss-Overlay aus dem Kamerasensormodell, kodiert im gleichen Farbraum.

Sensorrealistische Simulation
Für die Entwicklung und Erprobung zukünftiger Automobilsysteme ist eine sehr realistische Sensorsimulation entscheidend. Um autonome Fahrzeuge bis SAE Level 5 zu testen, ist es zwingend erforderlich, Sensordaten für jedes denkbare Szenario bereitstellen zu können. Dazu gehören Fahrzeugdynamik, Verkehr und sensorrealistische Daten auf Basis von Umwelteinflüssen.

Implementierung: dSPACE Werkzeugkette
Die dSPACE Automotive Simulation Models (ASM) setzen die Konzepte von SET Level um, die von AURELION genutzt werden, um sensorrealistische Kameradaten und Ground-Truth-Informationen zu generieren. Die resultierenden Daten werden über Open Simulation Interface (OSI) Sensor View in das System under Test (SuT) von Bosch eingespeist. Um die Funktion zu bewerten, wird die Ausgabe des SuT mit Ground-Truth-Informationen verglichen.

2.3.3.4.3 Umsetzung Kamera-Simulation in Werkzeugen von dSPACE

Die Sensorsimulation des Bosch Kameramodells zog sich wie auf Abbildung 192 dargestellt über die gesamte Projektlaufzeit hinweg. Hierbei wurde zu Beginn die dSPACE Sensorsimulation mit MotionDesk und SensorSim genutzt. Ungefähr ab dem Halbzeitevent erfolgte der Umstieg auf die neue Sensorsimulation mit dSPACE AURELION (siehe Abbildung 192). Die Lidarsimulation wurde zuerst parallel von dSPACE und IPG Automotive aufgebaut, danach erfolgte jedoch aufgrund der steigenden Komplexität die Aufteilung des Kamera Use Case zu dSPACE und des Lidar Use Case zu IPG Automotive.

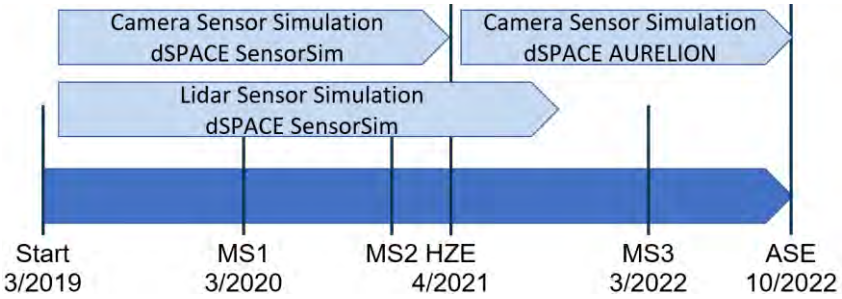


Abbildung 192: Zeitstrahl der dSPACE Open-Loop Sensormodellintegrationen

Die Kommunikation zwischen den dSPACE Tools und den Sensormodellen über die OSI-Schnittstelle erforderte eine Komponente, welche die proprietären Signale, in die des OSI-Standards überführt. Diese im folgenden OSI-Adapter genannte Komponente und der

dazugehörige OSI-Standard wurden während der gesamten Projektlaufzeit stetig weiterentwickelt, um den neu hinzugekommenen Anforderungen zu entsprechen.

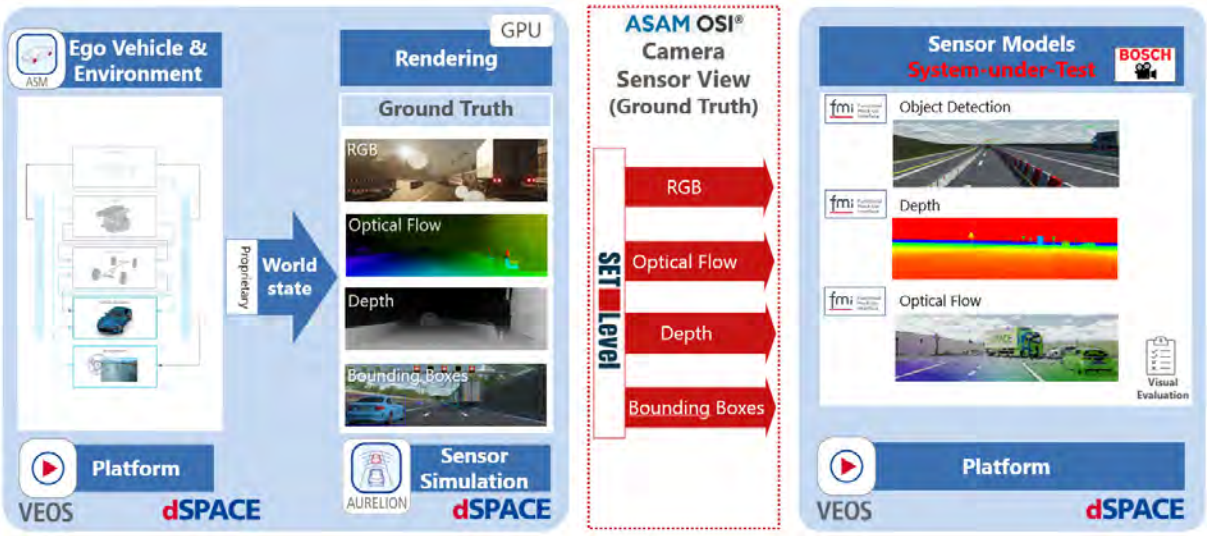


Abbildung 193: dSPACE Simulationsarchitektur für SUC 3 Kamerasensor mit AURELIO

In Abbildung 193 ist der finale Aufbau (Abschlussevent) der dSPACE Simulationsarchitektur dargestellt: Die Automotive Simulation Models (ASM) für die Umgebungssimulation und das externe Sensormodell laufen auf der Simulationsplattform VEOS. Die Sensorrohdaten der Ground Truth werden mit SensorSim/AURELION auf einer leistungsfähigen GPU berechnet und dann dem Sensormodell über den OSI-Adapter zur Verfügung gestellt.
Zum Meilenstein 1 arbeitete das Bosch Sensormodell auf Basis von Ground Truth-Objektlisten. Das Ziel hierbei war es, die übermittelten Objekte mit Bounding Boxes zu kennzeichnen, um sicherzustellen, dass die grundlegende Kommunikation über die OSI-Schnittstelle funktioniert.
Die Weiterentwicklung zum Meilenstein 2 bestand darin, Kamera-Rohdaten, anstatt von Objektlisten zu übermitteln. Hierzu wurde der OSI-Adapter erweitert, sodass das Sensormodell mithilfe der von dSPACE zur Verfügung gestellten RGB-Bilddaten eine Kantendetektion durchführend konnte. Der Schwerpunkt lag hierbei auf dem Testen verschiedener Optiken und der Beurteilung von deren Qualität.
Der 3. Meilenstein brachte die Erweiterung von einer Mono- auf eine Stereokamera, um aus den RGB-Bildern die Tiefe berechnen zu können. So konnte die Ground Truth Tiefeninformation mit der vom Sensormodell berechneten Tiefe verglichen werden, um die Berechnungen des Sensormodells beurteilen zu können.
Zum SET Level Abschlussevent wurde die Objektdetektion mit Darstellung der Bounding Boxes, im Gegensatz zu Meilenstein 1, nun auf Basis der RGB-Daten vorgenommen. Hinzu kam die Berechnung des optischen Flusses, um zusätzliche Informationen zur Bewegung und Geschwindigkeit von Objekten zu erhalten. Auch hier war das Ziel der Abgleich der dSPACE Ground Truth Daten und der vom Sensor berechneten Bounding Boxes bzw. des Flusses.

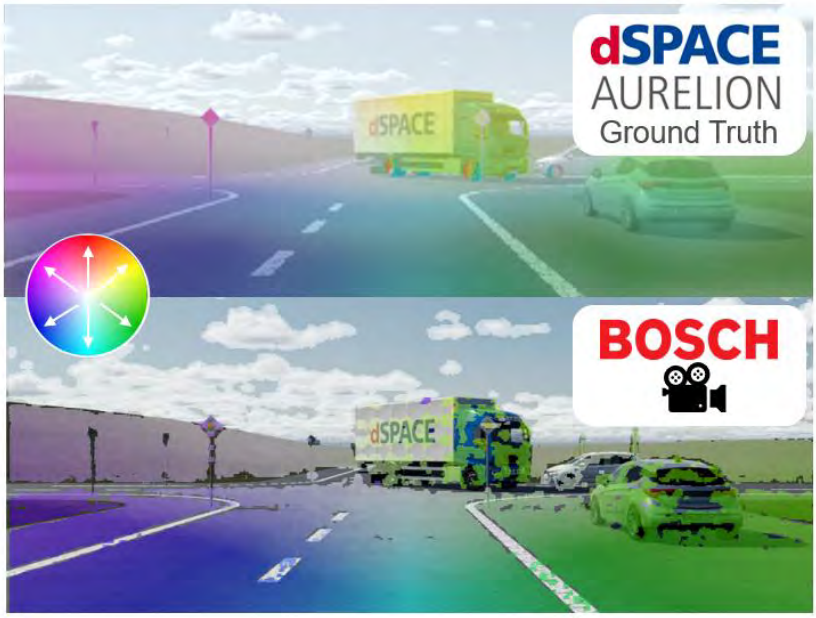


Abbildung 194: Optischer Fluss aus Groundtruth in AURELION als Overlay (oben) und Optischer Fluss im Kamerasensormodell als Overlay (unten)

Die Geschwindigkeit und der Bewegungsvektor sind in Abbildung 194 im HSV-Farbraum kodiert. Das obere Bild zeigt den optischen Fluss für jedes im Bild dargestellte Pixel. Im unteren Teil von Abbildung 194 sieht man den vom Sensormodell berechneten Fluss. Für Bereiche, in denen keine Bewegung identifiziert werden konnte (bspw. große zusammenhängende gleichfarbige Flächen wie die LKW-Plane) entstehen Lücken.
Zusammenfassend lässt sich sagen, dass der OSI-Standard zu Beginn des Projekts im Bereich der Sensor Rohdaten noch eine Lernkurve bei den Projektbeteiligten erforderte. Es bestanden viele Unklarheiten bzgl. der konkreten Anwendung. Im Zuge des Projekts konnte der Standard erfolgreich und anwendungsorientiert angewendet und auch erweitert werden (in Kommunikation mit ASAM). Ein klar definierter Standard spart bei der Integration externer Modelle Zeit und somit Kosten. Beim Umstieg von objektlistenbasierten auf rohdatenbasierte Daten konnte bedingt durch die erheblich gestiegenen Datenmengen (abhängig von Auflösung und Framerate der Kamera) eine deutlich reduzierte Simulationsgeschwindigkeit beobachtet werden. Ob hier eine weitere Optimierung der OSI 3.x Technologie (mit Protobuf) noch Potential zur Geschwindigkeitssteigerung besitzt, ist im Rahmen von SET Level bei dSPACE nicht überprüft worden.

2.3.3.4.4 Umsetzung Kamera-Simulation in Werkzeugen von IPG

Für die Kamera User Story wurde eine synchrone Übertragung der Ground Truth und mehrerer CameraSensorView Nachrichten in den Bildformaten RGB und Tiefenbild mit der CarMaker-Visualisierung IPGMovie umgesetzt. Durch diese Implementierung können, die zur Abbildung von Effekten einer Stereokamera benötigten Größen über die CarMaker OSI Erweiterung bereitgestellt werden. Beispiele für Daten, die über diese Schnittstelle übertragen wurden, sind in Abbildung 195 dargestellt.
Die Erzeugung der Kameradaten kann dabei vom Anwender individuell beeinflusst werden. So können die Auflösung der zu generierenden Kamerabilder, Linsenverzerrungen und Bilddatenformate für die zu modellierende reale Kamera angepasst werden.

Die Arbeiten an der Integration neuer Versionen des Kamerasensormodells vom Projektpartner Bosch, welches diese Daten nutzt, wurden iterativ fortgeführt. Aufgrund der

zunehmenden Komplexität des Kamera-Use-Cases wurde dieser im späten Projektverlauf nicht weiter unter der Nutzung von CarMaker verfolgt, sondern auf den Lidar-Anwendungsfall fokussiert. Dennoch ist mit der prototypischen Umsetzung der Kamera-User Story inklusive OSI::SensorView für Kameradaten der Grundstein für eine industrielle Nutzung der Schnittstelle mit CarMaker gelegt.

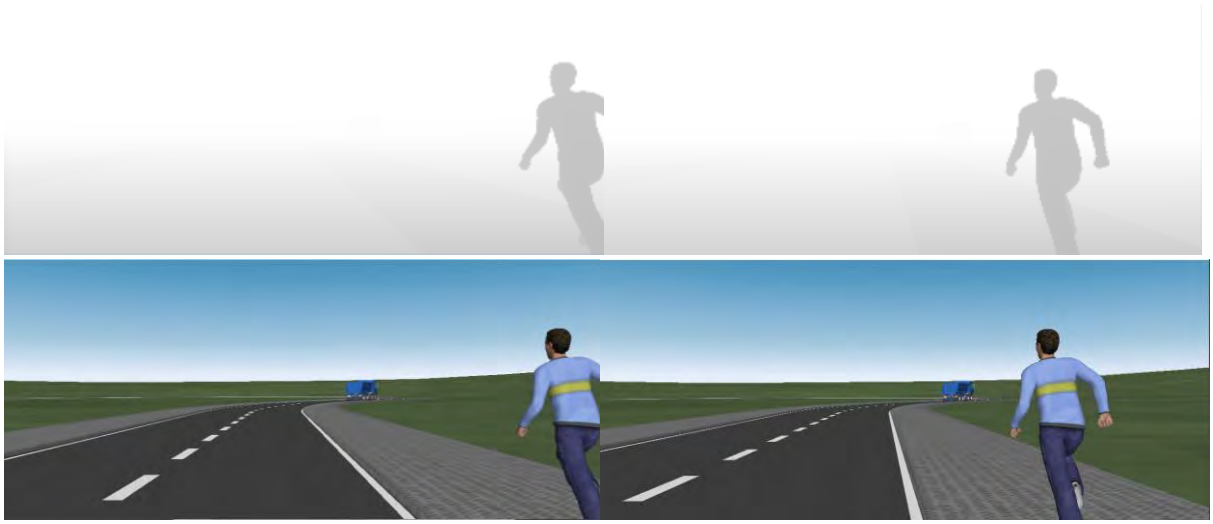


Abbildung 195: Ausgabe von Tiefeninformationen und von RGB-Bildern für eine Stereo-Kamera aus der CarMaker-Simulationsplattform

2.3.3.4.5 Allgemeines zur Umsetzung der LIDAR-Simulation

Der Fokus der Entwicklungen des LiDAR in MS1 bis MS 3 lag auf den folgenden Arbeitspaketen und die daraus resultierenden bis zum Projektende zu erreichenden Ziele waren:

- Genauere Nachbildung des Valeo SCALA1
- Steigerung der Leistung, Effizienz und Wiedergabetreue (LiDAR-Beam-Pattern) durch Überarbeitung der LidarSensorView & LidarSensorViewConfiguration
- Erste Umsetzung des sensor_view_config_request()-Mechanismus zur automatisierten Parametrierung des Renderings
- Dokumentation und Open Source Veröffentlichung des Modells

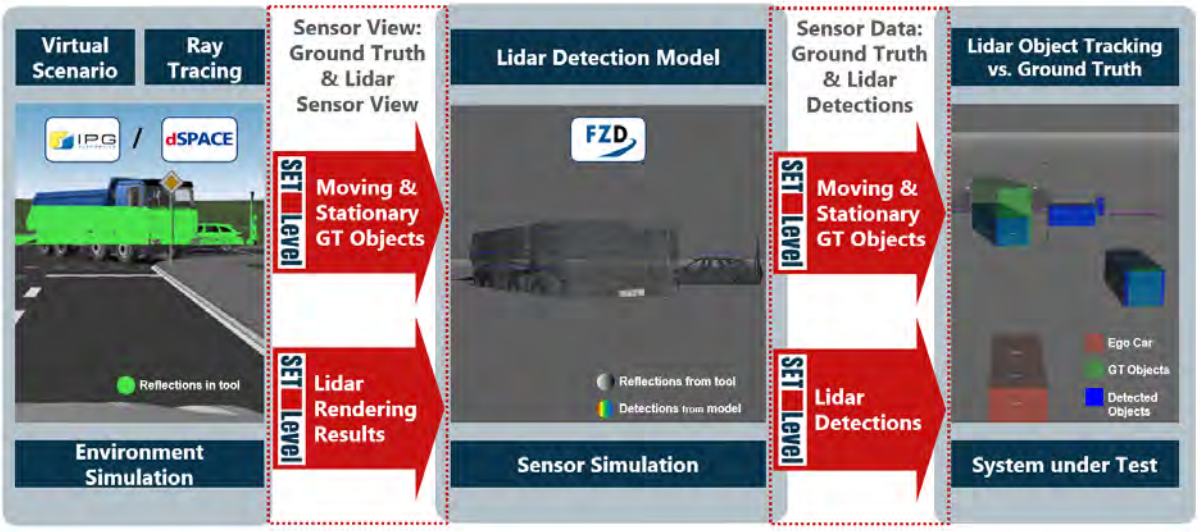


Abbildung 196: SUC 3 MS 3 LiDAR Sensor Simulation im Detail

In Abbildung 196 wird die LiDAR Sensor Simulation im Detail gezeigt. Die roten Pfeile stellen die im SET Level Projekt erarbeiteten Ergebnisse plakativ dar.

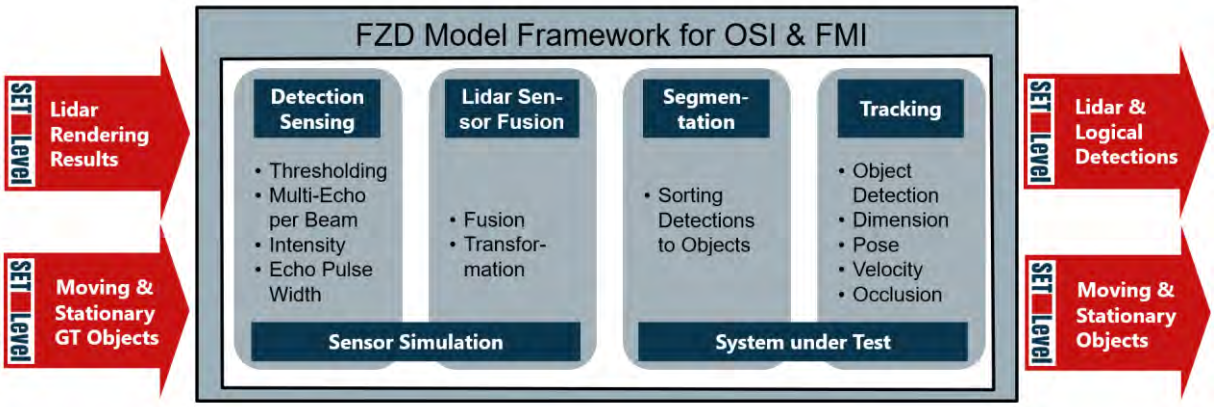


Abbildung 197: SUC 3 MS 3 LiDAR Sensor Simulation Framework

In Abbildung 197 ist der interne Aufbau des Model-Frameworks des FZI zu sehen. Die verwendeten Schnittstellen sind OSI und FMI.

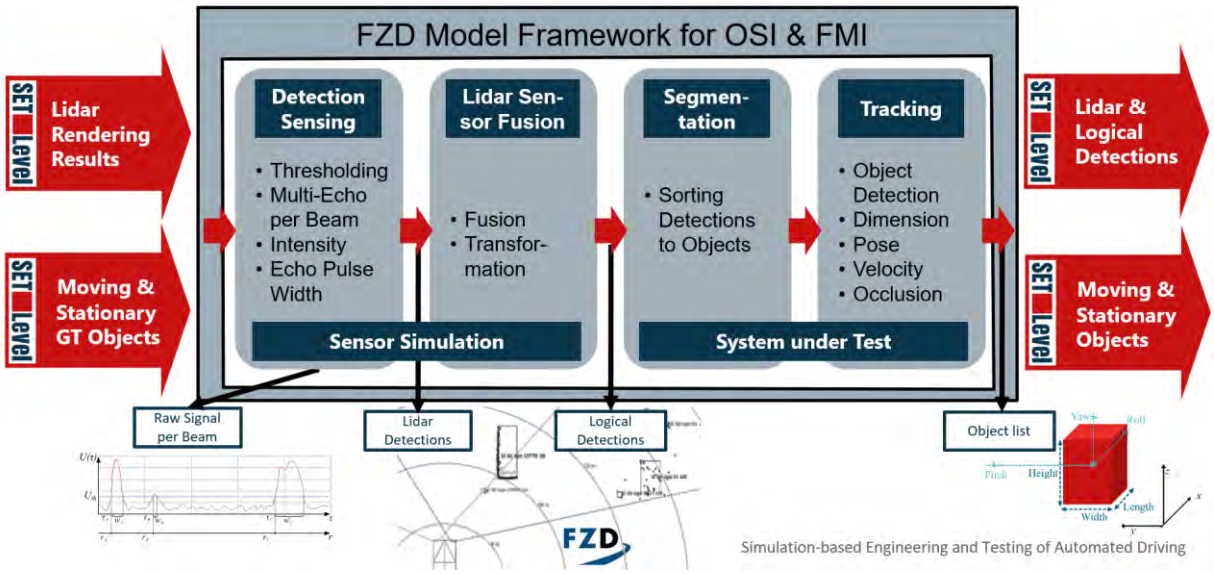


Abbildung 198: SUC 3 MS 3 LiDAR Sensor Simulation Framework mit Zwischenschritte

In Abbildung 198 werden im unteren Bereich die physikalischen Details des LiDAR Sensor Simulation Frameworks gezeigt.

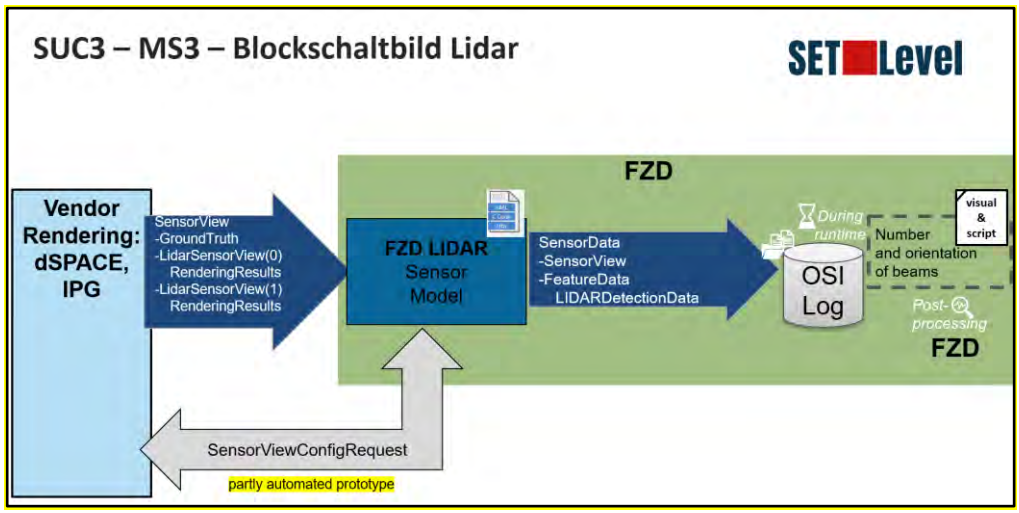


Abbildung 199: SUC 3 MS 3 Blockschaltbild LiDAR

In Abbildung 199 wird das vereinfachte Blockschaltbild der LiDAR Simulation für den Meilenstein MS 3 gezeigt. Es zeigt die OSI Nachrichten. In der Umsetzung konnte die Übertragung des SensorViewConfigRequest prototypisch gezeigt werden (siehe gelb hinterlegte Beschreibung)

Die Projekt-Ziele der drei Meilensteine MS 1 bis MS 3 konnten in der verlängerten Projektlaufzeit erreicht werden.

Trotzdem bleiben weitere Forschungsarbeiten offen. Diese noch geplanten Tätigkeiten für weitere laufende oder zukünftige Projekte umfassen:

- Stichproben-Validierung des Modells ist im Projekt VVMethoden in Arbeit
- Weitere Unterstützung/Weiterentwicklung im Projekt VVMethoden
- Einbringen nicht rückwärtskompatibler Änderungen an OSI in ASAM OSI 4.0

Entscheidende Leistungsverbesserungen konnten im Projekt SET Level erreicht werden. Anhand des für das Abschlussevent erzeugten Materials wird nachfolgend eine detailliertere Übersicht über die Umsetzung der Lidar-Simulation gegeben:

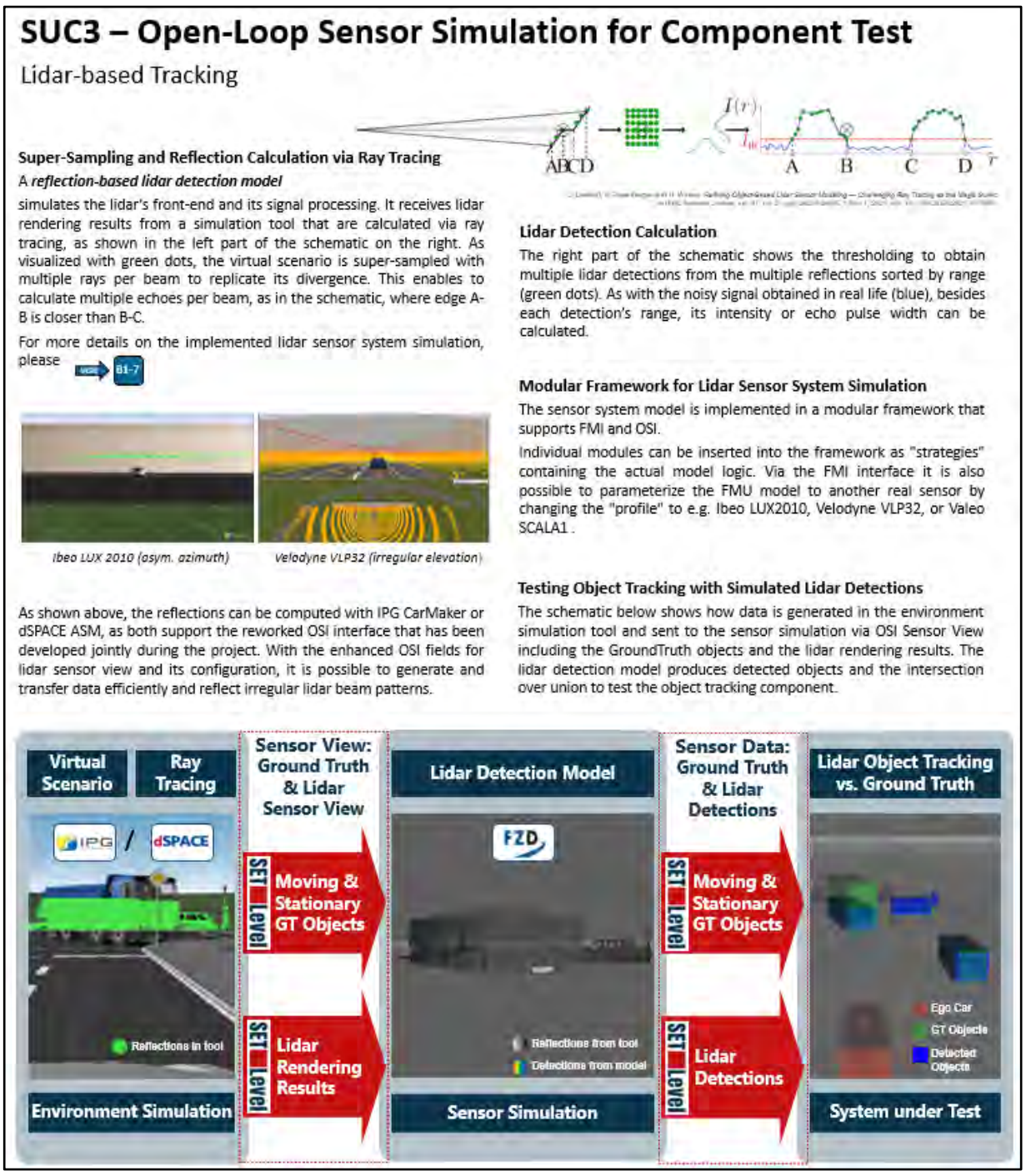


Abbildung 200: Modulare Werkzeugkette zur LIDAR Simulation innerhalb SUC 3 (Posterform)

Das obige Übersichtsbild (Abbildung 200) wurde, für die Präsentation der Ergebnisse auf dem Abschlussevent in Form eines Posters, erstellt.
Dieses Poster gliedert sich in vier Bereiche:

1) Super-Sampling und Reflexionsberechnung über Raytracing
2) Berechnung der Lidar-Erkennung
3) Modulares Framework für die Simulation von Lidar-Sensorsystemen
4) Testen der Objektverfolgung mit simulierten LIDAR-Erkennungen

Diese Bereiche werden im Folgenden kurz beschrieben.

Super-Sampling und Reflexionsberechnung über Raytracing
Ein reflexionsbasiertes LIDAR-Detektionsmodell simuliert das Frontend des LIDARs und dessen Signalverarbeitung. Es erhält Lidar-Rendering-Ergebnisse von einem Simulationstool, die über Raytracing berechnet werden, wie im linken Teil des Schaltplans rechts gezeigt. Wie mit grünen Punkten visualisiert, wird das virtuelle Szenario mit mehreren Strahlen pro Strahl abgetastet, um seine Divergenz zu replizieren. Dies ermöglicht die Berechnung mehrerer Echos pro Strahl, wie im Schaltplan, wo die Kante A-B näher ist als B-C.
Wie oben gezeigt, können die Reflexionen mit IPG CarMaker oder dSPACE ASM berechnet werden, da beide die im Projekt gemeinsam entwickelte überarbeitete OSI-Schnittstelle unterstützen. Mit den erweiterten OSI-Feldern für die LIDAR-Sensoransicht und deren Konfiguration ist es möglich, Daten effizient zu generieren und zu übertragen und unregelmäßige Lidar-Strahlmuster zu reflektieren.

Berechnung der Lidar-Erkennung
Der rechte Teil des Schaltplans zeigt den Schwellenwert, um mehrere LIDAR-Erkennungen aus den Mehrfachreflexionen zu erhalten, die nach Bereich sortiert sind (grüne Punkte). Wie bei dem verrauschten Signal aus realen Messungen (blau) kann neben dem Bereich jeder Detektion auch deren Intensität oder Echopulsbreite berechnet werden.

Modulares Framework für die Simulation von Lidar-Sensorsystemen
Das Sensorsystemmodell ist in einem modularen Framework implementiert, das FMI und OSI unterstützt.
Einzelne Module können als "Strategien" mit der eigentlichen Modelllogik in das Framework eingefügt werden. Über die FMI-Schnittstelle ist es auch möglich, das FMU-Modell auf einen anderen realen Sensor zu parametrieren, indem das "Profil" auf z. B. Ibeo LUX2010, Velodyne VLP32 oder Valeo SCALA1 geändert wird.

Testen der Objektverfolgung mit simulierten LIDAR-Erkennungen
Das Schema im unteren Teil von Abbildung 200 zeigt, wie Daten im Umgebungssimulationstool generiert und über OSI Sensor View an die Sensorsimulation gesendet werden, einschließlich der GroundTruth-Objekte und der LIDAR-Rendering-Ergebnisse. Das LIDAR-Erkennungsmodell erzeugt erkannte Objekte und den Schnittpunkt über die Vereinigung, um die Objektverfolgungskomponente zu testen.

2.3.3.4.6 Umsetzung LIDAR-Simulation in Werkzeugen von dSPACE

Auch für die Kopplung dieser FMU konnte die bereits für Meilenstein 1 entwickelte Komponente (OSI-Adapter) genutzt und weiterentwickelt werden. Um eine Performance-Steigerung zu erzielen und die Wiedergabetreue zu verbessern, wurde eine Erweiterung und Abänderung des OSI-Standards vorgenommen. Für die Simulation des Lidarsensormodells mussten die Informationen aus dem tool-internen Raytracing in das OSI-Format überführt werden. Damit die Abtastung des dSPACE-Raytracers mit der vom Sensormodell erwarteten übereinstimmt, wurde bereits zum Meilenstein 2 ein gemeinsames Scan-Pattern festgelegt. Dieses bestimmt, in welchem Winkel und in welcher Auflösung Strahlen zur Abtastung in die Szene geschossen werden. Die sich daraus ergebenden Reflektionen werden dem Sensormodell dann in Form einer Punktewolke zur Verfügung gestellt. In Ergänzung zu dem vorhandenen Scan Pattern wurden weitere Profile für das Sensormodell erstellt, welche sich an den Scan Pattern bereits im Markt etablierter Lidarsensoren orientieren.

Auch an dieser Stelle wurde der OSI-Converter in die Simulationskette integriert. Da sich die Raytracing-Informationen und somit OSI-Nachrichten von Lidar und Kamera grundsätzlich unterscheiden, musste der Converter für den Lidar-Fall angepasst werden.

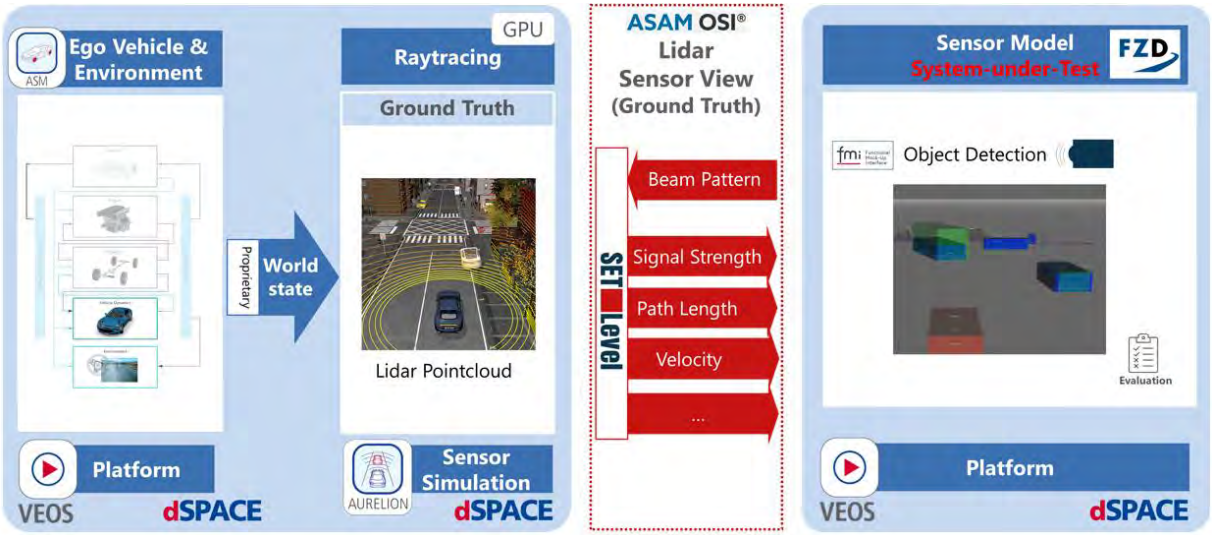


Abbildung 201: dSPACE Simulationsarchitektur für SUC 3 Lidarsensor

Die Architektur, siehe Abbildung 201, der Lidarsimulation entspricht vom Grundaufbau dem der Kamerasimulation. Allerdings besteht hier der Unterschied darin, dass das Beam Pattern des Sensors vor Simulationsbeginn an das Simulationstool übermittelt, werden muss.
Die erstmalige Umsetzung dieses im OSI-Standard vorgesehenen Konzepts ist im untenstehenden Abschnitt beschrieben.

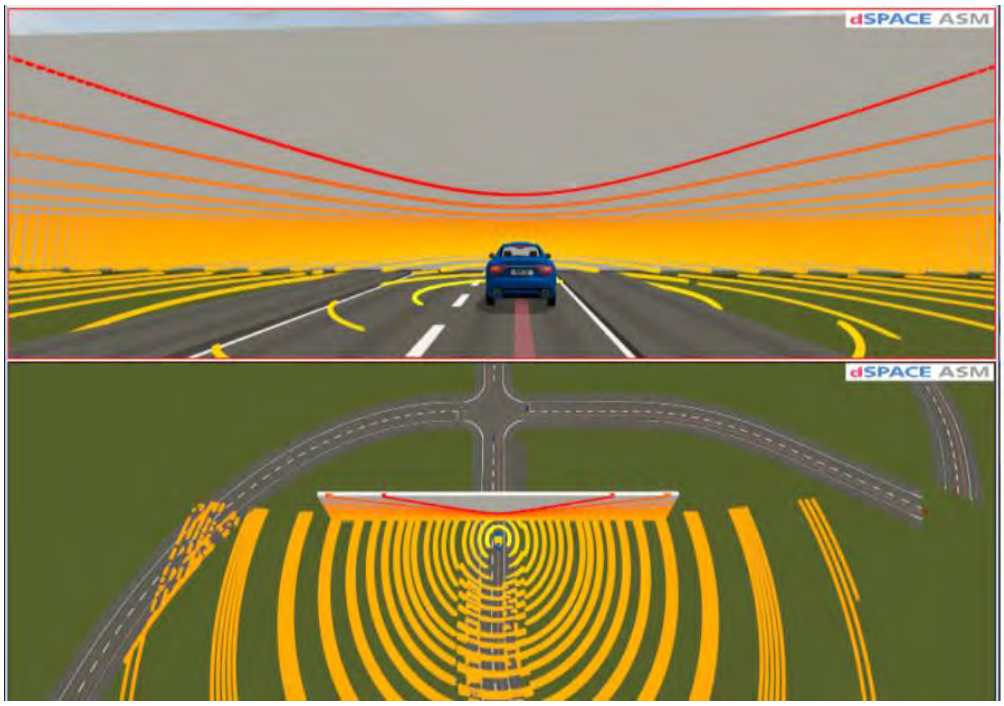


Abbildung 202: LIDAR Beam Pattern für Velodyne VLP32 (unregelmäßige Elevation) in dSPACE SensorSim

In enger Zusammenarbeit mit den Partnern des Forschungsprojekts VVMethoden wurde im Projekt SET Level die Vorarbeit zur späteren Sensormodellvalidierung eines realen Sensors

geleistet. Hierzu wurden Simulationen unter Integration eines Sensormodells vom Hersteller Valeo durchgeführt. Diese dienten dazu, eine möglichst realitätsnahe Kopie der sensorspezifischen Eigenschaften, die sich auf das Beam Pattern auswirken, zu ermöglichen. Dies diente als Grundlage für die spätere Nachstellung dieser Simulationen in der Realität.
Auch im Lidar Use Case konnte beim Umstieg von objektlistenbasierten auf rohdatenbasierte Daten bedingt durch die erheblich gestiegenen Datenmengen, abhängig vom Beam Pattern des Lidar, eine deutlich reduzierte Simulationsgeschwindigkeit beobachtet werden. Siehe Abbildung 202 für einen visuellen Eindruck des Beam Patterns.
Die Einflüsse auf die Simulationsgeschwindigkeit waren aufgrund der höheren Datenmenge beim Lidar (der Velodyne VLP32 ist z. B. ein Sensor mit 32 Kanälen und 360° Rundumsicht mit bis zu 1,2 Millionen Punkten pro Sekunde) noch drastischer als bei der Kamerasimulation. Aufgrund dieser Beobachtungen wurden Optimierungen im OSI Standard durchgeführt, die die Simulationsgeschwindigkeit signifikant erhöht haben. Ob hier eine weitere Optimierung der OSI 3.x Technologie (mit Protobuf) noch Potential zur Geschwindigkeitssteigerung besitzt, ist im Rahmen des Projekts SET Level von dSPACE nicht überprüft worden.

Konzept zur Konfiguration der Sensor Simulation mit OSI
Des Weiteren wurde als Proof of Concept eine prototypische Umsetzung des OSI-SensorView Configuration-Request-Konzepts zur automatisierten Parametrierung des Raytracings entwickelt. Hierbei gibt der Sensor vor Simulationsbeginn seine gewünschte Konfiguration in Form eines OSI-Signals an eine vorgelagerte Anwendung. Diese Informationen werden in einem zweiten Schritt dann verwendet, um das Simulationstool zu konfigurieren. Dieser teilautomatisierte Prozess soll Fehleranfälligkeiten bei der Toolkonfiguration reduzieren.
Zum Thema Parametrierung von Sensormodellen über OSI::SensorViewConfiguration wurde in Kooperation mit dem Projektpartner FZD eine erste prototypische Lösung (POC: Proof-of-Concept) erarbeitet, die nachfolgend kurz vorgestellt wird.
Grundsätzliche Idee von OSI::SensorViewConfiguration ist die automatische Konfiguration der Tools bzw. Modelle die die OSI::SensorView Nachricht für die Sensormodelle zur Verfügung stellen. D.h. bevor es zu einer Simulation kommt müssen Konfigurationen des Sensormodells an den Sender der OSI::SensorView Nachricht übertragen werden.
Ideal passiert dies automatisiert, dies ist aber im Moment in der dSPACE Toolkette nicht durch eine passende Toolschnittstelle freigelegt, so dass im POC eine teilautomatisierte Lösung umgesetzt wurde (Abbildung 203).

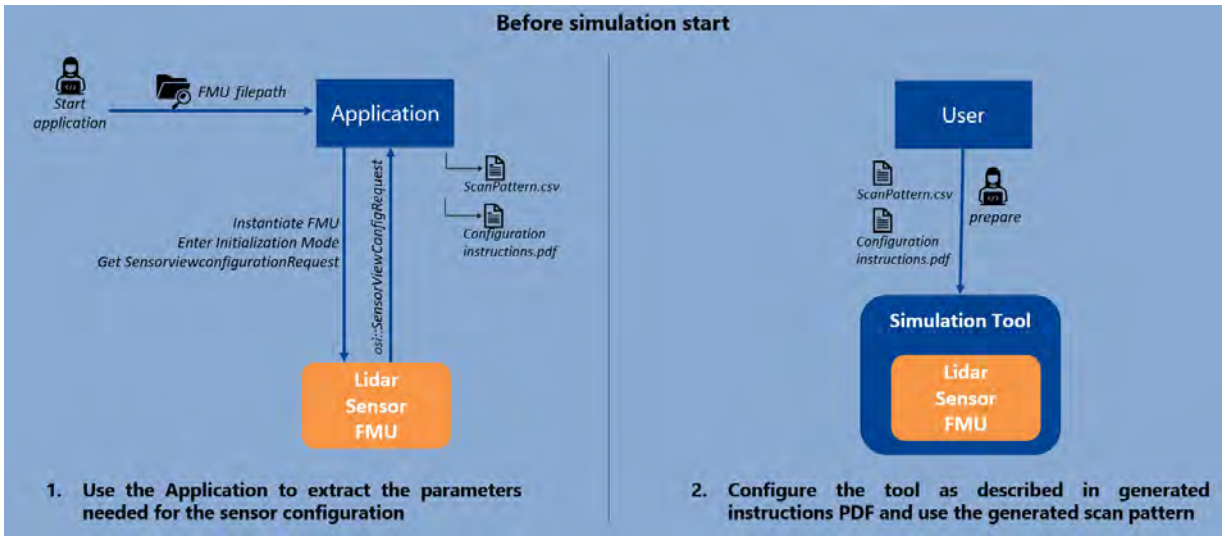


Abbildung 203: dSPACE-FZD POC (Proof of Concept) für LIDAR OSI::SensorViewConfiguration

Der Workflow ist in zwei Schritte aufgeteilt:
In Schritt 1 startet der Anwender das POC Tool unter Angabe des Pfades zur FMU des Sensormodells. Danach ermittelt das POC Tool über die OSI::SensorViewConfiguration Nachricht die im Sensormodell hinterlegten Konfigurationsparameter. Diese Information wird für den Anwender in Form einer Excel CSV Datei und einer PDF-Datei abgespeichert und angezeigt.
In Schritt 2 wertet der User die CSV-Datei und die PDF-Datei aus und konfiguriert das Simulationstool (als Sender von OSI::SensorView) entsprechend. Dieser Schritt sollte dann zukünftig automatisiert erfolgen, um mögliche Konfigurationsfehler zu vermeiden.
Nachfolgend zwei Beispiele in Abbildung 204 und Abbildung 205 für die konkreten Konfigurationsdaten die aus dem FZD LIDAR Modell ausgelesen wurden.

Name	Value
Sensor Id	0
Mounting position X	1,5205
Mounting position Y	0
Mounting position Z	1,232
Mounting position roll	0
Mounting position pitch	0
Mounting position yaw	0
Max interactions	1
Max path length	400
Emitter wave length	905

Abbildung 204: Beispiel für eine LIDAR SensorViewConfiguration

Wesentliche Konfigurationsdaten sind z. B. die „Mounting Position“ des Sensors am Fahrzeug und für LIDAR Sensoren das sog. Scan Pattern für das Raytracing im Simulator. Hierdurch wird z. B. gesteuert wie viele Strahlen betrachtet werden, wie sie ausgerichtet sind und welche Intensität sie besitzen.

1	Azimuth Angles	Elevation Angles	Launch Time	Relative Intensity
962	-12.513	-0.0296063	0	242.867
963	-12.513	-0.0291248	0	242.867
964	-12.513	-0.0286434	0	242.867
965	-12.513	-0.0281619	0	242.867
966	-12.513	-0.0276804	0	242.867
967	-12.513	-0.0271989	0	242.867
968	-12.513	-0.0267175	0	242.867
969	-12.513	-0.026236	0	242.867
970	-12.513	-0.0257545	0	242.867
971	-12.513	-0.0252731	0	242.867
972	-12.513	-0.0247916	0	242.867
973	-12.513	-0.0243101	0	242.867
974	-12.513	-0.0238287	0	242.867
975	-12.513	-0.0233472	0	242.867
976	-12.513	-0.0228657	0	242.867
977	-12.513	-0.0223842	0	242.867
978	-12.513	-0.0219028	0	242.867
979	-12.513	-0.0214213	0	242.867
980	-12.513	-0.0209398	0	242.867
981	-12.513	-0.0204584	0	242.867
982	-12.513	-0.0199769	0	242.867
983	-12.513	-0.0194954	0	242.867
984	-12.513	-0.019014	0	242.867
985	-12.513	-0.0185325	0	242.867
986	-12.513	-0.018051	0	242.867
987	-12.513	-0.0175695	0	242.867
988	-125.082	-0.0310503	0	242.957
989	-125.082	-0.0305689	0	242.957
990	-125.082	-0.0300874	0	242.957

Abbildung 205: Beispiel für ein LIDAR Scan Pattern in einer SensorViewConfiguration

2.3.3.4.7 Umsetzung LIDAR-Simulation in Werkzeugen von IPG

Für alle Sensortechnologien, die im Rahmen des SUC 3 betrachtet wurden, konnte von den Projektpartnern das Physical Sensor Models (PSM) Modul für IPG CarMaker genutzt werden (siehe **Error! Reference source not found.**). Dieses Modul ermöglicht die Erzeugung von Sensorrohdaten für die Technologien Radar, Kamera, Lidar und Ultraschall. Zur Nutzung mittels OSI::SensorView wurden die Daten entsprechend der Vorgaben im Standard serialisiert und den OSI-FMUs der Projektpartner zugänglich gemacht.

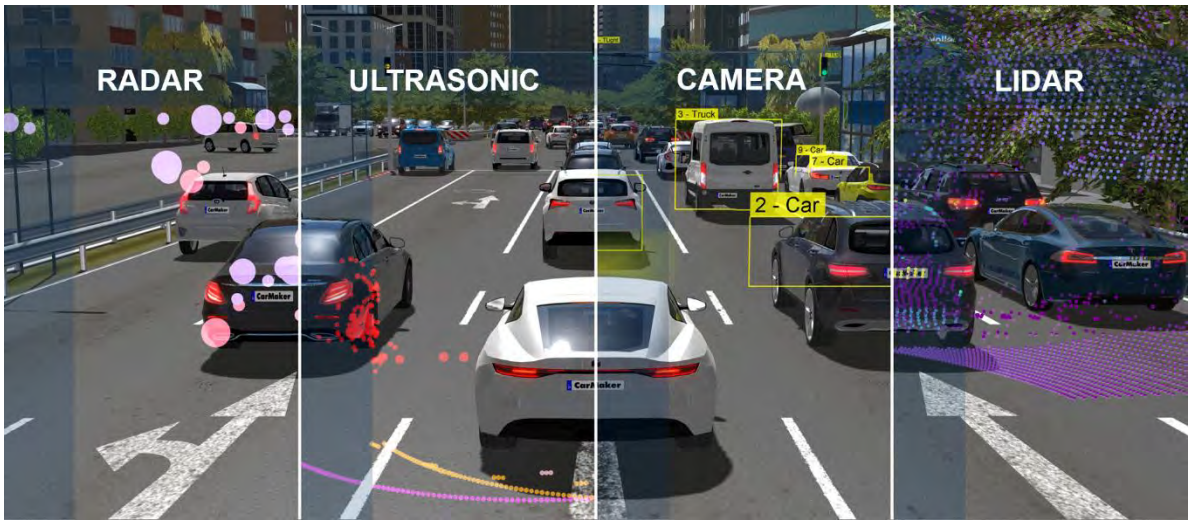


Abbildung 206: Erzeugung von Sensorrohdaten mit Physical Sensor Models (PSM) Modul für IPG CarMaker

Das Lidar RSI (Raw Signal Interface) als Teil des PSM-Moduls berechnet die Ausbreitung von Lidarsignalen in der 3D-Szene unter Berücksichtigung der Objektgeometrien, Materialeigenschaften und Umgebungsbedingungen. Mögliche Reflektionstypen bei der Interaktion eines Strahls mit anderen Objekten sind beispielsweise Retroreflektion, Transparenz (d.h. keine Reflektion) oder diffuse Reflektion. Während der Ausbreitung des Signals werden atmosphärische Effekte sowie Strahlaufweitung berücksichtigt.
Für die Umsetzung des Simulation Use Case 3 wurde auf die modulare Umgebung mit CarMaker gesetzt, die im Projekt SET Level erarbeitet wurde. Sie erlaubt es, beliebige externe Modelle einzubinden oder CarMaker-interne Modelle zu verwenden (siehe Abbildung 207).

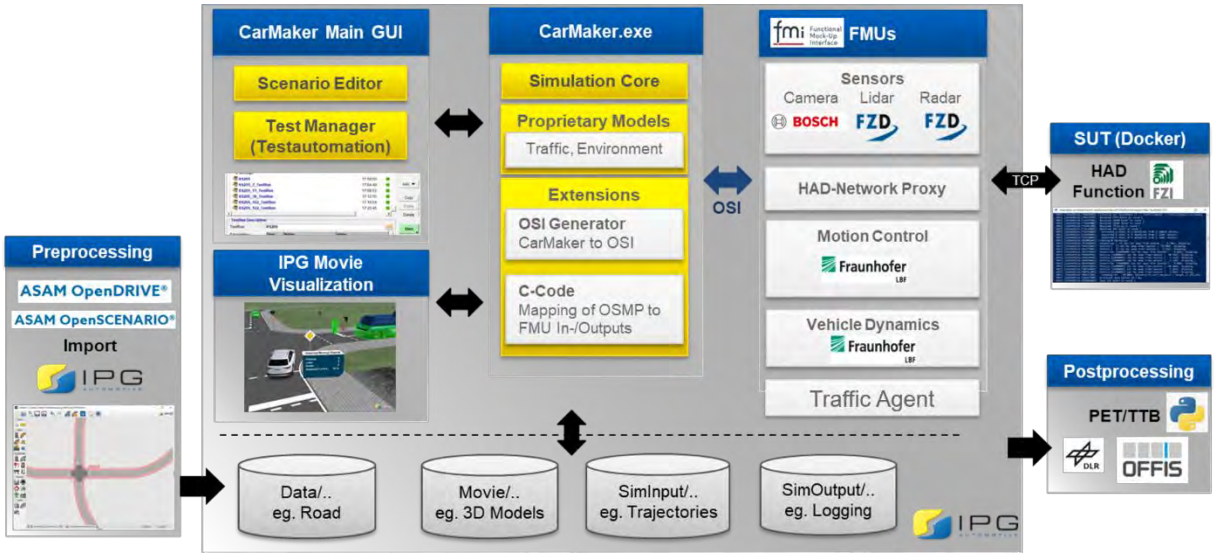


Abbildung 207: Modulare CarMaker-Simulationsarchitektur im Projekt SET Level

Im Laufe des Projektes erfolgte eine Umstrukturierung der OSI::SensorView-Nachricht für Lidarsensordaten. Durch die Arbeiten wurde es ermöglicht, OSI::SensorView für Lidar nicht nur für reguläre, sondern auch für asymmetrische, nutzerspezifische Beam Patterns zu verwenden. Das ermöglicht eine effizientere Modellierung von Sensoren, die bisher OSI-basiert nur durch rechenintensive Überabtastung (siehe Abbildung 208) realisiert werden konnte. Durch eine direkte Zuordnung der aus dem Raytracing resultierenden Reflexionen zu den gesendeten Strahlen konnte die Menge der zu übertragenden Daten über die OSI Schnittstelle

signifikant reduziert werden. In der Umsetzung für den Simulation Use Case 3 konnte damit eine deutlich gesteigerte Simulationsperformance gegenüber früheren Versionen erreicht werden.

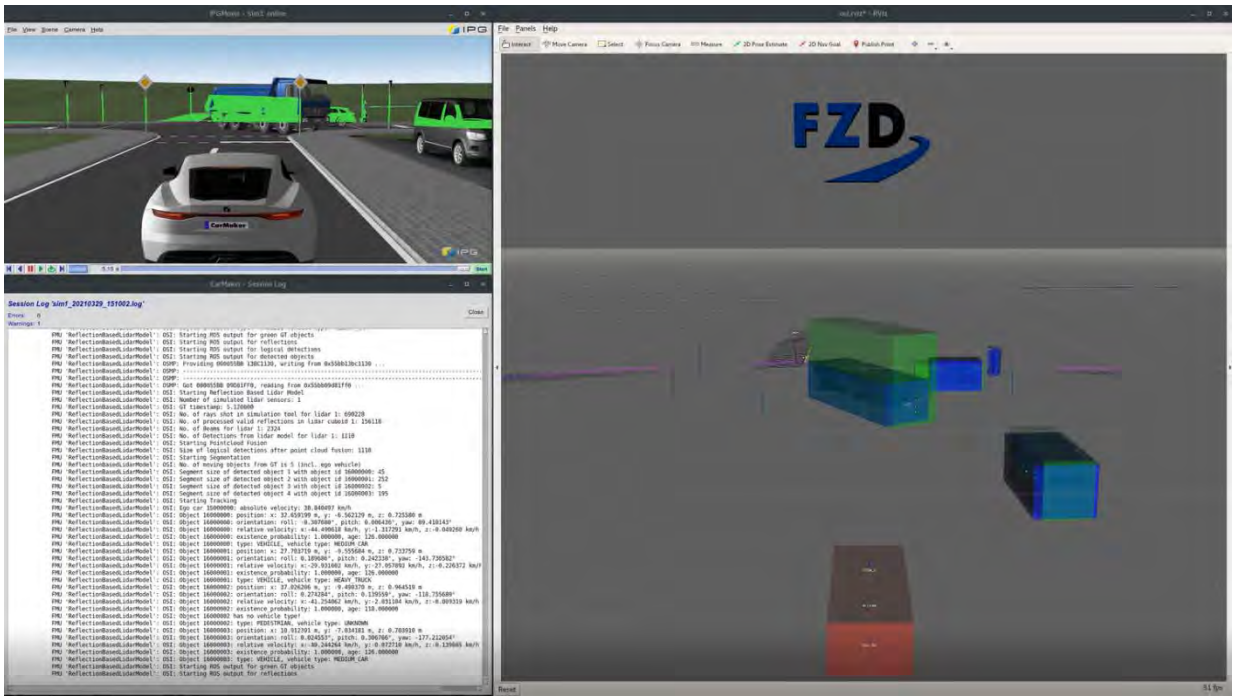


Abbildung 208: Generierung der Lidar-Sensordaten mit Überabtastung (oben links) und Detektion von Objekten mittels FZD-Lidarmodell (rechts).

Die Verwendung des integrierten Modells wurde bei der Abschlussveranstaltung demonstriert und ist in Abschnitt 2.3.3.4.5 detailliert erläutert.

2.3.3.4.8 Umsetzung Radar-Simulation in Werkzeugen von ETAS und OpenSource

Der Fokus der Entwicklungen des Radars in MS 1 bis MS 3 lag auf den folgenden Arbeitspaketen:

- der Implementierung
- der Integration und
- der Inbetriebnahme

des Raytracing-basierten Radarmodells des Projektpartners FZI in der Werkzeugkette.
In diesem Zuge wurde die konsistente Nutzung und Synchronisation von 3D-Modellen und die Annotation von Materialien in Umgebungssimulation und (externen) physikalischen Sensormodellen evaluiert.
Hierfür wurden im von BMW betreuten Open-Source Projekt OpenMATERIAL Spezifikationen für 3D-Modelle und die Annotation von Materialparametern erarbeitet und bereitgestellt.
Ebenso wurde eine Spezifikation für die Vereinheitlichung der Struktur von 3D-Fahrzeugmodellen bereitgestellt. Im Laufe des Projekts wurden die Spezifikationen auf Basis der gewonnenen Erkenntnisse überarbeitet und nach und nach in die angestrebte Standardisierung überführt.
Die Motivation des SUC 3 ist unter anderem der folgende Use Case:
Bei der Zusammenarbeit von Entwicklern der Firma A mit Entwicklern der Firma B in verschiedenen Frameworks gibt es folgende Anforderungen:

- Beitrag unterschiedlicher Expertisen und Fokus auf unterschiedliche Analyseziele, d.h. Betrachtung von Sensoreffekten auf unterschiedlichen Abstraktionsniveaus
- Austauschbarkeit von Sensorsystemmodellen durch definierte Schnittstellen mit definierten Verarbeitungsebenen
- Standardisierte Materialparameter und 3D-Modelle, damit eine Vergleichbarkeit zwischen Simulationsmodellen gewährleistet und die Simulation reproduzierbar ist

Um dieses Ziel zu erreichen, werden die Schnittstelle OSI und das noch zu standardisierende OpenMATERIAL eingesetzt.
Die modulare Radarsimulationskette kann die gesamte Datenverarbeitungskette typischer Radarsensoren abbilden. Eine typische Radarverarbeitung besteht aus den folgenden 8 Funktionsblöcken:

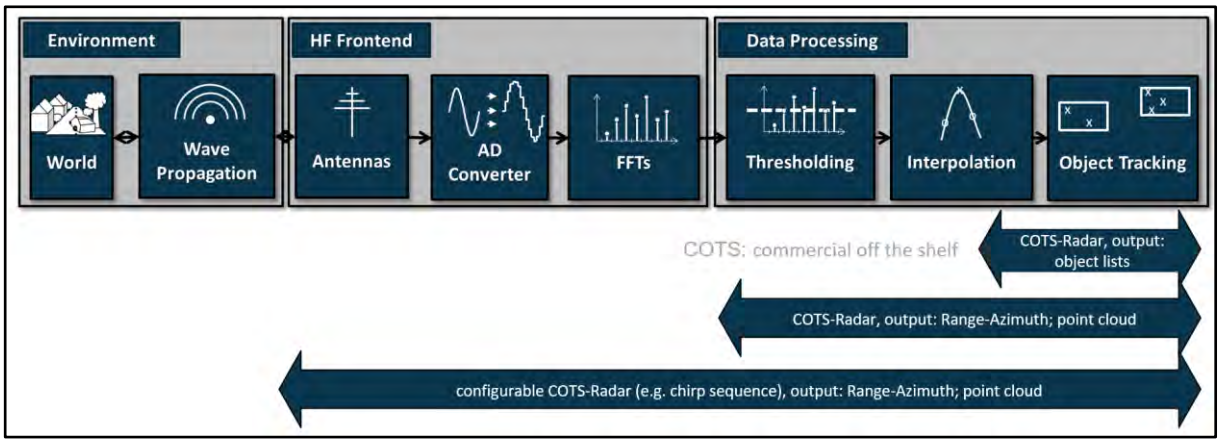


Abbildung 209: SUC 3 Meilenstein MS 3 – 8 Funktionsblöcke der Radar Simulation

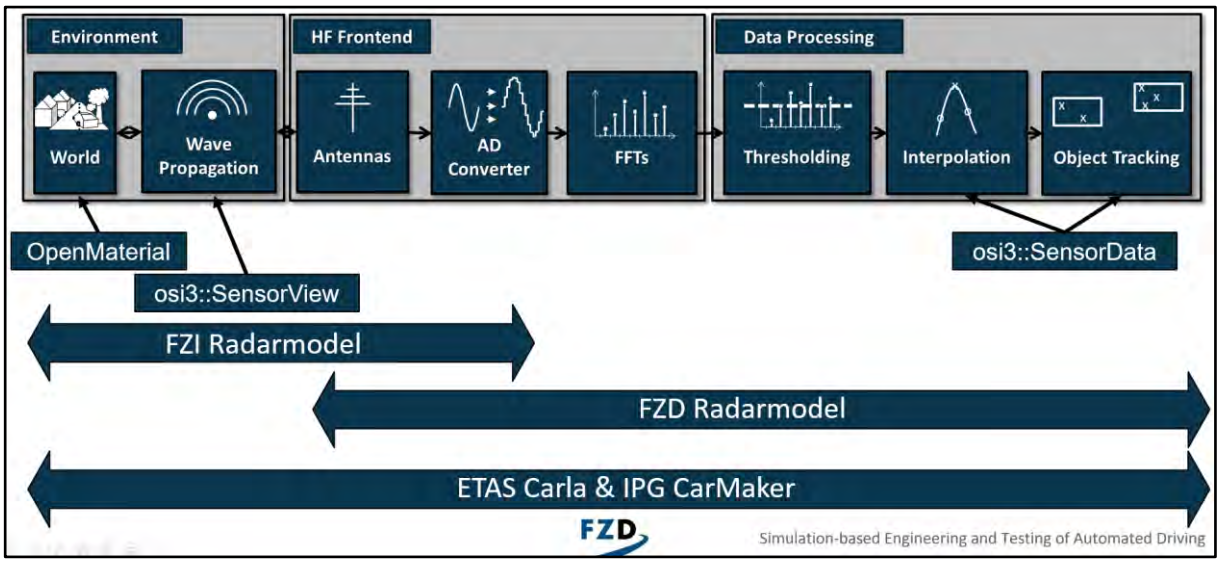


Abbildung 210: SUC 3 Meilenstein MS 3 – 8 Funktionsblöcke der Radar Simulation mit der Spannweite der Toolketten

Die Abbildung 210 und die Abbildung 211 stellen diese modulare Werkzeugkette vor und zeigen das Mapping auf die typische Radarverarbeitungskette.
Auf der linken Seite ist der Umgebungssimulator aufgeführt, der das Szenario modelliert (Objektpositionierung, Objektgeometrie usw.) und Grundlage für das Sensormodell ist. Im modularen Aufbau können sowohl in Simulationsumgebungen integrierte Reflection-Tracing-

Ansätze zur Erfassung der Szene genutzt werden als auch ein im Projekt entwickeltes externes Tracing.
Zusätzlich können die Objektinformationen mittels OpenMATERIAL angereichert werden, um relevante Materialinformationen den Sensormodellen bereitzustellen. Die berechneten Reflektionen werden anschließend in einem für unterschiedliche Radarsensoren parametrisierbaren Sensormodell verarbeitet und im sensorspezifischen Feature-Data abgebildet. Diese Daten stehen dann der klassischen Radarsignalauswertung zur Verfügung.
Im Folgenden finden sich die detaillierten Dokumentationen der an AP 4.4 beteiligten Partner bzw. der Integrierten Simulationsmodule. Dies erfolgt anhand des für das Abschlussevent erzeugten Materials:

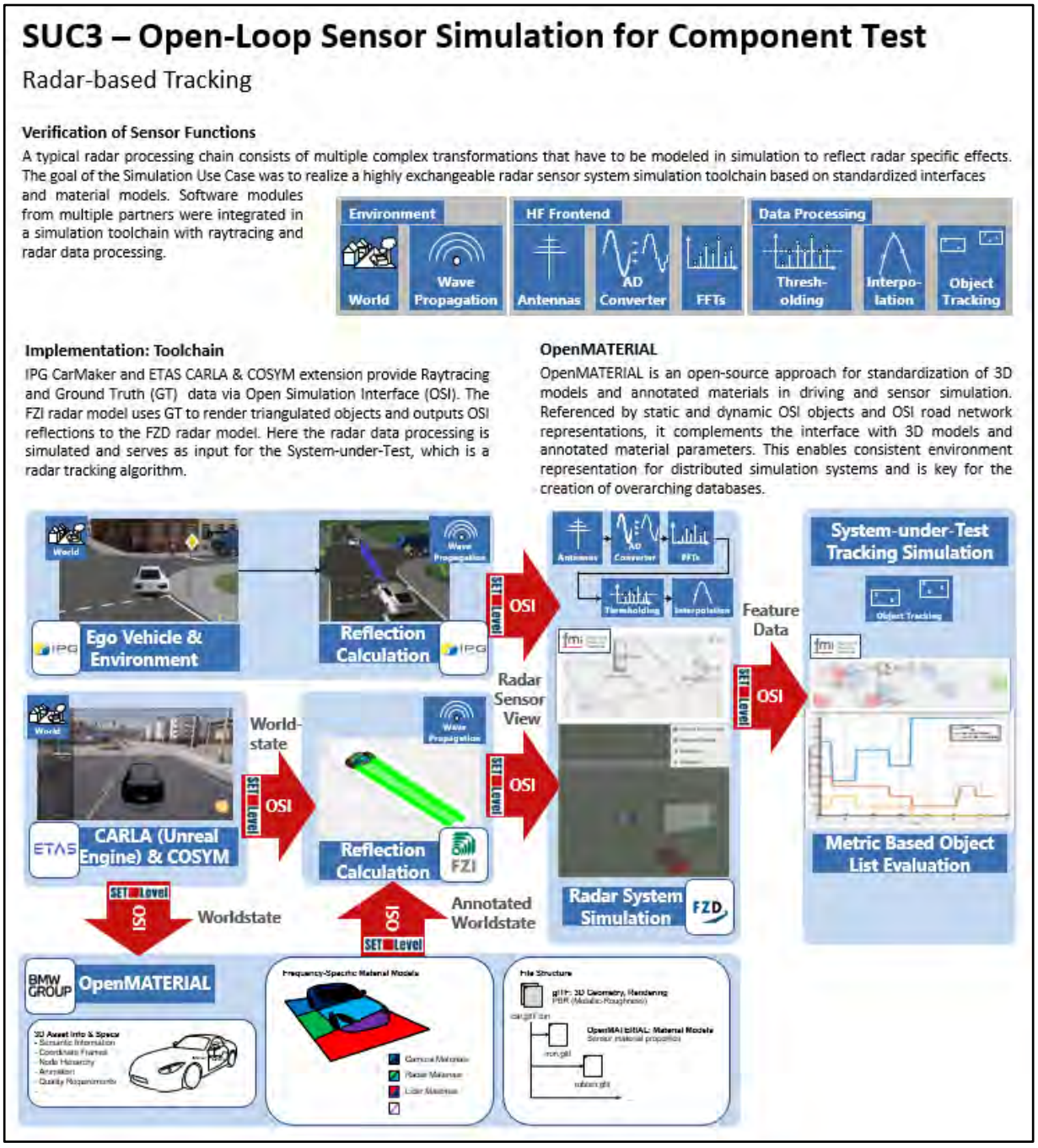


Abbildung 211: Modulare Werkzeugkette zur Radar Simulation innerhalb SUC 3

Das obige Übersichtsbild (Abbildung 211) wurde, für die Präsentation der Ergebnisse auf dem Abschlussevent in Form eines Posters, erstellt.
Dieses Übersichtsbild zur Radar-Simulation gliedert sich in drei Themen:

1) Verifikation von Sensorfunktionen
2) Implementierung der Werkzeugkette (Toolchain)
3) OpenMATERIAL

Diese werden im Folgenden kurz beschrieben und im Anschluss wird auf weitere Details eingegangen.

Verifikation von Sensorfunktionen
Eine typische Radarverarbeitungskette besteht aus mehreren komplexen Transformationen, die in der Simulation modelliert werden müssen, um radarspezifische Effekte widerzuspiegeln. Ziel des Simulation Use Case war es, eine hochgradig austauschbare Werkzeugkette zur Simulation von Radarsensorsystemen auf Basis standardisierter Schnittstellen und Materialmodelle zu realisieren. Softwaremodule mehrerer Partner wurden in eine Simulations-Toolchain mit Raytracing und Radardatenverarbeitung integriert.

Implementierung der Werkzeugkette (Toolchain)
IPG CarMaker und ETAS CARLA & COSYM stellen Raytracing- und Ground Truth (GT)-Daten über Open Simulation Interface (OSI) bereit. Das FZI-Radarmodell verwendet GT zum Rendern triangulierter Objekte und gibt OSI-Reflexionen an das FZD-Radarmodell aus. Hier wird die Radardatenverarbeitung simuliert und dient als Eingabe für das System under Test, einen Radar-Tracking-Algorithmus.

OpenMATERIAL
OpenMATERIAL ist ein Open-Source-Ansatz zur Standardisierung von 3D-Modellen und annotierten Materialien in der Fahr- und Sensorsimulation. Es wird von statischen und dynamischen OSI-Objekten und OSI-Straßennetzdarstellungen referenziert und ergänzt die Schnittstelle mit 3D-Modellen und annotierten Materialparametern. Dies ermöglicht eine konsistente Umfeld-Darstellung für verteilte Simulationssysteme und ist der Schlüssel für die Erstellung übergreifender Datenbanken.

Der simulationsbasierte Test der Radarobjektverfolgung und die entsprechende modulare Werkzeugkette wird jetzt im Detail vorgestellt.

Das SuT („System under Test“) ist in diesem Anwendungsfall eine radarbasierte Tracking-Simulation, begleitet von einer OSPA-Metrik (optimal sub-pattern assignment metric), die die Positionen verfolgter Objekte in Bezug auf die GT-Objektliste vergleicht. Es wurde eine modulare, standardisierte Radarverfolgungskette implementiert, die typische Effekte der Radarverarbeitung auf phänomenologischer Ebene simuliert.

Einführung und Radar-Werkzeugkette
Bevor Implementierungsdetails beschrieben werden, wird ein Blick auf die Struktur einer allgemeinen Radarverarbeitungskette geworfen.
Das zugrunde liegende Problem kann in Umgebungssimulation, Modellierung des Hochfrequenz-Front-Ends und Datenverarbeitungskomponenten unterteilt werden.
In der Umgebungssimulation wird die Wechselwirkung der gesendeten elektromagnetischen Radarwellen mit der Umgebung abgebildet. Die Wellen, die von der Umgebung

zurückreflektiert werden, werden vom HF-Frontend über die Antenne des Radars empfangen. Diese Elemente werden dann in ein elektrisches Signal umgewandelt und digitalisiert.
Die von einem FFT-Algorithmus (Fast Fourier Transformation) durchgeführte Frequenzanalyse dient als Eingabe für die Datenverarbeitung zur Generierung von Detektionen und Objekten.
Die Datenverarbeitung erfolgt mittels dynamischer Schwellenwerte, Peak-Interpolation und anschließender Objektverfolgung.
Um das Potenzial einer modularen Simulations-Werkzeugkette zu demonstrieren, wurde eine Kombination verschiedener Simulationsmodule betrachtet, wie z. B. Umgebungs- und Sensorsimulation, und in verschiedenen Simulationen kombiniert, um verschiedene Anwendungsfälle zu simulieren (objektbasierte/reflexionsbasierte Modelle).

Umgebungssimulation
Zunächst soll mit einer allgemeinen Beschreibung der Umweltsimulation und Modularisierung begonnen werden.
In der Fahrsimulation bedeutet Umweltsimulation die Schaffung virtueller Welten auf Basis von Karten, Szenarien und 3D-Modellen.
Während Karten den statischen Inhalt der virtuellen Welt in Bezug auf das Straßennetz und seine Infrastruktur beschreiben, werden dynamische Inhalte (z. B. Verkehrsteilnehmer und deren Interaktion) durch Szenarien beschrieben. 3D-Modelle bereichern Karten und Szenarien durch dreidimensionale Darstellungen der virtuellen Welt für Rendering und Sensorsimulation. Da Umgebungssimulationen typischerweise alle relevanten Informationen besitzen, um alle Objekte der virtuellen Welt zu beschreiben, werden sie in einem verteilten Simulationsframework als zentrale Quelle für die Verteilung globaler Objektlisten verwendet.
Gängige Nachrichtendefinitionen für diese Art von Informationen werden vom Open Simulation Interface (OSI) spezifiziert:

Open Simulation Interface
Für die Radar-Werkzeugkette wurden drei OSI-Schnittstellen implementiert:

- SensorView
- GroundTruth und
- FeatureData

1. **SensorView**-Nachrichten liefern die Eingangsdaten für Sensormodelle. Entsprechende Nachrichten werden aus **GroundTruth**-Nachrichten abgeleitet, die eine ideale Beschreibung der virtuellen Welt liefern und alle simulierten Objekte im globalen Koordinatenrahmen spezifizieren. SensorView-Nachrichten enthalten GroundTruth-Informationen als Referenz.
2. **FeatureData**-Nachrichten enthalten erkannte Features im Sensorbezugssystem. Sie dienen als Eingabe für Sensormodelle, die die Objekterkennung simulieren, oder Feature-Fusionsmodelle.

Durch die Verwendung von OSI-Schnittstellen konnten die Komponenten der Werkzeugkette miteinander verbunden und austauschbar gemacht werden.
Trotzdem spezifiziert OSI keine Modellintegration. Hier setzt das Functional Mockup Interface (FMI) an.

Functional Mockup Interface

FMI ermöglicht die standardisierte Integration von Simulationsmodellen (Definition von Containern, Schnittstellen und entsprechenden Binärdateien/Code).
In der Werkzeugkette ermöglichte es eine standardisierte Integration der Sensormodelle, wie die FZD-Radarsimulation und das SuT (Tracking Simulation).
Neben der Schnittstellendefinition und der Modellintegration war ein weiteres Problem zu lösen: Die Reflexionsberechnung erfordert 3D-Geometrie und Materialinformationen. Daher musste OSI um Spezifikationen aus dem Open-Source-Projekt OpenMATERIAL erweitert werden.

OpenMATERIAL
Ziel von OpenMATERIAL ist die Entwicklung eines generischen, standardisierten 3D-Modellaustauschformats und einer Modellstruktur mit annotierten, physikalisch basierten Materialien für Rendering und Sensorsimulation.
Zu diesem Zweck enthält das Projekt Vorschläge für Erweiterungen des Khronos Group glTF 2.0 Dateiformats.
3D-Modelle und entsprechende Materialien, die von OpenMATERIAL spezifiziert wurden, können aus OSI-Nachrichten referenziert werden.
Jedes Objekt in OSI-Objektlisten enthält eine so genannte Model Reference, die einen Dateipfad angibt, der auf ein entsprechendes 3D-Modell verweist.
Diese Informationen können von jeder Komponente eines verteilten Simulationsframeworks ausgewertet werden, um 3D-Modelle und Materialien aus einer (gemeinsamen) Dateifreigabe zu laden und eine eigene konsistente Kopie der virtuellen Welt zu erstellen.
Zusätzlich zu den 3D-Modellspezifikationen für Rendering und Sensorsimulation bietet OpenMATERIAL anwendungsspezifische Materialparameter.
Auf diese Weise liefert ein einziges 3D-Modell Materialparameter für konventionelles Rendering, Kamera-, Radar-, Lidar- und Ultraschallsimulation.

Integration der Sensormodelle
Basierend auf der bereits erwähnten Implementierung standardisierter Schnittstellen konnten sowohl das industrieerprobte IPG CarMaker als auch die Open-Source-Umgebungssimulation Carla evaluiert werden. Während CarMaker die Reflexionsberechnung selbst bereitstellt, wurde Carla um diese Funktionalität durch ein entsprechendes FZI-Modell (angebunden mit Hilfe der OSI GroundTruth Schnittstelle) erweitert. Für beide Lösungen wurden OSI SensorView Ausgabeschnittstellen implementiert.
Dies ermöglichte die Integration der FZD-Radarsystemsimulation als Sensormodell. Schließlich ermöglichte die Implementierung von OSI FeatureData als Ausgabe dieser Komponente die Kopplung des SuT (Tracking Simulation) und die entsprechende Metrik-basierte Auswertung. Im folgenden soll mit einer kurzen Beschreibung der Reflexionsberechnung begonnen werden.

Reflexionsberechnung in IPG Carmaker
Innerhalb der CarMaker-Reflexionsberechnung werden Strahlen in die Szene (= virtuelle Umgebung) geschossen und auf Interaktion mit dieser überprüft. Trifft hier ein Strahl auf ein Objekt, wird ein direkt reflektierter Strahl berechnet und ein weiterer reflektierter Strahl in der Szene weiterverfolgt. Die Reflexion wird durch die Fresnel-Gleichung unter Berücksichtigung der Permittivität des getroffenen Materials modelliert.
Neben Verlusten durch die Reflexionen werden auch Ausbreitungsverluste berücksichtigt. Hier wird die Dämpfung der elektrischen Feldamplitude nach dem Signal-Rausch-Verhältnis bestimmt und ist abhängig von Frequenz, Regenrate und Sichtweite (z. B. beeinflusst durch

Nebel). Zusätzlich wird die Dämpfung durch Freiraumausbreitung berücksichtigt. Schließlich, um Streueffekte von rauen Oberflächen zu berücksichtigen, wird eine materialabhängige stochastische Komponente auf der Normalen der Fläche verwendet.

Berechnung der Reflexionen im FZI-Modell
Der modulare Charakter der Simulations-Werkzeugkette ermöglicht den Austausch von Modellen gemäß den Simulationszielen.
Es wurde eine zweite Reflexionsberechnung des FZI integriert, das sogenannte dreiecksbasierte Radarreflexionsmodell, das eine Objektliste als Eingabe verwendet.
Zusätzlich zur Objektliste ist die Spezifikation von triangulierten 3D-Modellen pro Objekt erforderlich.
Basierend auf diesen Informationen berechnet das Modell die Reflexionen aus der Abstraktion der empfangenen elektromagnetischen Wellen.
Dreiecke, die zur Rückstreuung des Radarsensors beitragen, werden durch einen Strahlenschnittpunkt mit der Radarsensorebene erkannt.
Ein physikalisch-optischer Ansatz wird verwendet, um die Rückstreuung der Dreiecke zu berechnen. Vorherige Reflexionen des Strahlenpfads werden basierend auf der Weglänge und den Informationen des getroffenen Dreiecks berechnet.
Materialeigenschaften für jedes Dreieck werden aus triangulierten 3D-Modellen abgeleitet, die in das Modell eingegeben werden.
Das Modell wertet „model reference“-Informationen aus der OSI GroundTruth aus, um Referenzen zu 3D-Modellen zu erhalten.
Die Modellausgabe sind Reflexionen, die von OSI RadarSensorView spezifiziert werden.
Alle Details zum den internen Aufbau des FZI-Dreieck-basierten Radarreflexionsmodells wurden während des Abschluss-Events (ASE) am Stand der Radar-Simulation präsentiert.

Radarmodell FZD
Als Empfänger der Ausgangsdaten der zuvor beschriebenen Reflexionsberechnung verwendet das FZD-Radarmodell (verpackt als FMU) OSI-Reflexionen als Eingang. Mit diesen Daten startet das HF-Frontend (High Frequency) und die Datenverarbeitungssimulation.
Für die Simulation werden Fensterfunktionen und der FFT-Algorithmus für jede Reflexion aus der Reflexionsberechnung berücksichtigt. Dies ermöglicht die Berücksichtigung von Effekten wie Interferenzen im Modell. Als Ergebnis der Simulation steht der sogenannte Radarwürfel zur Verfügung, der die Grundlage für weitere Verarbeitungsschritte bildet. Für die Berechnung der Detektionen wird ein dynamischer Schwellwert über den Radarwürfel in allen Raumrichtungen (Reichweite, Dopplergeschwindigkeit und Azimut) durchgeführt. Eine Sub-Bin Interpolation führt zum endgültigen Ausgangssignal.
Die Umwandlung von Sensorkoordinaten in Fahrzeugkoordinaten führt zu OSI Logical Detections, die als OSI FeatureData übertragen werden (siehe oberer Teil Abbildung 211).
Neben den Detektionen sind zum besseren Verständnis die OSI GroundTruth Bounding Boxen der Objekte in der Simulation dargestellt. Das untere Bild ist ein Screenshot der ROS-Ausgabe des Radarmodells, die zur gleichen Zeit wie das Reflexionsberechnungsbild für IPG CarMaker aufgenommen wurde.
Hier sind GT Bounding-Boxen grau dargestellt, die getrackten Objekte. Weiterhin sind die Detektionen an den Rändern der Bounding-Boxen dargestellt. Details des Modells, wie es funktioniert und eine aufgezeichnete Demo konnten während des Abschluss-Events (ASE) am Stand der Radar-Simulation genauer angesehen werden.

SuT-Tracking Simulation und Metrik-Basierte Evaluierung

Als zu testendes System wurde in diesem SUC eine Radartrackingsimulation verwendet. LogicalDetections, die in entsprechende Objekte umgewandelt wurden, dienten als Eingabe für diesen Teil der Werkzeugkette. Die Auswertung der verfolgten Objekte in Bezug auf die GT-Objektliste erfolgte mit Hilfe der OSPA-Metrik („optimal sub-pattern assignment metric“). Diese Metrik ordnet getrackte Objekte GT-Objekten zu und bestimmt den Abstand zwischen den Bounding-Box-Zentren.
Die obere Abbildung (Radar-Übersichts-Poster) visualisiert, wie die Metrik mathematisch funktioniert, wobei die blauen Bounding-Boxen GT Objekte und die roten Bounding-Boxen getrackte Objekte visualisieren.
Das grüne Objekt auf der rechten Seite ist das Ego-Fahrzeug.
Das Optimum dieser Metrik ist 0 und die schlechteste Situation wird mit 1 wiedergegeben.
Das Diagramm darunter zeigt das Zeitdiagramm der Metrik im SUC 3-Szenario.
Hier zeigt die blaue Linie den Wert der OSPA-Metrik, die rote Linie visualisiert die Anzahl der GT-Objekte, die sich im Sichtfeld des Sensors befinden, und die Anzahl der erkannten Objekte wird in Gelb dargestellt.

Zusammenfassung der Arbeiten bezüglich des Radar Use Cases

Zusammenfassend wurde eine modulare, standardisierte Radar-Tracking Werkzeugkette implementiert, die typische Effekte der Radarverarbeitung auf phänomenologischer Ebene simuliert.
Die Werkzeugkette bietet dem Anwender die Möglichkeit, einerseits die Reflexionsberechnung direkt vom Toolanbieter zu verwenden (im Falle von IPG Automotive) und andererseits ein benutzerdefiniertes Reflexionsberechnungsmodell in Verbindung mit einer Open-Source-Umgebungssimulator zu verwenden.
Dies ermöglicht es dem Modell- oder Algorithmen-Entwickler, die spezifische Modellierungsebene für seine Simulationsaufgabe zu wählen: objektbasiert oder reflexionsbasiert.
Die Open-Source-Umgebungssimulation Carla bietet Flexibilität für die Integration neuer Ansätze: Hier konnten von OpenMATERIAL spezifizierte 3D-Modelle integrieret werden
Es wurden alle OSI-Schnittstellen für die Sensormodellintegration genutzt (OSI::GroundTruth, OSI::SensorView und OSI::SensorData) und entsprechende Systemgrenzen evaluiert. Das Urteil in dieser Hinsicht fiel durchweg positiv aus.
Dieser modulare Ansatz ermöglichte:

1. Integration von objektbasierten und reflexionsbasierten Sensormodellen in die gleiche Simulations-Werkzeugkette (Modellqualität kann basierend auf Simulationsaufgabe gewählt werden)
2. Austauschbarkeit der Umgebungssimulation (ermöglicht die Konsistenz zwischen Simulationsinstanzen, z. B. SiL<->HiL)
3. Skalierbarkeit (Sensormodelle und Rechenlast können verteilt werden, mehrere Modelle können parallel ausgeführt werden)
4. Verantwortlichkeiten sind durch spezifizierte Schnittstellen klar getrennt, was für die Validierung wichtig ist
5. OpenMATERIAL ermöglicht die Modellierung konsistenter virtueller Umgebungen in der Umgebungssimulation und in Sensormodellen. Es wurde hier ein erster POC erstellt, um das Potenzial dieses Ansatzes zu demonstrieren. In weiteren öffentlich geförderten Projekten (z. B. Vivaldi, KI Data Tooling) wird OpenMATERIAL weiterentwickelt, um den Aufbau gemeinsamer Datenbanken für 3D-Modelle und Materialien zu ermöglichen

FZD

Die reflexions-basierten Modelle wurden in SUC 3, dem Komponententest, angewendet. Das System under Test war hier eine Trackingfunktion. Am Modelleingang wurden CarMaker oder ASM angekoppelt. Am Modellausgang wurden CSV-Dateien von den getrackten Objekten und von der Ground Truth ausgegeben und in einer Matlab Pipeline evaluiert. Dadurch wurde auch die Kopplung mit nachgeschalteten Analysetools nachgewiesen.

FZI

Die ursprünglich geplante Integration der MATLAB/Simulink-basierten Implementierung des modellinternen Raytracers konnte aufgrund werkzeugseitiger Einschränkungen bzw. der Verfügbarkeit von Toolboxen nicht realisiert werden. Zur Kompensation wurde das komplette Modell auf eine C++-Implementierung umgezogen. Dies ermöglicht die Verwendung der im Projekt abgestimmten Schnittstellen FMI/OSI.
Im Projekt SET Level wurde die Portierung des Modells abgeschlossen. Basierend auf dieser portierten Version des radar-spezifischen Raytracers (triangle_based_radar_reflection_model) wurden die projektweit abgestimmten Schnittstellen und Kapselung implementiert. Als Input verwendet das Modell eine OSI SensorView Nachricht. Im speziellen das Feld global_ground_truth mit den Informationen model_reference, base.position und base.orientation. Für die model_reference wurden nur die Formate STL und GLTF getestet, es können aber eine Vielzahl weiterer Formate gelesen werden. GLTF stellt die Basis für die prototypische Anbindung von OpenMATERIAL dar.
Zusätzlich nutzt das Modell die radar_sensor_view.view_configuration zur Paramterisierung des Modells, insbesondere die Parameter view_configuration.field_of_view_horizontal, field_of_view_vertical, number_of_rays_horizontal, number_of_rays_vertical, max_number_of_interactions und emitter_frequency. Die Anbindung des Modells an einen Umgebungssimulator der Partner wurde aufgrund von technischen Einschränkungen bei der Bereitstellung von 3D-Modellen nicht umgesetzt. Konzeptionell lässt sich die Brücke aber schließen.
Als Rückfalllösung wurde eine interne Simulationsumgebung basierend auf der Unreal Engine verwendet.
Als Ausgabeformat erweiterte das Modell die radar_sensor_view um die berechneten Reflektionen (radar_sensor_view.reflection). Pro Reflektion werden die Felder source_horizontal_angle, source_vertical_angle, time_of_flight und signal_strength befüllt. Das Modell nutzt nur statische Szenen, weswegen der doppler_shift aktuell nur die Informationen der groundtruth nutzt. D.h. sensorspezifische Effekte bzgl. der Dopplergeschwindigkeit sind aktuell nicht berücksichtigt. Siehe hierzu auch die Abbildung 212 mit dem radarspezifischen Raytracing-Modell.

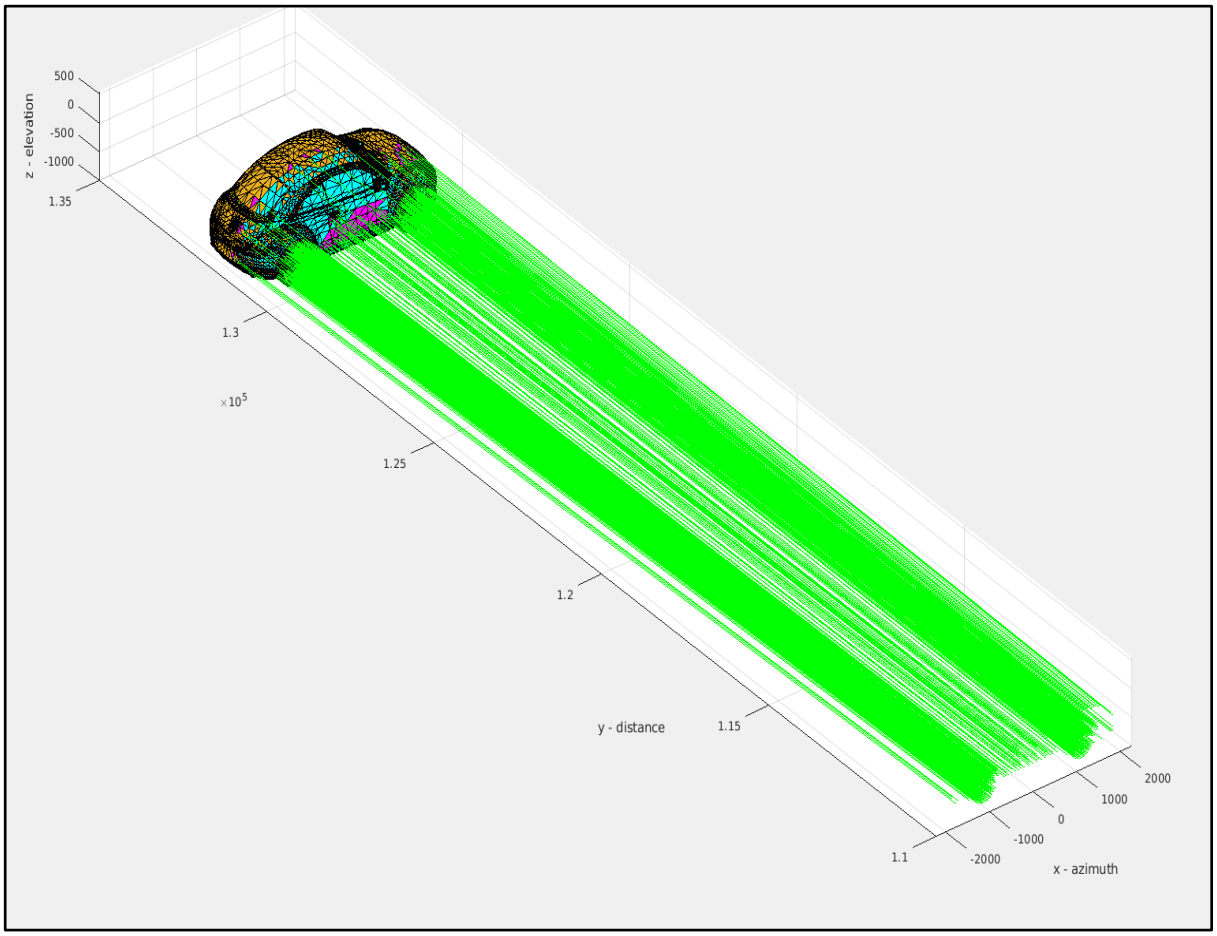


Abbildung 212: Radarspezifisches Raytracing-Modell

Die Integrationskette ist in Abbildung 213 dargestellt.

Die prinzipielle Funktionsweise des zu implementierenden Ansatzes kann somit exemplarisch unter Verwendung einzelner 3D-Objekte gezeigt werden. Die Echtzeitfähigkeit des Raytracers ist nicht gegeben. Dieser Wunsch wurde im Rahmen der Validierung von einem Partner gestellt, wurde aber aufgrund der Komplexität der Berechnung nicht angestrebt und somit nicht umgesetzt. Vielmehr handelt es sich um einen Demonstrator, welcher die Nutzung eines hybriden Ansatzes, d.h. die Kombination von physikalischer und geometrischer Optik, untersucht. Insbesondere zur Berechnung der sensorspezifischen signal_strength verspricht der Ansatz Vorteile.
Die Lösung wurde im Rahmen des Abschlussevents (ASE) vorgestellt (vergleiche auch das Sensormodell welches im TP 3 detailliert beschrieben wird).

BMW
Im Rahmen von AP 4.4 wurde die Umsetzung des Simulation Use Case 3 (SUC 3) vorangetrieben. In der dazugehörigen Regelrunde wurden die Demonstrationsziele bzgl. der im Rahmen von SUC 3 geplanten Open-Loop Simulationen geplant und umgesetzt. Hierbei lag der Fokus in der Anbindung externer Sensormodelle an Umgebungssimulationen mit Hilfe standardisierter Schnittstellen z. B. ASAM Open Simulation Interface (OSI).
Während der gesamten Projektlaufzeit wurde das BMW Open-Source Projekt OpenMATERIAL weiterentwickelt. Hierbei wurde insbesondere die Spezifikation zur Materialbeschreibung an die Anforderungen der Radar User Story angepasst, welche innerhalb von SUC 3 definiert ist.

Im Rahmen der Radar User-Story sollte ein externes, Raytracing-basiertes Radarmodell des Projektpartners FZI an die Simulationsumgebungen angebunden werden. Hierdurch sollte die konsistente Nutzung und Synchronisation von 3D Modellen und die Annotation von Materialien in Umgebungssimulationen und (externen) physikalischen Sensormodellen evaluiert werden. Zu diesem Zwecke wurden im vom BMW betreuten Open-Source Projekt OpenMATERIAL (https://github.com/LudwigFriedmann/OpenMaterial/) Spezifikationen für 3D Modelle und die Annotation von Materialparametern erarbeitet und bereitgestellt. Ebenso wurde eine Spezifikation für die Vereinheitlichung der Struktur von 3D Fahrzeugmodellen bereitgestellt. Im Laufe des Projekts sollten die Spezifikationen auf Basis der gewonnenen Erkenntnisse überarbeitet und mittelfristig in die Standardisierung überführt werden.
In der zweiten Projekthälfte wurde gemeinsam mit den Projektpartnern FZI und FZD ein Demonstrator spezifiziert und entwickelt, der die Anbindung des FZD Radarmodells und des Ray-Tracing-basierten FZI Radarmodells an die kommerzielle Simulationsumgebung IPG CarMaker und die Open-Source Simulationsumgebung Carla (https://carla.org/) umfasst. Der Demonstrator sollte den Aufbau modularer Simulationsarchitekturen, die Anbindung externer Sensormodelle und die Synchronisation von Umgebungsrepräsentationen in verteilten (Ko-)Simulationen zeigen (siehe Abbildung 213).

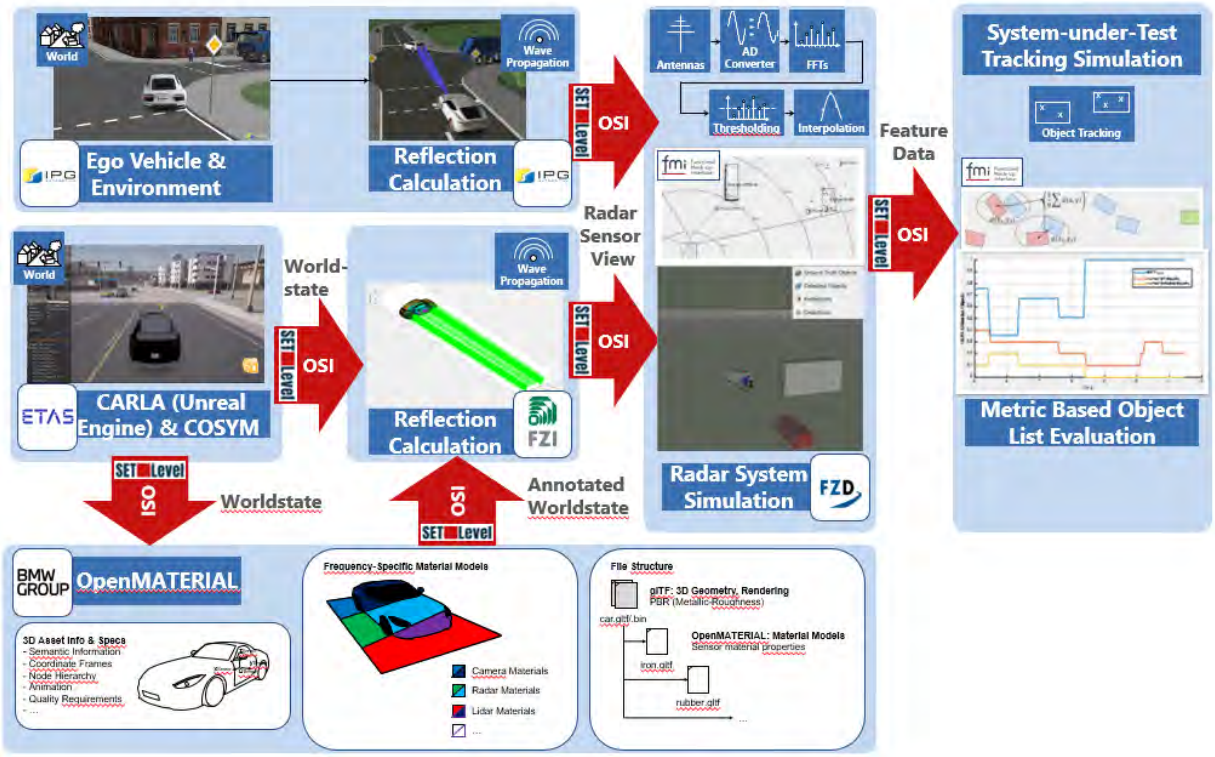


Abbildung 213: Radar-based Tracking im SUC 3

Für die Anbindung der verschiedenen Software-Komponenten wurde auf ASAM OSI Schnittstellen zurückgegriffen. Für die Radarsimulation erforderliche Erweiterungen wurden in den OSI-Schnittstellenbeschreibungen ergänzt und in die Standardentwicklung zurückgespiegelt. Physikalische Sensormodelle wie das Ray-Tracing-basierte FZI Radarmodell erfordern über die von den OSI Schnittstellen bereitgestellte Umgebungsmodellierung in Form von Objektlisten hinaus auch 3D Geometrien und entsprechende Materialrepräsentationen, um die Ausbreitung von Strahlen oder Wellen korrekt modellieren zu können. Daher wurden die OSI Schnittstellen für den Demonstrator um Referenzen zu anhand von OpenMATERIAL Spezifikationen aufgebauten 3D Modellen und Materialien ergänzt. Im Open-Source Repository von

OpenMATERIAL wurden neue Entwicklungszweige für die Abbildung von Radar-relevanten Materialparametern in den OpenMATERIAL Spezifikationen angelegt, namentlich

- Elektrische Leitfähigkeit
- Relative Permittivität
- Relative Permeabilität

Die konkrete Spezifikation der Parameter sowie der zu dokumentierenden Messparameter (z. B. Einfallswinkel, Oberflächenrauigkeit, Luftfeuchtigkeit) wurden über SET Level hinaus mit Teilnehmern des Forschungsprojekts VIVALDI abgestimmt, so dass auch hier Durchgängigkeit bzgl. 3D Modellen und Materialien entstehen kann.
Auf Basis einer Literaturrecherche wurden entsprechende Werte für den Frequenzbereich 77 GHz und die in der nachzustellenden Kreuzungssituation relevanten Materialien

- Eisen
- Gummi
- Glas
- Plexiglas
- Polypropylen
- Beton
- Vegetation
- Gipskarton

gesammelt. Es wurden außerdem 3D Modelle anhand der OpenMATERIAL Spezifikationen aufgebaut, z. B. ein generisches PKW-Modell (siehe Abbildung 214).

Abbildung 214: Generisches OpenMATERIAL PKW-Modell

Neben dem PKW-Modell wurde auch ein 3D Modell eines generischen LKW und ein 3D Modell der im Projekt verwendeten Straßenkreuzung erstellt. Das Modell der Kreuzung wurde projektintern abgelegt, die Fahrzeugmodelle wurden im Open-Source Repository von OpenMATERIAL veröffentlicht (https://github.com/LudwigFriedmann/OpenMaterial/tree/master/model_structure/objects).
Der beschriebene Demonstrator wurde im Rahmen des Abschlussevents des Projekts der Öffentlichkeit vorgestellt.

ETAS

Aus technischen und wirtschaftlichen Gründen wurde die geplante Simulationsumgebung „aiSim (von aiMotive) + COSYM (ETAS Tool)“ nicht weiterentwickelt.
Die Verwendung von CARLA (Open Source) und die Co-Simulations-Umgebung COSYM gewährt eine Unabhängigkeit der verwendeten Modelle von der Umgebungssimulation. Auf dem Papier und in der Theorie klingt das vorteilhaft und vielversprechend. Man vermeidet so den

sogenannten „vendor lock-in“, bei dem der Kunde (OEM, Tier1) nachdem er sich für eine Simulationsumgebung entschieden hat von diesem Werkzeug-Hersteller abhängig ist.
Allerdings hat sich im Laufe des SET Level Projekts immer mehr herausgestellt, dass diese große Unabhängigkeit dazu führt, dass alle Modelle als FMUs vorliegen und dann von der Co-Simulations-Plattform ausgeführt werden. Leider bedeutet jede Änderung der FMU ein komplettes Generieren der FMU und ein neues Aufsetzen der Simulation. Das erfordert selbst bei einem erfahrenden Entwickler einen zeitlichen Aufwand im Bereich einer Stunde.
Eine Gegenmaßnahme ist es, die FMUs konfigurierbar zu machen. Das führt aber dazu, dass jede FMU andere Konfigurationsmechanismen hat. So zum Beispiel ein anderes GUI oder eine andere Konfigurations-Datei. Auch die Simulationsumgebung CARLA ist in diesem maximal offenen Konzept als FMU an COSYM angeschlossen.
Als reine Ausführungsumgebung ist dieses offene Konzept oft vorteilhaft. Allerdings liegt ein Großteil der Aufgaben im Bereich der Entwicklung und des Fine-Tunings. Hierbei müssen sehr schnell Parameter geändert werden. Nach Möglichkeit sogar während der Laufzeit einer Simulation.
Die intrinsischen Nachteile dieses maximal offenen Ansatzes haben die Entwicklungsgeschwindigkeit von ETAS im SET Level Projekt stark reduziert.
Der zu Beginn von 2021 gefasste Entschluss die Simulationsumgebung aiSim für die Implementierung von SUC 2 und SUC 3 zu verwenden, erwies sich leider nach circa 6 bis 9 Monaten als nicht gangbar.
Der Aufwand für den Wechsel zu aiSim im SET Level Projekt war wegen Inkompatibilitäten und technischer Probleme zu hoch, um diesen Ansatz weiterzuverfolgen. Im Laufe der 3-monatigen Umsetzung wuchsen auch die strategischen und wirtschaftlichen Herausforderungen, welche letztendlich den Erfolg verhinderten. Die Lernkurve für die ETAS Mitarbeiter war sehr steil und die abrupte Entscheidung die Integration von aiSim zu beenden wird weiterhin als richtig eingestuft. Deshalb setzt ETAS nun ausschließlich auf CARLA als Simulations-Umgebung.

Grundlagen der ETAS-Umsetzung
Die Co-Simulationssoftware COSYM wird für die Implementierung von SUC 3 Aufgaben verwendet. COSYM importiert Modelle, die mit Entwicklungsumgebungen wie Simulink, C-Kompilatoren, FMU-Generatoren erzeugt werden. COSYM fungiert als Co-Simulationsmaster, der die Modelle mit Timing-Instanziierung und Kopplung unterstützt:

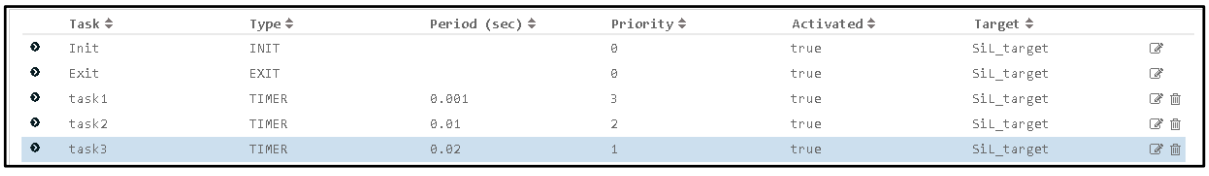


Abbildung 215: COSYM als Master zur Simulation von Modellen in verschiedenen Aufgabenstellungen

Die SET Level-Modelle (meist FMU-Einheiten), die aus verschiedenen Quellen stammen, werden in COSYM importiert. Die Toolkette zusammen mit CARLA, einem Open-Source-Umgebungssimulator, erzeugt die Visualisierung während der Durchführung der Co-Simulation. Die Nachrichtentransaktion basiert auf dem ASAM-OSI-Standard. Derzeit werden die FMUs mit dem OSMP-Paket gepackt, das die OSI-Nachrichtenarrays in Form von codierten Skalaren sendet. Diese Skalare sind Byteinformationen für den Speicherort der Nachrichten. Daher müssen die FMUs, die auf denselben Speicherplatz zugreifen, im selben Prozess ausgeführt werden, was durch die Projektkonfiguration in COSYM sichergestellt wird.

OpenDRIVE CARLA Implementierung

Die Straßennetzbeschreibung wird im OpenDRIVE-Format ausgeführt, das direkt in die CARLA FMU eingebettet ist. Diese Funktionsbeschreibungen des Straßennetzes sind stark von der CARLA-Python-Bibliothek abhängig, die derzeit die grundlegenden Informationen wie Straße, Gehwege und Ampeln unterstützt. Straße und Szenario, das Teil des 6-Ebenen-Modells von PEGASUS ist, werden in der Simulationsimplementierung mit einem Umgebungs- und Wettermodell ausgeführt.

Abbildung 216: Einfache Kreuzung, die aus einer OpenDRIVE-Datei in CARLA importiert wurde

OpenSCENARIO CARLA Implementierung

CARLA unterstützt auch das OpenSCENARIO Format für die Definition der Umgebung. Die OpenSCENARIO Umgebungs-Beschreibungen werden über XML-basierte ".xosc"-Dateien in CARLA importiert (siehe Abbildung 217). Die Funktion wird vom Scenario Runner-Projekt bereitgestellt. Dieses wird mit den Python-basierten Bibliotheken in der CARLA-Python-API aufgerufen.

Abbildung 217: Einfaches Szenario, das aus der OpenSCENARIO-Datei in CARLA importiert wurde

Realisierung von Radar User Stories in der ETAS Toolkette

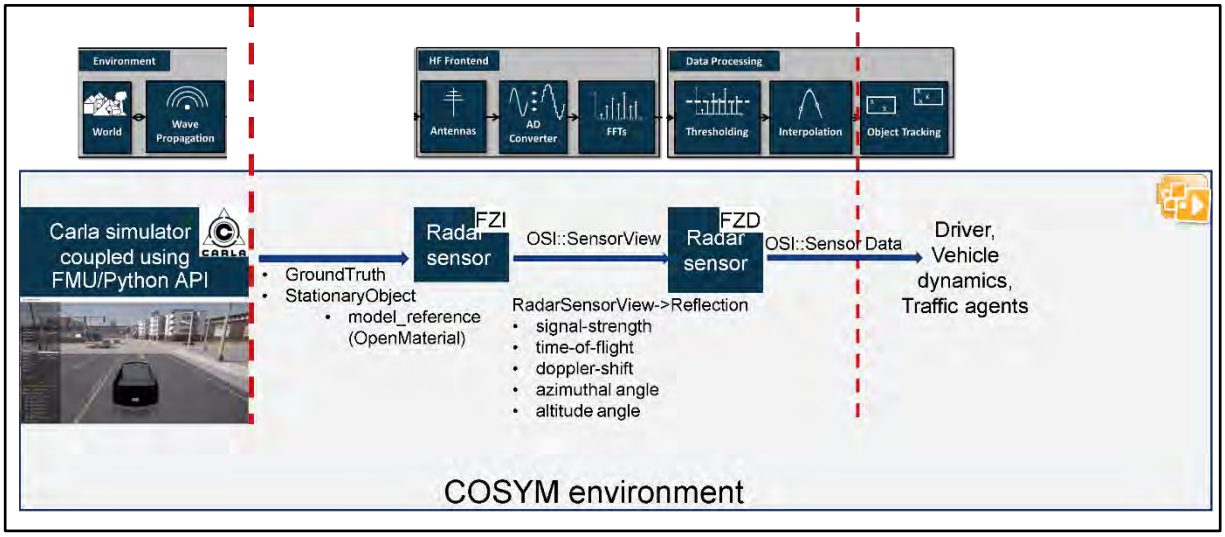


Abbildung 218: Radarbasiertes Komponententest in der COSYM-Umgebung

Die ETAS-Toolkette unterstützt das Radarmodell von FZD und FZI (siehe Abbildung 218). Unter Meilenstein 3 wurde ein PoC (Proof of Concept) für die Radarsimulation entwickelt. CARLA, als Teil des Umgebungssimulators, unterstützt bei der Welt-Implementierung und Wellenausbreitung. Es erzeugt eine Kegelansicht der Laser-Strahlen. Das funktioniert nach dem Raycasting Verfahren, welches in CARLA implementiert ist. Die Laserstrahlen bilden eine 2-dimensionale Schicht, die aus Informationen wie der Geschwindigkeit des Objekts mit dem Ego-Fahrzeug, dem Azimut-Winkel und dem Höhen-Winkel, dem Einbauort relativ zum Objekt usw. besteht. Die der Anforderung entsprechende Nachrichtenausgabe durch die radarbasierten Modelle wird untersucht. Es ist weiter in der Pipeline, diesen Satz von Skalaren

als Ausgabe aus dem Umgebungssimulator in Form einer OSI::SensorView-Nachricht zu senden. Gemäß der Eingabeanforderung der Radarmodelle enthält die OSI::SensorView-Nachricht die „Ground-Truth“, die OSI-Reflexionsnachricht, die aus der Signalstärke, der Laser-Signal-Laufzeit, der Dopplerverschiebung, dem Azimut-Winkel und dem Höhenwinkel besteht.

OpenMATERIAL mit Unreal Engine
OpenMATERIAL ist ein 3D-Modellaustauschformat, das die physikalischen Materialeigenschaften für die virtuelle Entwicklung, den Test und die Validierung bereitstellt. Die CARLA-Umgebungssimulations-Umgebung basiert auf der sehr realistischen Visualisierung für Video-Spiele „Unreal Engine“. „Unreal Engine“ unterstützt den Import von Objekten, die auf dem Dateiformat ".gltf" basieren. Diese Objekte könnten innerhalb von CARLA als Assets verwendet werden, z. B. Verkehrsagenten, Umgebungsobjekte.

Integration der Radarmodelle
Konzeptionell erleichtert der Umgebungssimulator das Raycasting/ -tracing. Die Radar-Reflektionen als Ergebnis der Wellenausbreitung wird an das FZI-Radarmodell gesendet. Derzeit benötigt das FZI-Radarmodell in der Modellentwicklungsphase die Objekt-Informationen aus Objekten vom Typ ".stl". Da die ETAS-Lösung den Import von benutzerdefinierten Objekten nicht ermöglicht, wurde die Integration des FZI-Radar-Modells nicht in Betracht gezogen und die Ergebnisse wurden nur konzeptionell veröffentlicht.

2.3.3.4.9 Bewertung der Ergebnisse des SUC 3

Als Fazit und Ergebnis kann folgendes festgehalten werden:

- Die Simulation Use Cases demonstrieren reale Anwendungsfälle
- Verschiedene Werkzeugketten, einschließlich Open-Source-Tools, wurden eingesetzt
- Weit verbreitete Nutzung von Standards, einschließlich der vom Projekt SET Level vorgeschlagenen Erweiterungen
- Es konnten Erweiterungen von Normen zu verschiedenen Normungsprojekten beigetragen werden
- Die Simulationsmodelle wurden Open Source zur Verfügung gestellt
- Die Anwendbarkeit der SET Level-Ergebnisse wurde in der Praxis gezeigt
- Wissen und Erfahrung wurden aufgebaut, gesteigert und dokumentiert.

2.4 Teilprojekt 5: Einbettung und kritische Reflexion

2.4.1 Übersicht über Inhalte und Ziele von TP 5

Das Teilprojekt „Einbettung und kritische Reflexion“ setzte auf den Resultaten der TP 1 bis 4 auf. Zum einen prüfte und bewertete es die Resultate intern, zum anderen unterstützte es die Industriepartner bei der exemplarischen Umsetzung in der Praxis und führte die daraus gewonnenen Erkenntnisse in das Projekt zurück. Damit hatte es eine zentrale, projektübergreifende Klammerfunktion über die einzelnen Teilprojekte des Projektes SET Level.
Das Teilprojekt war in zwei Arbeitspakete gegliedert:

AP 5.1 Proof of Concept
Das Arbeitspaket „Proof of Concept“ hatte die zentral übergreifende Aufgabe im Projekt, die in den Teilprojekten 1 bis 4 erstellten Artefakte (Simulationsmethodik, Szenariobeschreibung, Architektur, Modelle, Spezifikationen, Templates, exemplarische Modelle, Werkzeugketten) zu evaluieren, in einem projektinternen „Proof of Concept“ anzuwenden und einer kritischen Reflexion zu unterziehen. Die Ergebnisse dieser Bewertung und Überprüfung wurden zeitnah an die federführenden Teilprojekte zurückgegeben und für die Ausrichtung der weiteren Arbeiten verwendet.

Ein enger Bezug der Aktivitäten von AP 5.1 entwickelte sich zu den beispielhaften Anwendun-gen in den SUCs. Durch die Realisierung der Demonstratoren wurde ein beträchtlicher Teil der Prüfungen bezüglich der technologischen Konsistenz und der praktischen Nutzbarkeit abgedeckt.

Für die weiteren Prüfungen wurde ein allgemeiner Referenzprozess definiert, der für die einzelnen Reviewgegenstände spezifisch angepasst wurde. Damit wurde ein weitgehend einheitliches Vorgehen der beteiligten Projektpartner erreicht. Zu den wichtigsten Ergebnissen, die über die Werkzeuge und Demonstrationsketten hinaus im Verlauf des Projektes im Fokus des AP 5.1 standen, gehören die durchgängige Testbeschreibung, die Sprachmittel zur Beschreibung von Szenarien, die Simulationsarchitektur, der Credible Simulation Process und der Entwicklungsprozess von Modellen der Simulation

AP 5.2 Informationsgewinn aus den Erkenntnissen der Industriepartner aus der firmenspezifischen Erprobung
Entwicklung eines Einführungs- und Unterstützungskonzept mit Hilfe eines generisches Einführungsprozesses. Aufnahme und Dokumentation der Anforderungen der Industriepartner an die Methoden, Modelle, Prozesse, sowie die Werkzeuge und deren Schnittstellen („Artefakte“). Anonymisierung und Bewertung dieser Erkenntnisse. Industrielle Erprobung von Projektergebnissen und anschließende anonymisierte Rückmeldung der Ergebnisse. Inhaltliche Abstimmung mit anderen Forschungsvorhaben wie z. B. dem Projekt VVMethoden oder ausgewählten Projekt von prostep ivip, z. B. SmartSE.

2.4.2 Ergebnisbeiträge von TP 5

Aufgrund der vorwiegend projektinternen Ausrichtung des **AP 5.1** *Proof of Concept* finden sich die Ergebnisse hauptsächlich indirekt in den Resultaten der anderen TPs. Projektextern relevante Ergebnisse bestehen hauptsächlich in der zusammenfassenden Beurteilung der Nutzbarkeit und Reichweite der Lösungsbausteine für das zentrale Anliegen von SET Level, eine Grundlage für modulare, flexible Simulationslösungen zu legen. Dies betrifft die

vorgeschlagene Architektur, ihre Umsetzbarkeit, die Austauschbarkeit von Komponenten und die werkzeugunabhängige Einbindung in der Verifikations- und Validierungsprozess.

Die Ergebnisse von **AP 5.2** *Informationsgewinn aus den Erkenntnissen der Industriepartner aus der firmenspezifischen Erprobung* finden sich in folgenden Ergebnisbeiträgen

- Generischer Einführungsprozess
- Anforderungserhebung
- Diskussionsplattform für die Industrie
- Industrielle Erprobung
- Baselining im Projekt
- Austausch mit anderen Projekten

2.4.3 Detaillierte Ergebnisbeiträge von TP 5

2.4.3.1 Proof of Concept

Wie oben dargestellt, fand der „Proof of Concept“ sowohl auf konzeptioneller wie auch auf praktischer Ebene statt. Der Prozess der Prüfung und Bewertung hat entsprechend zwei wesentliche Ausprägungen. Einerseits beschreibt dabei der erste Strang die Prüfung einzelner ausgewählter Projektergebnisse auf Basis eines Prüfkriterienkatalogs in Form von Reviews mit den Ergebnisverantwortlichen. Andererseits beschrieb der zweite Strang das Vorgehen zur Prüfung von Projektergebnissen in deren Zusammenwirken anhand konkreter Umsetzungen durch die Simulation Use Cases, welches dem Vorgehen im AP 2.1 entspricht. Hierbei sollen die am „Proof of Concept“ beteiligten Simulation Use Cases speziell durch die durchgängige Dokumentation der Projektergebnisse unterstützt werden. Beide Ausprägungen finden sich in Top-Level-Sicht in Abbildung 219. Der Hauptprozess für den ersten Strang mit seinen In- und Outputs ist dabei oben dargestellt. Der zweite Strang befindet sich mit den zugehörigen In- und Outputs im unteren Teil der Abbildung.

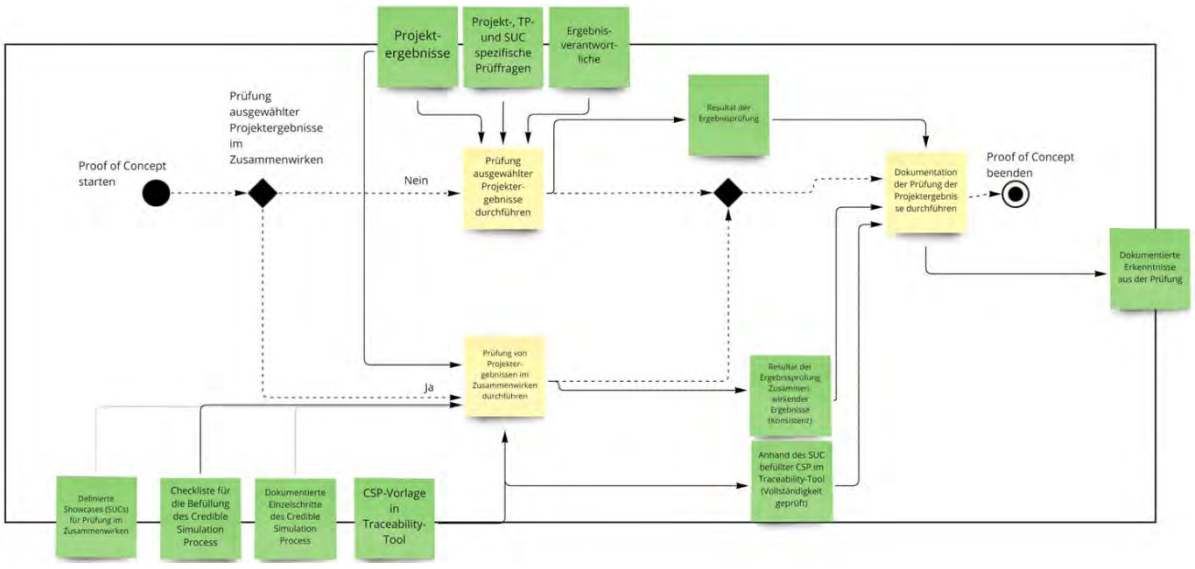


Abbildung 219: Top-Level-Sicht des Referenzprozesses zum „Proof of Concept“

Für die Durchführung des Prozesses zum Prüfen der Projektergebnisse werden als Inputs die Projektergebnisse selbst, deren Verantwortliche und die ergebnisspezifischen Prüffragen benötigt. Beim Durchlaufen des Prozesses wird entschieden, ob die Ergebnisse bereits reif genug für eine Prüfung sind. Zusätzlich wird entschieden, ob gegebenenfalls noch weitere Prüffragen für das jeweilig zu prüfende Artefakt aufgenommen werden sollen. Die Entscheidung

beziehungsweise Abstimmung darüber findet in Zusammenarbeit mit den Ergebnisverantwortlichen statt. Sie haben somit die Möglichkeit, zusätzlich eigene Fragen zum Review beizusteuern. Mit Hilfe der daraus entstandenen Fragen wurde das eigentliche Review durchgeführt. Im Anschluss werden die jeweiligen Ergebnisse des Reviews dokumentiert und damit der Prozess verlassen. Abbildung 220 zeigt die generische Darstellung eines Durchlaufs. Für jeden Prozessschritt wurde zusätzlich eine exemplarische Beschreibung des Prozessschrittes mit dessen In- und Outputs erstellt. Dabei wurden für die In- und Outputs konkrete Beispiele oder Erläuterungen für das Durchlaufen des Prozesses beschrieben. Abbildung 221 zeigt exemplarisch den Schritt „Durchführung der Prüfung auf Basis ausgewählter Fragen“ mit den Beispielen und Erläuterungen im unteren Teil.

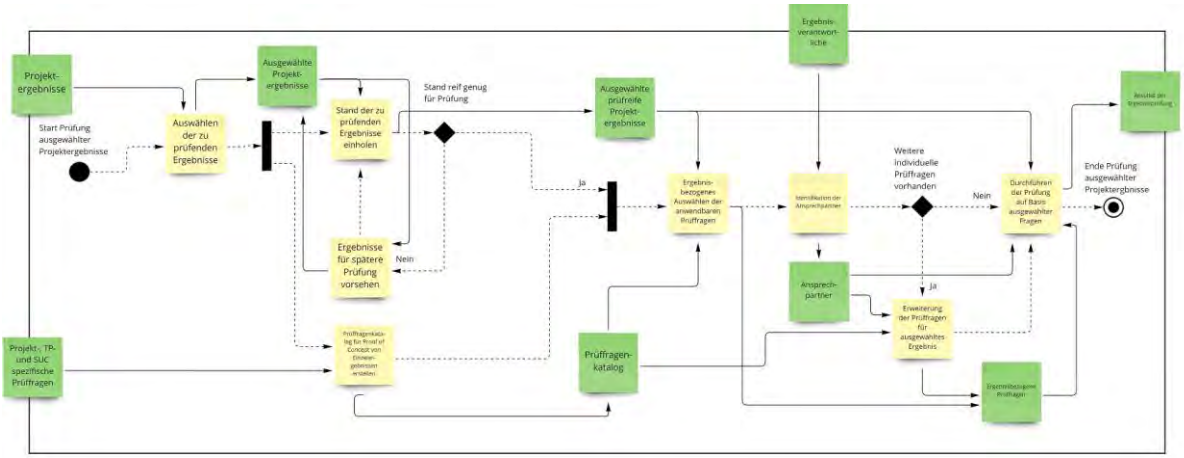
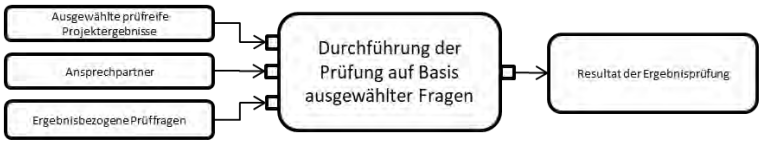


Abbildung 220: Prüfung einzelner Projektergebnisse



Input	Generisch	Spezifisch /Beispiel
	Ausgewählte prüfreife Projektergebnisse	Beispiel: Integrative Simulationsarchitektur
	Ansprechpartner	Verantwortliche für das jeweils zu prüfende Projektergebnis
	Ergebnisbezogene Prüffragen	Beispiel: „Haben die SUCs die integrative Simulationsarchitektur umgesetzt (um Anwendbarkeit zu zeigen)?“
Output	**Generisch**	**Spezifisch /Beispiel**
	Resultat der Ergebnisprüfung	https://gitlab.setlevel.de/pm/tp5/documents-tp5/-/tree/master/02_Durchfuehrungen/AP5.1/Reviewgegenstände
Detaillierte Aktivitäten (falls vorhanden)		

Abbildung 221: Exemplarische Beschreibung eines konkreten Prozessschrittes

Für die Durchführung des Prozesses zur Prüfung der „Projektergebnisse im Zusammenwirken“ werden als Inputs, die Simulation Use Cases, die Checkliste für den Credible Simulation Prozess, die dokumentierten Einzelschritte des Prozesses, sowie die Vorlage für diesen im Traceability Tool benötigt. Beim Durchlaufen des Prozesses werden die einzelnen Artefakte der SUCs anhand des Credible Simulation Prozess dokumentiert und fehlende Inhalte für die Durchgängigkeit identifiziert, was durch die Checkliste unterstützt wird. Zuletzt besteht die Auswahlmöglichkeit diese Artefakte und deren Zusammenhänge im Traceability Demonstrator toolgestützt abzubilden. Die einzelnen Schritte des Prozesses sind in Abbildung 222

dargestellt. Der Prozess stellt dabei das Vorgehen in der prozesssicheren Gesamtbeschreibung in AP 2.1 dar. Die Anwendung wurde von AP 2.1 gesteuert und in enger Zusammenarbeit mit den im Bild dargestellten Arbeitspaketen durchgeführt. Zur Vollständigkeit ist im Prozess auch die Entscheidungsmöglichkeit einer Weiterarbeit im Rahmen der industriellen Erprobung enthalten und der damit verbundene mögliche Übergang zum Einführungsprozess dargestellt.
Für diesen Prozess wurden ebenfalls wie zuvor beschrieben, für jeden Prozessschritt zusätzlich eine exemplarische Beschreibung des Prozessschrittes mit dessen In- und Outputs erstellt. Auch dabei wurden für die In- und Outputs konkrete Beispiele oder Erläuterungen gegeben.

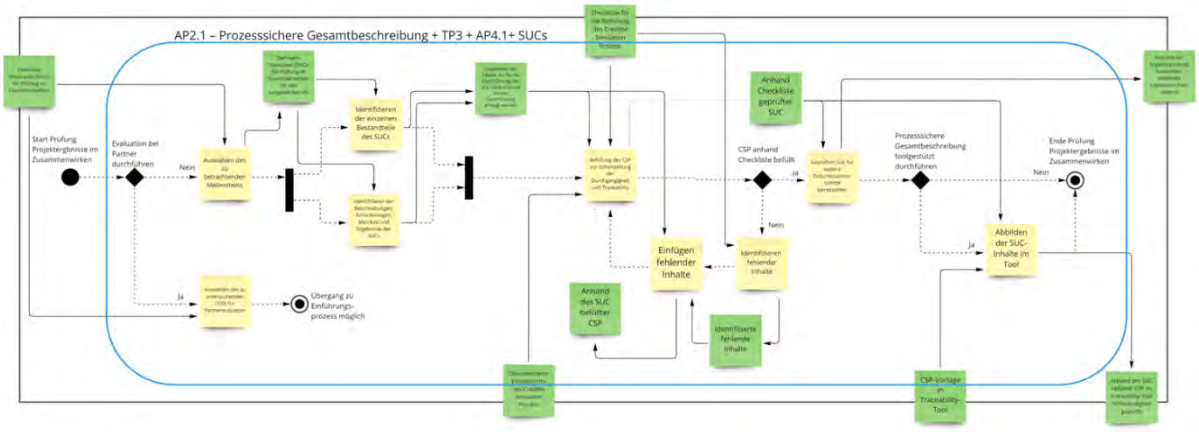
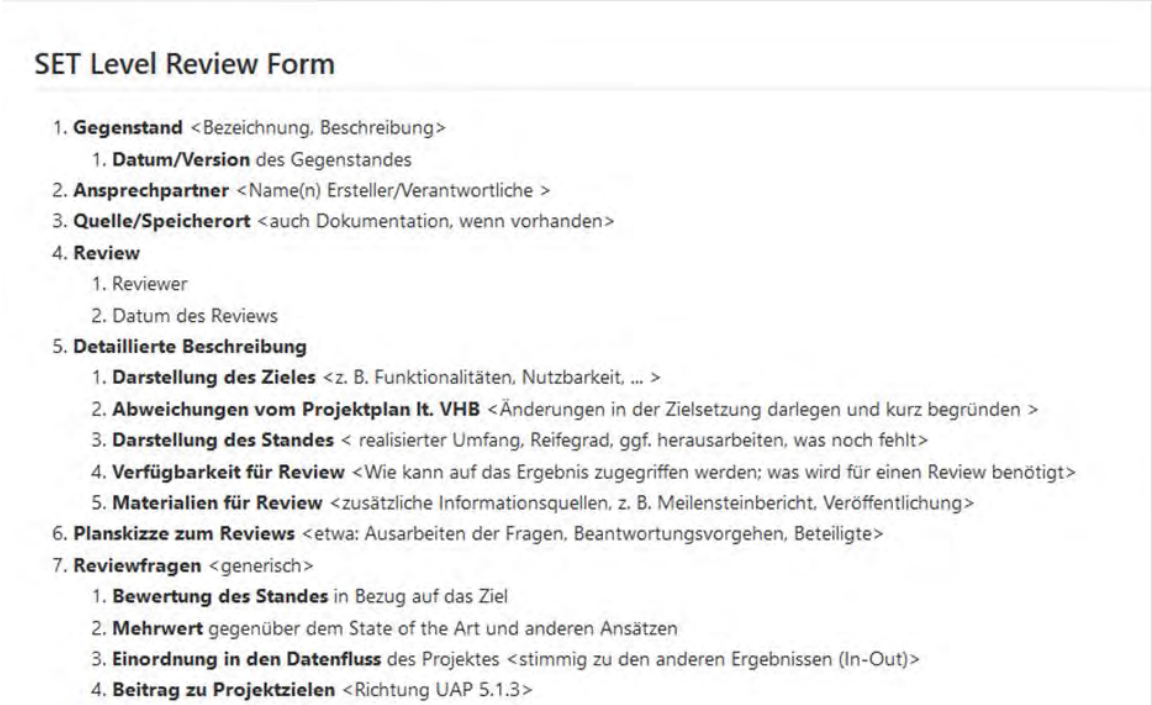


Abbildung 222: Prüfung von Projektergebnissen in deren Zusammenwirken

Für die Durchführung des „Proof of Concept“ wurden im SET Level Projekt TP- und Simulation Use Case spezifische Reviewfragen formuliert. Wie oben beschrieben wurden die unterschiedliche Reviewkriterien zu einem Fragenkatalog zusammengeführt und für die Verwendung während des Reviews in einem Reviewtemplate für den jeweils angenommenen Reviewgegenstand zusammengefasst. Hierbei konnten auch ergebnisspezifische Fragen der Teilnehmer des Reviews noch mit aufgenommen werden. Abbildung 223 zeigt beispielhaft einen Auszug aus dem für die Reviews verwendeten Template. Die einzelnen Reviews wurden jeweils mit den Reviewverantwortlichen aus dem AP 5.1 und den Ergebnisverantwortlichen basierend auf den erstellten Templates durchgeführt.



SET Level Review Form

1. **Gegenstand** <Bezeichnung, Beschreibung>
    1. **Datum/Version** des Gegenstandes
2. **Ansprechpartner** <Name(n) Ersteller/Verantwortliche >
3. **Quelle/Speicherort** <auch Dokumentation, wenn vorhanden>
4. **Review**
    1. Reviewer
    2. Datum des Reviews
5. **Detaillierte Beschreibung**
    1. **Darstellung des Zieles** <z. B. Funktionalitäten, Nutzbarkeit, ... >
    2. **Abweichungen vom Projektplan lt. VHB** <Änderungen in der Zielsetzung darlegen und kurz begründen >
    3. **Darstellung des Standes** < realisierter Umfang, Reifegrad, ggf. herausarbeiten, was noch fehlt>
    4. **Verfügbarkeit für Review** <Wie kann auf das Ergebnis zugegriffen werden; was wird für einen Review benötigt>
    5. **Materialien für Review** <zusätzliche Informationsquellen, z. B. Meilensteinbericht, Veröffentlichung>
6. **Planskizze zum Reviews** <etwa: Ausarbeiten der Fragen, Beantwortungsvorgehen, Beteiligte>
7. **Reviewfragen** <generisch>
    1. **Bewertung des Standes** in Bezug auf das Ziel
    2. **Mehrwert** gegenüber dem State of the Art und anderen Ansätzen
    3. **Einordnung in den Datenfluss** des Projektes <stimmig zu den anderen Ergebnissen (In-Out)>
    4. **Beitrag zu Projektzielen** <Richtung UAP 5.1.3>

Abbildung 223: Beispielhafte Darstellung des Reviewtemplates

Basierend auf den Reviews sollen hier einige der wesentlichen Erkenntnisse exemplarisch erläutert werden. Die Durchführung der Simulation Use Cases ermöglichte eine erfolgreiche interne Erprobung verschiedenster Projektaktivitäten. Dazu zählen unter anderem die aktive Nutzung und Integration verschiedener im Projekt entwickelter Modelle, sowie die Anwendung des Credible Simulation Process. Zusätzlich konnte für die Simulation Use Cases 2 und 3 die Übertragbarkeit der Ansätze auf unterschiedliche Werkzeugketten gezeigt werden. Dies wurde auch im Review einer exemplarischen (vendorspezifischen) Werkzeugkette bestätigt. Zusätzlich wurden bei der Umsetzung der Simulation Use Cases verschiedenste Standards verwendet. Hier sind beispielweise OSI, FMI und die Open Standards, OpenDRIVE und OpenSCENARIO zu erwähnen. Damit konnte auch die Einsetzbarkeit dieser Standards untersucht und sichergestellt werden. Insbesondere wurde eine OSI-Kommunikation in allen Simulation Use Cases als Schnittstelle verwendet.
Mit der integrativen Simulationsarchitektur konnte eine herstellerunabhängige Sicht von Simulationslösungen erstellt werden, die gemeinsame, weitgehend auf Standards basierende interne und externe Schnittstellen benennt. Sie erlaubt eine detaillierte Anknüpfung an den Credible Simulation Process. Sie unterstützt die Fehlerminimierung und Automatisierung z. B. durch die Ausleitung von SSP-Dateien für Simulationssoftware. Durch die Verknüpfung der einzelnen Simulationsläufe mit der Architektur des realen Fahrzeugs gemäß SaFAD-Paper (SaFAD, 2019) ist die Übertragbarkeit von Simulationsaussagen auf die Realität sichergestellt. Der industrielle Mehrwert der Architektur wird zudem durch den ihr zu Grunde liegenden MBSE-Gedanken unterstützt. Der dabei umgesetzte RFLP-Ansatz bietet eine gute Basis für die Diskussion über die einzelnen Architekturelemente, da die einzelnen Ebenen eindeutig voneinander abgegrenzt werden können. Speziell die Einbeziehung des gesamten Systems inklusive seiner Umgebung trägt dabei dem Trend vom klassischen Ansatz hin zum software-defined Vehicle Rechnung, da damit die deutlich zunehmenden Interaktions- und Kommunikationspunkte mit der Umgebung erfasst werden können.
Im Rahmen einer beispielhaften Werkzeugkette konnte gezeigt werden, dass sich die OSI-Kommunikation für die Zusammenschaltung unterschiedlicher im Projekt entwickelter Modelle anwenden lässt. OSI definiert sowohl semantische und technische Ebene. Dadurch, dass die

Modelle über eine OSI-Kommunikation angebunden werden können, wird dabei die Anforderung für die Komposition auf semantischer Ebene und technischer Ebene erfüllt. Hinsichtlich der Anwendung der Architektur ergibt sich eine Einschränkung. Die im Projekt entwickelte Architektur findet in den Simulation Use Cases Anwendung und die Vendorenwerkzeugketten sind darin eingebunden. Allerdings wird eine einheitliche Architektur über beziehungsweise innerhalb verschiedene Vendorentools als nicht sinnvoll angesehen.
Der Mehrwert in der Anwendung des Credible Simulation Process liegt in der Schaffung eines geleiteten Ansatzes, um das aktuell häufig dokumentenzentrierte Vorgehen durchgängig und prüfbar in einem konsistenten Prozess abzubilden. Dies konnte auch durch seine Anwendung in den Simulation Use Cases gezeigt werden. Zusätzlich konnte ein industrieller Mehrwert durch das Mapping zwischen den Phasen des Credible Simulation Process und der Simulationsarchitektur identifiziert werden. Insbesondere die Möglichkeit der Ausleitung als SSP wird dabei als ein großer Mehrwert angesehen, da durch die Erzeugung von Metamodellen die Wiederverwendbarkeit signifikant erhöht wird und der Wiederverwendungsgedanke nicht nur abstrakt, sondern zentral verankert ist. Zusätzlich kann die Anwendung des Credible Simulation Process durch den im Projekt entwickelten Traceability Demonstrator toolgestützt unterstützt werden. Diese Unterstützung wird durch ein im Tool hinterlegtes Template vereinfacht. Ein weiterer Mehrwert, auch im Zusammenspiel mit dem Credible Simulation Process, entsteht durch die Reduzierung der Komplexität durch die Schaffung eines Informationsnetzwerks über bestehende Systeme und Datenbestände. Dazu schafft das Tool eine Verlinkung von Artefakten und persistiert Abhängigkeiten. Die Anwendbarkeit konnte im Rahmen verschiedener Dokumentationen der Simulation Use Cases anhand des Credible Simulation Process gezeigt werden.
Nach dem formalen Ende der Arbeiten zur „Prozesssicheren Gesamtbeschreibung“ im AP 2.1 wurde dieses Thema noch einmal im AP 5.1 aufgegriffen. Da die Simulation Use Cases für den „Proof of Concept“ verwendet wurden, liefert auch deren durchgängige Beschreibung anhand des im Projekt SET Level entwickelten Prozesses und die darauf aufbauende Abbildung im Traceability Tool einen Beitrag zur internen Erprobung und kritischen Reflexion der Ergebnisse. Im Rahmen der erneuten Aufnahme der Arbeiten konnten damit auch die aktuellsten Stände der Dokumentation der Simulation Use Cases erfasst werden. Zusätzlich war die Wideraufnahme der Arbeiten dadurch bedingt, dass im Traceability Tool ein neues Datenmodell zur Verfügung stand. Es erfolgte daher eine Überarbeitung der in GitLab zur Verfügung gestellten Dokumentation der Simulation Use Cases im Traceability Tool. Die im Tool neu hinzugekommenen Fähigkeit des Ziehens von konkreten Baselines konnte dabei für verschiedene Meilensteine angewendet werden. Die im Traceability Tool durchgeführte Dokumentation wurde in XML und als zip ausgeleitet und auf GitLab für die jeweiligen Simulation Use Cases abgelegt. Dies ermöglicht den Projektpartnern auch nach Projektende, den Zugriff auf die Dokumentation in einem neutralen Austauschformat. Abbildung 224 zeigt exemplarisch einen Subschritt des Credible Simulation Process nach dem die Dokumentation für den Simulation Use Case 1 im Tool mit dem neuen Datenmodell durchgeführt wurde.

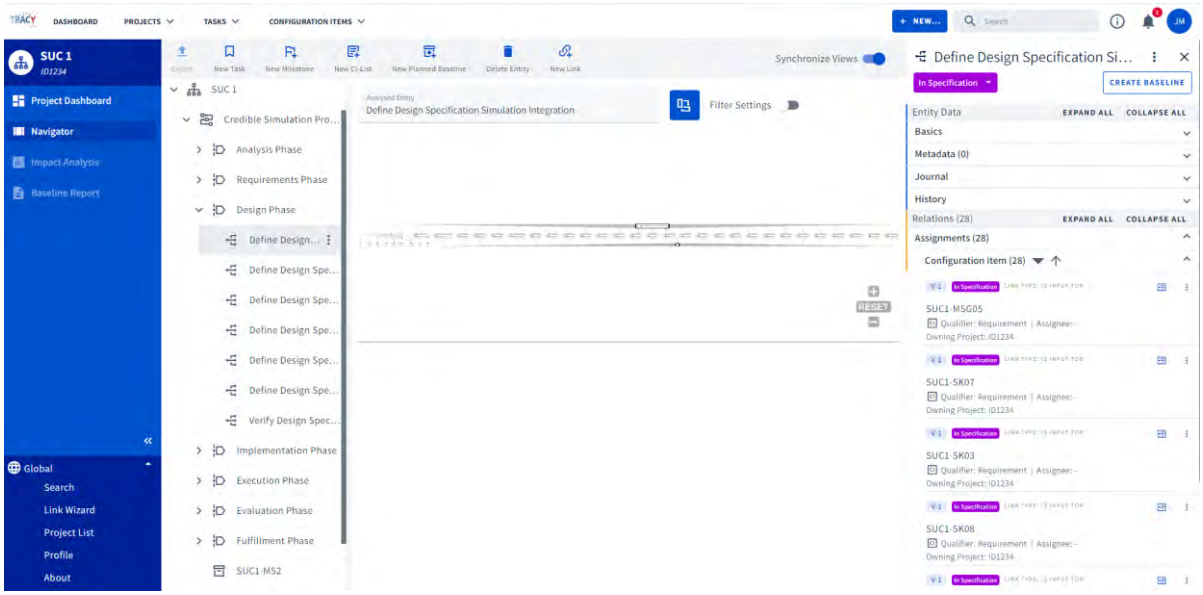


Abbildung 224: Dokumentation des Simulation Use Case 1 im Traceability Demonstrator anhand des Credible Simulation Process (ausgewählter Subschritt)

2.4.3.2 Generischer Einführungsprozess

Mit Hilfe des generischen Einführungsprozesses für SET Level-Artefakte, wird es allen am SET Level-Projekt beteiligten Unternehmen ermöglicht, die in den Simulation Use Cases (SUCs) erstellen Artefakte (Modelle, Parametersätze, Prozesse, Methoden etc.), im Firmenumfeld einzuführen und zu erproben. Des Weiteren bietet der generische Einführungsprozess den beteiligten Unternehmen die Möglichkeit, die Erkenntnisse, die sie bei der firmeninternen Erprobung gewonnen haben, dem Projekt zurückzugeben und so den Entwicklungs- und Verbesserungszyklus der Artefakte zu schließen.
Der generische Einführungsprozess für SET Level-Artefakte im firmenspezifischen Kontext ist in die nachfolgenden sieben Haupt-Prozessschritte unterteilt:

- Auswahl und Mapping der für den Simulation Use Case notwendigen Artefakte auf vorhandene Artefakte im Unternehmen (Modelle, Werkzeuge, Werkzeugketten, Schnittstellen, Parameter)
- Identifikation der für den SET Level Simulation Use Case fehlenden firmeninterne Artefakte (Modelle, Werkzeuge, Werkzeugketten, Schnittstellen)
- Aufbau der Simulationsumgebung für den Simulation Use Case
- Durchführung / Ausführung des Simulation Use Cases
- Aufbereitung und Dokumentation der Ergebnisse
- Aufgetretene Fehler dokumentieren
- Durchführung der Nachverfolgbarkeit (Traceability) der Entscheidungen, der eingesetzten Modelle, Werkzeuge, Werkzeugketten, Schnittstellen etc. für Simulation Use Case

Jedem Prozessschritt sind spezifische Ein- und Ausgabe-Artefakte zugeordnet. Diese werden von dem Prozessschritt als Input benötigt, bzw. als Output generiert. Der generische Einführungsprozess kann für alle im Projekt definierten SUCs und den darin benötigten Artefakten gleichermaßen eingesetzt werden. Der generische Einführungsprozess muss dafür nicht angepasst werden. Prinzipiell ist der Einführungsprozess derart generisch gehalten, dass auch andere Simulationen damit eingeführt werden können.

Die Abbildung 225 zeigt den generischen Einführungsprozess, bestehend aus den obigen Prozessschritten, inklusive aller Ein- und Ausgabe-Artefakte und Entscheidungen, auf der obersten Ebene. Die Darstellung des Prozesses erfolgt als UML/SysML-Aktivitätsdiagramm.

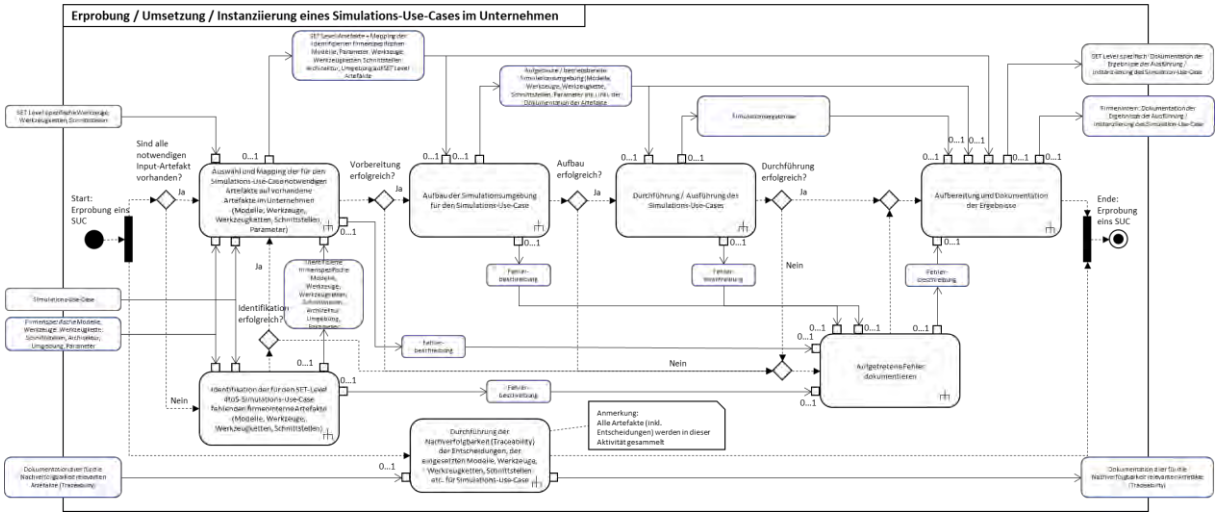


Abbildung 225: Der generische Einführungsprozess für SET Level-Artefakte

Für alle Prozessschritte sind die Ein- und Ausgaben (Artefakte), unterteilt im Projekt SET Level-spezifische (blauer Rahmen) und firmenspezifische Artefakte (grauer Rahmen), explizit modelliert und mit einer Kardinalität versehen. Die Beschreibung der Ein- und Ausgabe-Artefakte ist zum einen direkt am Ein- bzw. Ausgang („Port“) des Prozessschritts angegeben und zum anderen in einer Tabelle beschrieben. Dabei geht die tabellarische Beschreibung über die Beschreibung der Artefakte im Prozess hinaus, z. B. um die für einen spezifischen SUC benötigten Eigenschaften von Artefakten oder die Verantwortlichkeiten für ein spezifisches Artefakt.
Zur Vereinfachung der firmenspezifischen Anwendbarkeit wurde im Rahmen des Projekts entschieden, den allgemeinen generischen Einführungsprozess, ohne den Prozessschritt für die Fehlerdokumentation, zu verwenden. Des Weiteren wurden einige Entscheidungspunkte entfernt. Die Abbildung 226 zeigt den vereinfachten generischen Einführungsprozess für SET Level-Artefakte aus der Abbildung 225, ohne den Prozessschritt der Dokumentation und den zusätzlichen Entscheidungen (vgl. X1 - X5 und X7).

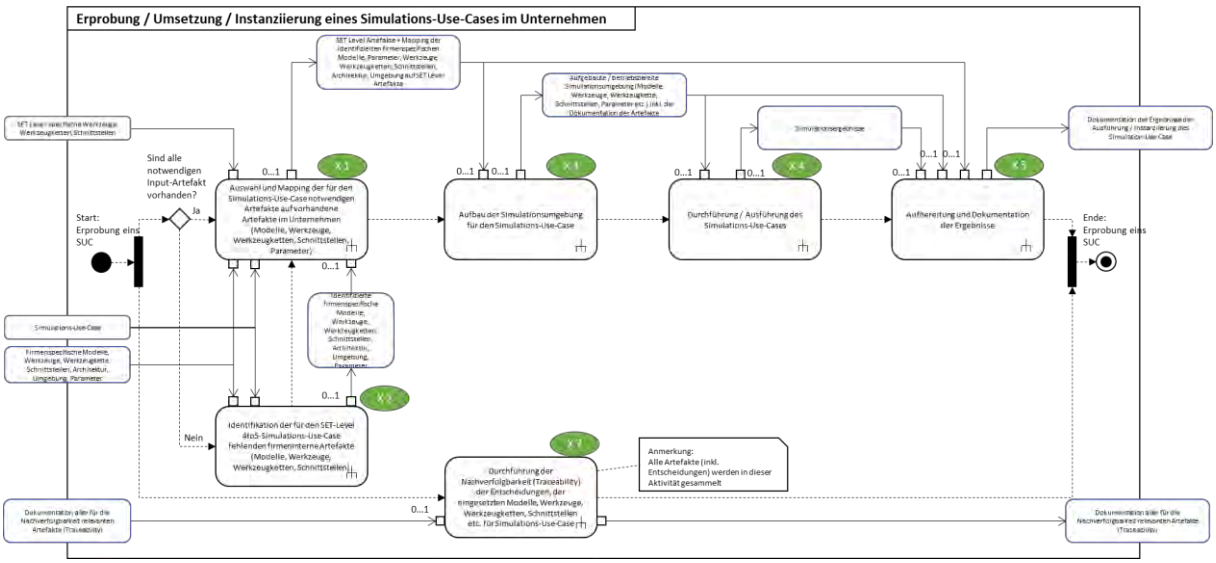


Abbildung 226: Vereinfachte Darstellung des generischen Einführungsprozesses

2.4.3.3 Anforderungserhebung

Abbildung 227 zeigt schematisch das gesamte Vorgehen von Erhebung bis Priorisierung der Anforderungen die aus Sicht der im Projekt SET Level beteiligten Industriepartner eine hohe Priorität im Hinblick auf automatisiertes Fahren haben.

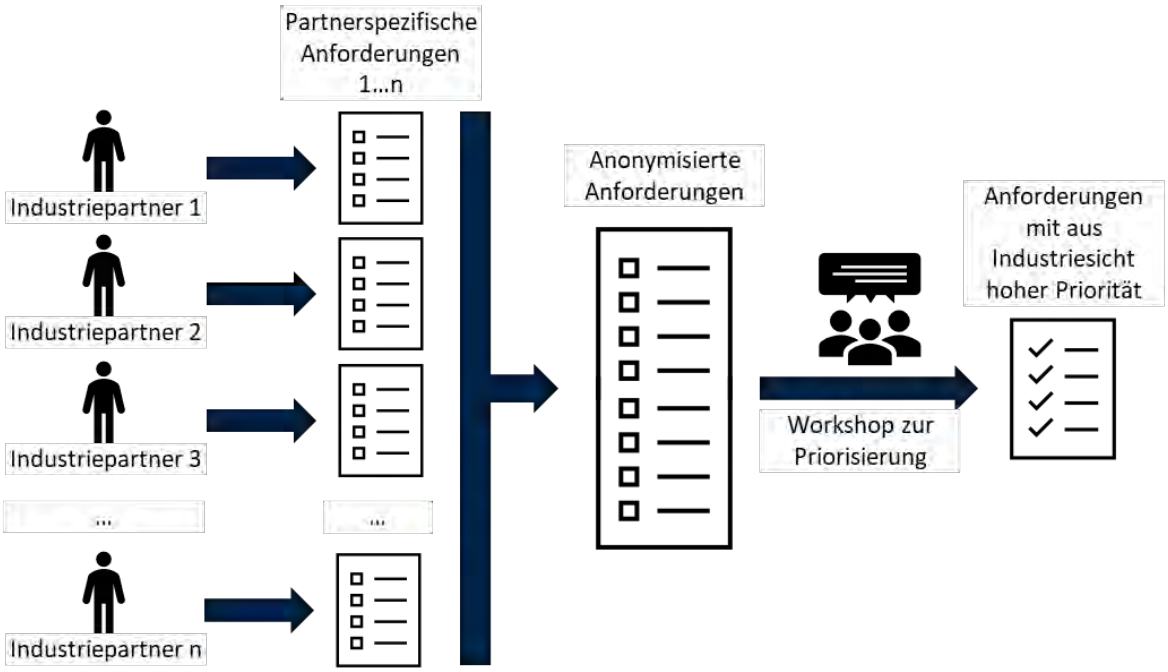


Abbildung 227: Vorgehen zur Ermittlung von Anforderungen die aus Industriesicht eine hohe Priorität haben

Für die Erhebung der Anforderungen aus der Industrie wurden im Projekt SET Level von PROSTEP separate Workshops einzeln mit den jeweiligen Industrievertretern in den Häusern durchgeführt. Dabei fand eine Erhebung des aktuellen IST-Zustandes sowie des angestrebten SOLL-Zustandes in Bezug aus SET Level relevante Themengebiete statt.
Dieses Vorgehens verfolgte dabei mehrere Ziele. Zum einen konnte dadurch der firmeninterne Austausch innerhalb der beteiligten Fachabteilungen gefördert und ein Bewusstsein für die jeweiligen aktuellen Stände, eingesetzten Tools, verwendeten Methoden und Prozesse

der Abteilungen geschaffen werden. Zum anderen konnte ein gemeinsames Bild, der teils innerhalb der Abteilungen unterschiedlichen die Erwartungen und Ziele erarbeitet werden. Diese gemeinsame Erarbeitung des SOLL-Zustandes förderte auch das Problembewusstsein und -verständnis unter den Beteiligten über Abteilungsgrenzen hinweg. Darüber bildeten die Inhalte des SOLL-Zustandes die Grundlage für die nachfolgende partnerübergreifende Anforderungserhebung. Abbildung 228 zeigt exemplarisch einen Teil der aufgenommenen Inhalte eines Workshops. Die konkreten Inhalte sind aus Vertraulichkeitsgründen unkenntlich gemacht.

Abbildung 228: Exemplarische Ergebnisse eines Workshops (aus Vertraulichkeitsgründen unkenntlich gemacht)

Die auf diesem Wege erarbeiteten Inhalte wurden aufbereitet und in ein für diesen Zweck definiertes Datenmodell in Cameo Systems Modeller überführt. Hierbei sind die Inhalte der einzelnen Partner für die spätere Ausleitung wieder getrennt im Modell abgebildet. Die innerhalb des Modells dokumentieren Artefakte wurden jeweils partnerspezifisch neutralisiert, zum Beispiel durch Entfernung von Produkt- oder Projektnamen aus den Inhalten, die einen direkten Rückschluss auf den jeweils beteiligten Partner zulassen.
Daran anschließend fand eine für jeden Partner separat durchgeführte Freigabe der neutralisierten Ergebnisse statt um sicherzustellen, dass keine Inhalte in das Projekt zurückgespielt werden, die vom jeweiligen Partner als intern oder vertraulich eingestuft werden. Abbildung 229 zeigt die Grobstruktur des im Cameo Systems Modeller definierten Datenmodells.

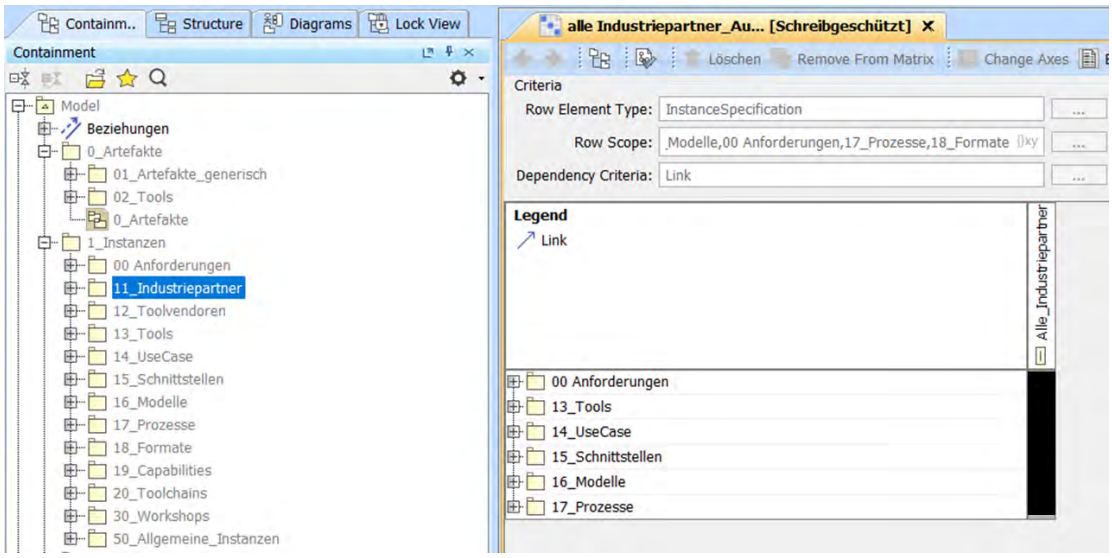


Abbildung 229: Grobstruktur des Datenmodells in Cameo

Im Anschluss an die Verifikation der Ausleitungen der neutralisierten Auswertungen der Workshops mit den Industriepartnern wurden diese für eine partnerübergreifende Priorisierung aufbereitet. Ziel ist auch bei einer gemeinsamen Betrachtung der erarbeiteten Inhalte keine direkten Rückschlüsse auf Aussagen, Probleme und Erwartungen der jeweiligen Industriepartner zu ermöglichen. Insgesamt wurden bei diesem Vorgehen mehr als 400 Inhalte gemeinsam mit allen beteiligten Industriepartnern erarbeitet.
Die Gesamtheit der Inhalte wurde themenspezifisch kategorisiert und nach IST-Zustand und SOLL-Zustand unterschieden, für die partnerübergreifende Priorisierung ausgeleitet. Abbildung 230 zeigt einen Ausschnitt aus dieser Ausleitung. Die Inhalte sind aus Vertraulichkeitsgründen unkenntlich gemacht.

Abbildung 230: Ausleitung (aus Vertraulichkeitsgründen unkenntlich gemacht)

Basierend auf der neutralisierten Ausleitung wurde die Priorisierung der Anforderungen in gemeinsamen, übergreifenden Workshops zusammen mit den im Projekt involvierten Firmenvertretern der jeweiligen Industriepartner durchgeführt. Generell lassen sich aus der Anforderungserhebung und der durchgeführten Priorisierung, die Anforderungen und Erwartungen der Industriepartner an das Projekt wie folgt formulieren:

Die Erwartung aus der Industrie an das Projekt SET Level war, anwendbare Werkzeuge und Methoden für Szenarien und Testcases bereitstellen, sowie eine breite Unterstützung von Standards in Simulationstools zu gewährleisten.

Die Ergebnisse der Priorisierung wurden auf GitLab abgelegt und damit dem Projekt für alle Partner zugänglich zur Verfügung gestellt. Aufbauend auf den in den Industrieworkshops erhobenen Anforderungen und der anschließenden Priorisierung, im Hinblick auf automatisiertes Fahren, fand eine Generalisierung der Anforderungen sowie eine Einteilung nach Themenclustern statt. Die durchführte Generalisierung und Zuordnung wurde mit den im Projekt involvierten Industriepartnern abgestimmt. Die Ergebnisse der Arbeiten sind in Abbildung 231, nach Themenclustern sortiert, abgebildet.

Im Rahmen der Generalisierung, wurden dabei auch unter Einbeziehung der Ergebnisse des IST-Zustandes die Anforderungen an das Projekt bezugnehmend auf die Themencluster konkretisiert. Darüber hinaus erfolgte ein initiales Mapping der generalisierten und konkretisierten Anforderungen auf die einzelnen Arbeitspakete und Arbeitsgruppen des SET Level Projektes, wobei keine Wiedersprüche zu den Projektzielen identifiziert, werden konnten, da in den Anforderungen die aus Industriesicht hohe Priorität für automatisiertes Fahren haben, auch Anforderungen enthalten sind die klar nicht im Scope des SET Level Projektes liegen. Die Ergebnisse wurden im Rahmen eines Quartalsworkshop dem Projekt übergeben und für alle Partner zugänglich auf GitLab abgelegt. Die daraus resultierenden Handlungsschwerpunkte wurden im Rahmen des Workshops kommuniziert und an das Projekt übergeben.

SET Level Industriepartner

Themencluster	Szenarien	Methoden	Modelle	Validierung	Standardisierung	Testen
Anforderungen	Identifikation kritischer Szenarien	Sicherstellung Zulassungfähigkeit	Validierung von Modellen und Validierungsmetrik für Modelle		KPI	Testen der AD-Funktion
	Kreuzungsszenario mit Spurwechsel	Toolqualifizierung für virt. Simulation	Toolübergreifende Modelle	Validierte Verkehrsteilnehmer	Standardisierte Testdefintion	
	Szenariobeschreibungs-sprache	Modellentwicklung und Simulation		Validierte Schnittstellen	Schnittstellen (HiL, SiL und Sensormodelle)	Testen von Sensoren und Komponenten
	Szenariobeschreibung und Handling		Validierung von (Sensor-)modellen		FMI, OSI, OSC, ODR, OpenSCENARIO	Definierter Testprozess
			Freigabe und Spezifikation (für Sensormodelle)		Standardisierte modulare Simulationsarchitektur	
			Rohdaten für Sensormodelle			

Abbildung 231: Aus Industriesicht priorisierte Anforderungen an automatisiertes Fahren, geordnet nach Themenclustern

2.4.3.4 Anforderungskonsolidierung

Zur Unterstützung der industriellen Anwendbarkeit der Projektergebnisse wurde ein Prozess für den kontinuierlichen Austausch und das Review der Projektergebnisse erarbeitet und mit den TP-Leitern abgestimmt. Die daraus resultierenden Abstimmungen mit den Industriepartnern ermöglicht es, den Projektpartnern konkrete Projekt- oder Zwischenergebnisse zur Diskussion zu stellen. Dabei existieren prinzipiell zwei Pfade. Einerseits kann Feedback direkt innerhalb dieser Runden gegeben werden. Anderseits besteht die Möglichkeit, dass Feedback aus Vertraulichkeitsgründen an das TP 5 übergeben wird und dann in neutralisierter Form durch das TP 5 eingebracht wird.

Im Rahmen der Anforderungskonsolidierung wurden die Themengebiete Anforderungen, Architektur, Schnittstellen, sowie SRMD/Metadaten für Modelle behandelt. Darüber hinaus wurden die initialen Abstimmungsrunden für die anschließende industrielle Erprobung der SET Level Kooperationsprozesse mit dem SmartSE Referenzprozess zum Modellaustausch zusammen mit TP 3 durchgeführt.

2.4.3.5 Industrielle Erprobung

Ziel der industriellen Erprobung von SET Level Ergebnissen war es, die industrielle Anwendbarkeit der Ergebnisse zu unterstützen. Dazu wurden verschiedene Projektergebnisse einem Review durch die SET Level Industrievertreter unterzogen. Trotz des teilweisen unterschiedlichen Vorgehens im Rahmen der Reviews bzw. der Erprobung wurde grundsätzlich darauf geachtet, dass partnerspezifische Inhalte, soweit dies gefordert war, nur neutralisiert und nach Freigabe der Partner in das Projekt oder weitere Diskussionen eingebracht wurden. Damit sollte sichergestellt werden, dass keine Partnerinterna in das Projekt fließen.

2.4.3.5.1 Kooperationsprozess

Basierend auf den initialen Abstimmungsrunden im Rahmen der Industrieplattform, wurde in TP 5 zusammen mit TP 3 ein Konzept entwickelt, um einen Review des Zusammenspiels der Kooperationsprozesse zusammen mit den SET Level Industriepartnern durchzuführen. Hierzu wurde zusätzlich zu den in TP 3 erarbeiteten Prozessen, Credible Simulation Process und Credible Modelling Process der in der prostep ivip Arbeitsgruppe SmartSE entwickelte Referenzprozess mit eingebracht.

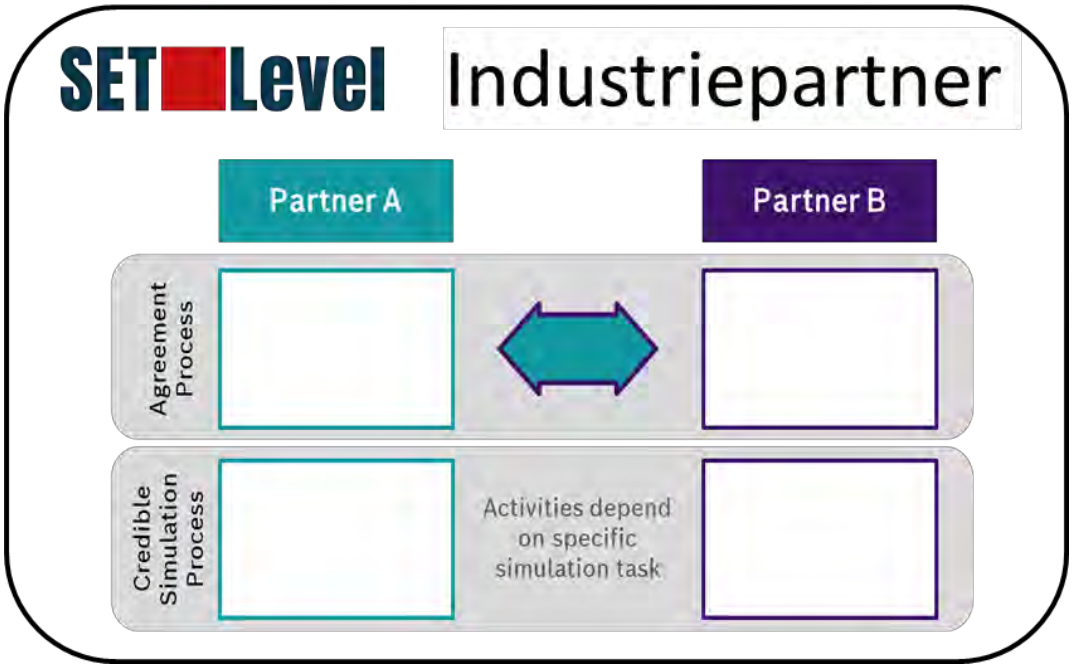


Abbildung 232: Abbildung beteiligter Partner und Hauptphasen des Kooperationsprozesses

Hintergrund der Definition des Kooperationsprozesses in TP 3 und der industriellen Erprobung der Anwendbarkeit des Prozesses in TP 5, ist der wachsende industrielle Trend hin zu gemeinsamen agilen Entwicklungen, was insbesondere im Bereich des automatisierten Fahrens gilt. Beispielhaft kann hier die gemeinsame Entwicklung einer automatisierten Fahrfunktion, die ein Kameramodell, als Wahrnehmungsanteil der Gesamtfunktion beinhaltet angeführt werden. Dieser Use Case zeigt die Notwendigkeit einer engen Zusammenarbeit und Koordination zwischen den beteiligten Partnern, im Gegensatz zu früheren Ansätzen, bei denen

das Kameramodell nach vorgegebener Spezifikation von einem Partner alleine entwickelt wurde.
Ein Kooperationsprozess besteht in der Regel aus einem Agreement-Prozess und einem Entwicklungsprozess. Er beginnt mit dem Agreement-Prozess, der zur Klärung und Festlegung der technischen und vertraglichen Rahmenbedingungen dient. Nachdem der Agreement-Prozess abgeschlossen ist, beginnt der eigentliche Entwicklungsprozess, der die gemeinsame Spezifikation, das Design und die Implementierung der Simulations- und Modellierungsaufgaben umfasst. In dem für die Erprobung im Projekt SET Level definierten Falle wird der Entwicklungsprozess durch Credible Simulation Process und Credible Modelling Process repräsentiert. Abbildung 232 zeigt neben den beiden Hauptphasen des Kooperationsprozesses auch die an der Erprobung beteiligten SET Level Industriepartner.
Um die industrielle Anwendbarkeit des Prozesses zu gewährleisten und sicherzustellen, dass der Prozess die Zusammenarbeit zwischen Industrieunternehmen unterstützt, wurden mehrere Workshops mit den Industrievertretern (siehe Abbildung 232) des SET Level Projektes durchgeführt. Ziel dieses Vorgehens, war es einerseits den Prozess im Allgemeinen zu diskutieren und ein gemeinsames Verständnis zu schaffen. Andererseits sollte der Prozess durch die beteiligten Industriepartner in einer möglichst realitätsnahen Konstellation durchgespielt werden. Dazu fanden die Workshops jeweils separat mit Paaren aus OEM und Zulieferer statt. Die Zusammenarbeit und Dokumentation fanden online über MS-Teams und das Tool Miro statt.
Die Ergebnisse der Einzelworkshops wurden jeweils für die beiden Partner separat erfasst und aufbereitet. Basierend auf, der aufbereiteten Rückmeldung an die Einzelgruppen fand durch diese, eine Freigabe statt um die gewonnenen Erkenntnisse in der gesamten Gruppe zu teilen und weiter diskutieren zu können. Insgesamt wurden dabei zusammen mit allen Partnern über 300 individuelle Rückmeldungen eingeholt. Für die weiterführende Bearbeitung fand eine Überführung und Konsolidierung in ein gemeinsames Miro-Board statt um eine weitere partnerübergreifende Zusammenarbeit zu ermöglichen. Ein Ausschnitt des Ergebnisses ist in Abbildung 233 dargestellt. Die genauen Inhalte sind aus Vertraulichkeitsgründen weichgezeichnet.

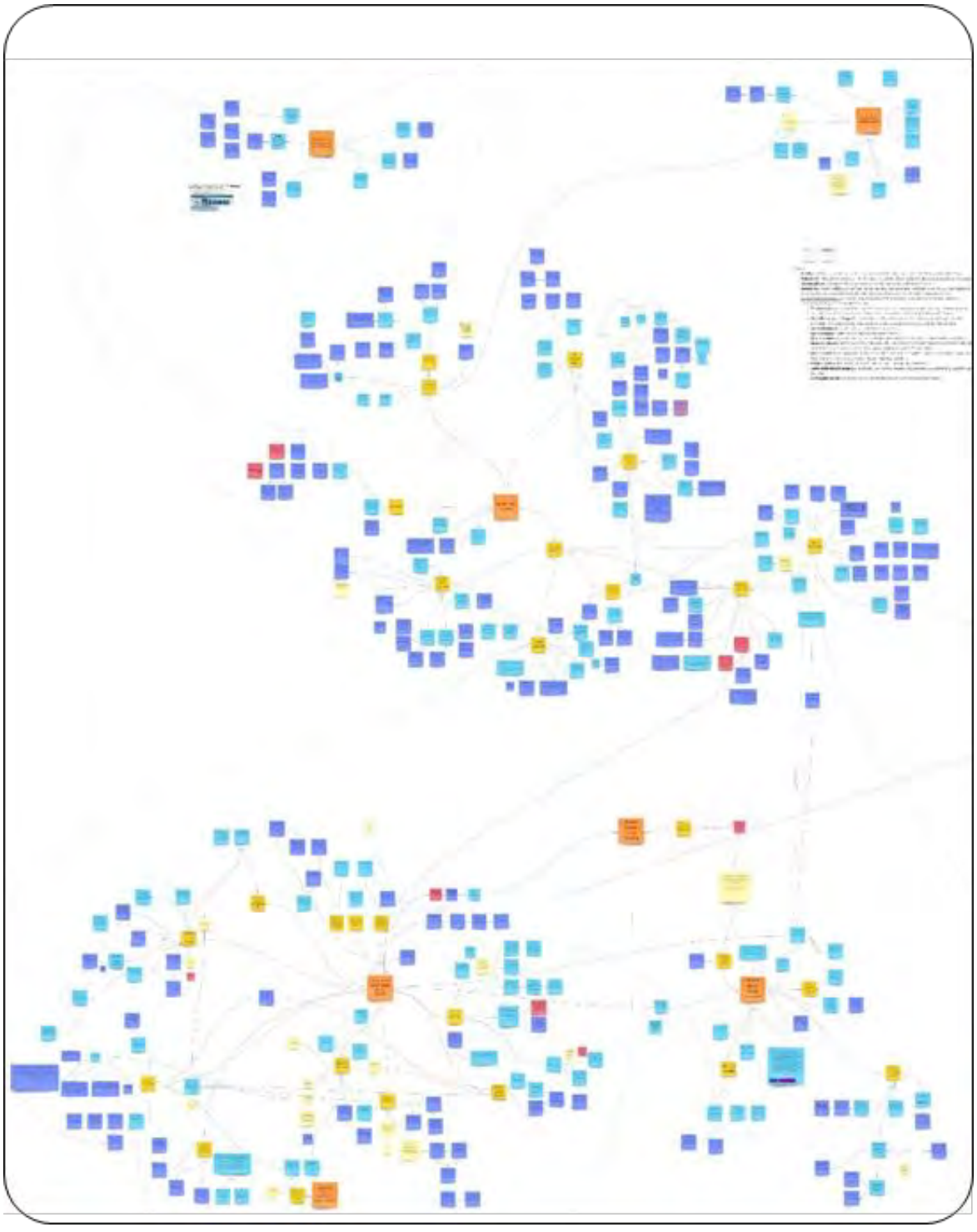


Abbildung 233: Auszug aus den konsolidierten Ergebnissen der Workshops (aus Vertraulichkeitsgründen weichgezeichnet)

Basierend auf den konsolidierten Ergebnissen konnte mit den SET Level Partnern eine Priorisierung der Erkenntnisse, in Bezug auf Zusammenarbeit und den Austausch zwischen Projektpartnern im realen Projektumfeld durchgeführt werden. Im Rahmen der Workshops und der anschließenden Priorisierung konnten jedoch nicht nur Herausforderungen, die sich durch die partnerübergreifende Zusammenarbeit bei der Entwicklung hochautomatisierter Systeme ergeben identifiziert werden, sondern es wurden auch darauf aufbauend Best Practices und Recommendations abgeleitet.

Nach Beendigung der Workshops und der gemeinsamen Priorisierung wurden die Ergebnisse an die Verantwortlichen im TP 3 übergeben und diese bei der Erstellung eines Dokuments, das speziell die Herausforderungen und Best Practices beschreibt, unterstützt. Darüber hinaus wurde über das Vorgehen und die Ergebnisse im Rahmen eines Vortrags auf dem SET Level Abschlussevents berichtet und zusammen mit dem TP 3 und Industrievertretern die Möglichkeit in Form eines Marktstandes geschaffen, die Diskussion in die Breite zu tragen.

2.4.3.5.2 Simulation / Modelling Request

Eine wichtige generelle Erkenntnis, die aus der industriellen Erprobung des SET Level Kooperationsprozesses gewonnen wurde, war das für den Austausch von Simulationsmodellen und der partnerübergreifenden Zusammenarbeit im Rahmen von Simulationen und deren Aufbau, ein einheitlicher Satz an Metadaten für die Übergabeartefakte definiert sein sollte. Dies unterstützt einerseits ein standardisiertes Vorgehen bei der Beauftragung von Simulationsaufgaben, sowohl innerhalb des Unternehmens als bei der partnerübergreifenden Zusammenarbeit. Der im TP 3 in entwickelte Simulation Request und Modeling Request definiert dazu ein Framework, in dem auch ein standardisiertes Datenmodell für den Austausch definiert wurde. Der Simulation Request bzw. Modelling Request ist dabei der Ausgangspunkt für eine Absprache zwischen zwei Partnern, um eine Simulation zu beauftragen. Die im Simulation Request bzw. Modelling Request enthaltenen Informationen dienen als Input des CSP (siehe Abbildung 234). Das zugehörige neutrale Austauschformat leitet sich aus dem bestehenden STMD (Simulation Task Meta Data) ab.
Darauf basierend wurde in einer ähnlichen Konstellation wie für die Erprobung der Kooperationsprozesse ein industrieller Review der Requests durchgeführt. Dazu wurden zuerst partnerübergreifende Workshops mit den Industrievertretern und den Verantwortlichen und Partnern aus TP 3 durchgeführt. Daran anschließend wurden die in den gemeinsamen Workshops gewonnenen Erkenntnisse in separaten Workshop mit jeweils einem OEM und Tier detaillierter in Bezug auf den jeweiligen Anwendungsfall diskutiert, um den in der Realität auftretenden Zusammenarbeitsmodus nachzubilden und die Anforderungen an eine reale Umsetzung des Requests im Zusammenspiel eines Auftraggebers und Auftragnehmers durchzuspielen.

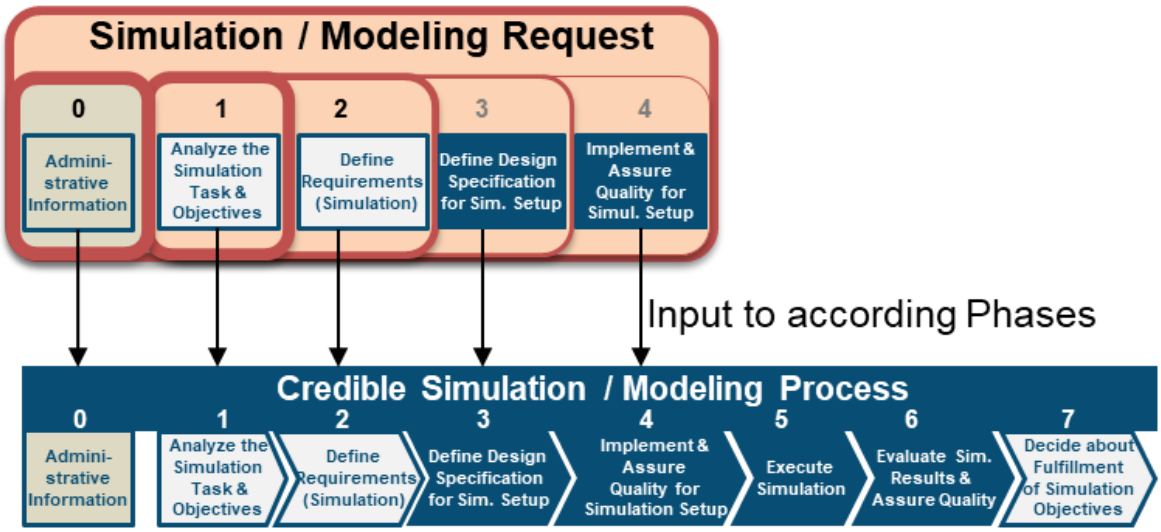


Abbildung 234: Zusammenhang zwischen Simulation / Modelling Request und Credible Simulation / Modelling Process

Bei den Reviews lag der Schwerpunkt hierbei auf den ersten drei Teilen des Simulation Request bzw. Modelling Request, die wie in Abbildung 234 dargestellt, den ersten Phasen des Credible Simulation und Credible Modelling Process zugeordnet werden können.
Einige der Haupterkenntnisse des Reviews und aus der Diskussion der Requests sind, die Identifikation von zwingend notwenigen Inhalten, die im Datenschema der Requests für den Austausch enthalten sein müssen, um ein vereinheitlichtes minimales Set an Metadaten zur Unterstützung des Kooperationsprozesses mit optionalen Bausteinen zu definieren. Darüber hinaus existiert Konsens darüber, dass bestimmte Elemente der Requests automatisch zu befüllen sein müssen und dies auch für den Nutzer explizit hervorgehoben werden sollte, um eine größtmögliche Vereinheitlichung sicherzustellen. Für die sinnvolle einheitliche Nutzung der Requests als standardisiertes Dokumentationsschema zur Unterstützung der Traceability im STMD-Format ist es darüber hinaus zwingend notwendig, die Inhalte bereits frühzeitig in der Agreementphase abzustimmen, klar zu definieren und die Ergebnisse auch in einem in der Agreementpahse erstellen Glossar festzuhalten und allen Beteiligten zur Verfügung zu stellen.
Zusätzlich konnten innerhalb der Partnerworkshops jedoch auch einige Einschränkungen identifiziert werden. Aktuell ist nicht davon auszugehen, dass ein OEM eine gesamte Simulation beauftragt, sondern nur eine Modellerstellung. Es ist aber nicht ausgeschlossen, dass sich dies in Zukunft ändern kann. Damit existiert an der Schnittstelle zwischen OEM und Lieferant primär nur das Modelling Request. Wobei das Modelling Request jedoch prinzipiell in beide Richtungen gestellt werden (z. B. wenn ein Lieferant eine Modellbeisteuerung vom OEM benötigt). Das Simulation Request kann jedoch firmenintern z. B. beim OEM als Leitfaden und Dokumentationsschema, welche Informationen der Auftragnehmer vom Auftraggeber benötigt, um eine Simulation durchführen zu können, verwendet werden und kann damit zur Unterstützung der Traceability beitragen. Zusätzlich besteht bei diesem Einsatz des Simulation Request weiterhin die Notwendigkeit auch bei der firmeninternen Verwendung eine klare Definition der Begriffe vorzusehen, da gegebenenfalls nicht ausschließlich Simulationsexperten beteiligt sind. Die Erkenntnis hinsichtlich Verwendung von Simulation und Modelling Request widerspricht damit jedoch nicht der Aussage, dass Simulation-Request und Modelling-Request die strukturierte und „standardisierte“ Zusammenarbeit zwischen Partnern erleichtern und unterstützen können.

Die gewonnenen Erkenntnisse aus den Workshops wurden neutralisiert aufbereitet an die Verantwortlichen im TP 3 übergeben und sind direkt in den aktuell vorliegenden Stand des Simulation Request und Modeling Request eingeflossen.

2.4.3.5.3 Traceability Demonstrator

Im Rahmen des SET Level Projektes wurden mit mehreren Industrievertretern Einzelworkshops zur Vorstellung, dem Review und Erprobung des Traceability Demonstrators TRACY durchgeführt. Dabei waren neben, den direkten Projektpartnern auch teilweise Kollegen aus dem Fachbereich der jeweiligen Häuser beteiligt. Für diese Workshops wurden häufig konkrete Anwendungsbeispiele aus dem Projekt verwendet, in denen die jeweiligen Partner direkt involviert waren, um einerseits den Einstieg für nicht direkt am Projekt beteiligte Personen zu erleichtern und andererseits auch den konkreten Mehrwert des Demonstrators zu zeigen.
Die Erkenntnisse aus den Workshops lieferten einen wichtigen Beitrag zur späteren industriellen Anwendbarkeit von TRACY und konnten zum größten Teil noch während der Laufzeit des Projektes im Demonstrator implementiert werden bzw. wurden in die Roadmap bis zum fertigen Produkt mit aufgenommen. Zu einigen der bereits implementierten Funktionalitäten zählen dabei beispielweise die grafische Darstellung der Abhängigkeiten der einzelnen Artefakte oder die im Demonstrator implementierten standardisierten Austauschformate.
Darüber hinaus wurde mit einem Industriepartner eine konkrete Evaluation des Traceability Demonstrators im Unternehmenskontext durchgeführt. Hierzu wurden reale Unternehmensprozesse zur Simulation von ADAS-Funktionen untersucht und identifiziert an welcher Stelle der Einsatz des Demonstrators einen Mehrwert für die digitale Durchgängigkeit der Prozesse leisten und die Traceability der Arbeitsabläufe unterstützen kann. Basierend darauf wurden Use Cases für den Einsatz im Rahmen der Simulation eines Bremssystems erarbeitet und eine Demonstrator-Instanz für die Nutzung im Unternehmen aufgesetzt. In das Vorgehen wurde ebenfalls der im Projekt entwickelte Credible Simulation Process, der als Template in TRACY hinlegt ist, mit einbezogen. Bereits während der Entwicklung konnten zusammen mit dem Industriepartner wertvolle Informationen und Anforderungen an den Demonstrator gesammelt werden. Diese betreffen zum Beispiel die Verlinkung von Artefakten, den Arbeitsfluss im Tool oder die Durchführung von Analysen. Der zusammen mit dem Industrievertreter aus SET Level final entwickelte PoC, wurde im Unternehmen im Rahmen einer Präsentation und Life-Demo einem größeren Publikum vorgestellt. Generell lässt sich aus dieser industriellen Erprobung ableiten, dass sich der toolgestützte Credible Simulation Process gut auf interne Unternehmensprozesse mappen lässt und die Arbeitsabläufe vereinfachen kann und dass der Einsatz von nichtproprietären Systemen und die Verwendung von neutralen Standards für die einfache Integration und Anwendung in Industrieunternehmen (z. B. durch die Anbindung an GitLab) einen deutlichen Vorteil mit sich bringt. Weiterhin hat sich gezeigt, dass die toolgestützte durch den Credible Simulation Process geleitete, strukturierte Vorgehensweise Querbeziehungen und Anhängigkeiten leichter identifizierbar macht und es zudem ermöglicht implizites Wissen im Unternehmen explizit und nachverfolgbar zu machen.
Während der Projektlaufzeit wurden kontinuierlich Anforderungen an TRACY gesammelt, die sich aus der konkreten Anwendung im Projekt und aus dem dazu gegebenen Feedback ergeben. Die gewonnenen Erkenntnisse sind in die kontinuierlich laufende Weiterentwicklung des Demonstrators eingeflossen.

Zusätzlich wurden neben der allen Beteiligten im Projekt zugänglichen Instanz des Traceability Demonstrators, TRACY, mit der die Workshops durchgeführt wurden, mehrere Cloud-Instanzen aufgesetzt, um den Industriepartnern eine Möglichkeit zu bieten, eigene Daten, die nicht für alle im Projekt sichtbar sein sollen in den Demonstrator einzupflegen und damit die industrielle Anwendbarkeit und Erprobbarkeit der Software zu erleichtern.

2.4.3.5.4 Simulation Use Cases

Zusätzlich zur Erprobung der Kooperationsprozesse wurde für die industrielle Erprobung der Ergebnisse des Projekts SET Level, der Aufbau des Simulation Use Case 2 und die dafür entwickelten Modelle, bei einem Industriepartner getestet. Dazu wurde ein Team aus verschiedenen Firmen gebildet, welche die jeweiligen Rollen innerhalb der Erprobung repräsentieren. Dieses Vorgehen hat das Ziel eine möglichst realitätsnahe Erprobung des Simulation Use Case durchzuführen, da davon auszugehen ist, dass auch im „realen“ industriellen Umfeld eine Aufgabenverteilung stattfinden wird. Abbildung 235 zeigt schematisch das Vorgehen innerhalb des Projektes zur Erprobung des Simulation Use Cases.

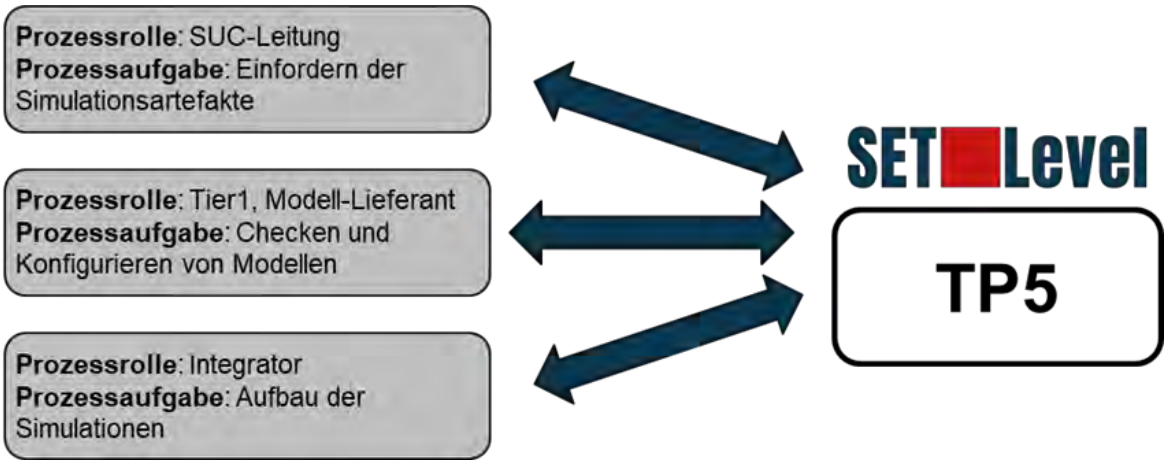


Abbildung 235: Schematische Darstellung der Simulation Use Case Erprobung

In der durchgeführten Erprobung konnte der Simulation Use Case 2 erfolgreich mittels drei verschiedener Architekturen umgesetzt werden: Innerhalb der Tools CarMaker (IPG) und ASM (dSPACE), monolithisch, sowie modular im Konzernumfeld.
Einige Erkenntnisse, die aus der Umsetzung des Simulation Use Cases 2 abgeleitet werden können und die mit großer Wahrscheinlichkeit generell auf den Aufbau und die Durchführung von Simulationen für das hochautomatisierte Fahren übertragbar sind, sind dass insbesondere für den Aufbau der Simulations-Infrastruktur und die Anpassung an existierende Simulationsframeworks das Vorhandensein von standardisierten Schnittstellen den Aufwand, der damit einhergeht, entscheidend verringern kann.
Darüber hinaus stellte auch die große Anzahl unterschiedlicher Stakeholder, Systeme, Lieferanten und Schnittstellen eine Herausforderung dar. Diese Situation stellt den industriellen Alltag jedoch gut dar. Durch die Anwendung in einem industriellen Umfeld wurde noch einmal deutlich, dass die praktische Umsetzung von Gesamtsimulationen für das hochautomatisierte Fahren eine große Herausforderung an die Modularisierung und Standardisierung der Simulation-Elemente darstellt.
Weiterhin wurde deutlich, dass die Integrationsfähigkeit der Modelle in die verwendeten Systeme, sowie deren Quantifizierbarkeit und Genauigkeit vor Beginn einer Simulationsaufgabe definiert sein sollte. Zudem sollten geeignete Prozesse vorhanden sein,

um die unterschiedlichen Handlungsfelder geordnet abarbeiten zu können. Eine Möglichkeit hierzu besteht darin nach dem Credible Simulation Process (CSP) und dem Credible Modelling Process (CMP) vorzugehen. Mit diesen kann ein konsistenter Aufbau und verkürzter Prozessdurchlauf erreicht werden.
Die Ergebnisse der Erprobung des Simulation Use Cases wurden seitens des Industriepartners in einem Projektbericht zusammengefasst und eine um die Konzerninterna „gekürzte“ Version, in der nur Inhalte enthalten sind, die an das Gesamtprojekt gehen können, auf GitLab abgelegt und damit dem Projekt zur Verfügung gestellt.
Zusätzlich zur Erprobung des Simulation Use Cases fand ebenfalls ein Review des Traceability Demonstrators TRACY innerhalb des gleichen Projektes statt. Die Ergebnisse der TRACY-Erprobung sind bereits während der Erprobung und auch in Form eines finalen Reviews an die Verantwortlichen im TP 4 übergeben worden. Hierbei wurde insbesondere die im Tool implementierte Projektplanung und die unterstützenden Funktionen zum Projektmanagement positiv hervorgehoben. Ein wesentliches Ergebnis des Reviews ist, dass das vorimplementierte Template des Credible Simulation Process dessen Anwendung und Befüllung deutlich erleichtert. Darüber hinaus wurden auch Verbesserungsvorschläge hinsichtlich Benutzerfreundlichkeit und Rollenmanagement erarbeitet.

2.4.3.6 Baselining im Projekt

Das Projekt SET Level hat sich für die Nutzung einer projekt- und partnerübergreifenden Datenablage entschieden. Im Rahmen des SET Level Projektes wurde dafür die Verwendung von GitLab forciert. Die Entscheidung wurde dabei aus Gründen der Datensicherheit zugunsten von git mit dem Server von GitLab getroffen. Dazu hat das DLR einen projekteigenen GitLab-Server aufgesetzt und zur Verfügung gestellt.
Im Zuge des Projektes wurden die Nutzungsmöglichkeiten von GitLab dahingehend ausgebaut und im Projekt etabliert, dass das Repository nicht nur als persistente Datenablage verwendet werden kann, sondern auch die partner- und teilprojektübergreifende Zusammenarbeit unterstützt.
Die in GitLab abgelegten Ergebnisse, Zwischenergebnisse und Arbeitsstände können von allen beteiligten Partnern eingesehen werden und sind diesen über die Downloadfunktion von GitLab zur Verfügung gestellt. Dies ermöglicht darüber hinaus, über die Nutzung von GitLab-Issues einen partnerübergreifenden Review von Ergebnissen und Arbeitsständen. Dieser kann dabei sowohl zentral (z. B. innerhalb einer Arbeitsgruppe) wie dezentral (durch Anmerkungen nicht direkt am Ergebnis beteiligter Partner) durchgeführt werden. Insbesondere im Zuge der Coronapandemie konnte hierdurch die Zusammenarbeit innerhalb des Projektes unterstützt werden. Beispielsweise wurde im Teilprojekt 3 ein teilprojektübergreifender Review des Credible Simulation Process über den Mechanismus der GitLab-Issues durchgeführt.
Durch die persistente Ablage von Zwischenständen, Projektergebnissen und die Verlinkung dieser Artefakte untereinander in git, konnte durch die Nutzung von GitLab zusätzlich das Baselining im Projekt unterstützt werden. Dies wurde speziell auch für die Entwicklungsstände der Simulation Use Cases angewendet. Auch hier bestand über die Verwendung von GitLab-Issues wieder die Möglichkeit, konkretes Feedback zu geben bzw. einzelne Reviews durchzuführen. Abbildung 236 zeigt exemplarisch die Möglichkeit, des Baselinings durch GitLab für den Simulation Use Case 1.

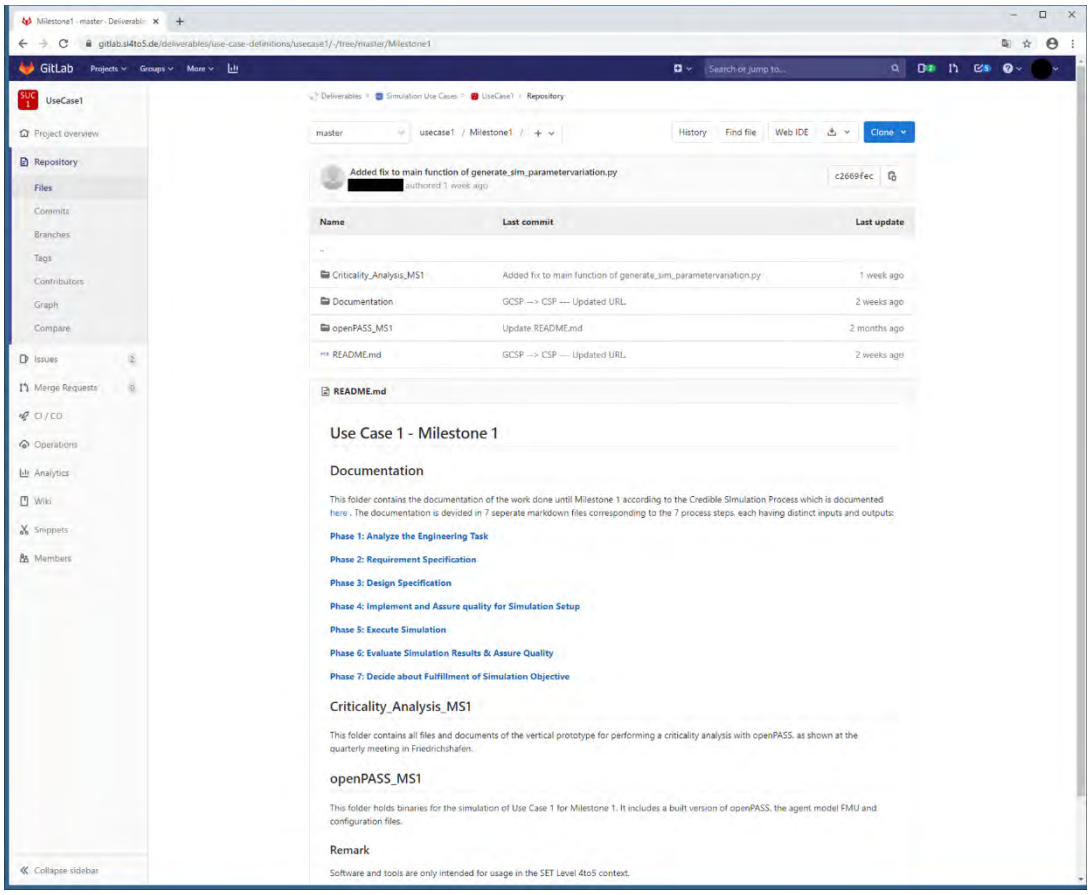


Abbildung 236: Baselining am Bespiel des Simulation Use Case 1 für den ersten Meilenstein

Zusätzlich wurde der im Projekt entwickelte Traceability Demonstrator TRACY verwendet, um die im GitLab dokumentierten Inhalte der Simulation Use Cases nach dem Credible Simulation Process toolgestützt konsistent abzubilden. Dazu wurden die dokumentierten Inhalte in den standardisierten Formaten zip und XML aus TRACY ausgeleitet und ebenfalls im GitLab zu den jeweiligen Use Cases abgelegt.

2.4.3.7 Austausch mit anderen Projekten

Im Rahmen des Austausches mit anderen (Förder)-Projekten und Institutionen wurde von BMW der Austausch von SET Level Themen, speziell Themen mit Standardisierungscharakter und Themen mit Referenzcharakter in gemeinsamen Workshops mit SIP-adus durchgeführt. Darüber hinaus fand zusammen mit BMW, Bosch, dem DLR und PROSTEP eine Zusammenarbeit mit der prostep ivip Projektgruppe SmartSE (Smart Systems Engineering) und den ASAM Gremien statt. Insbesondere im Rahmen des im TP 3 entwickelten Credible Simulation Prozess Framework und der Kooperationsprozesse fand dabei ein intensiver Austausch mit der prostep ivip Arbeitsgruppe SmartSE statt.
Nach Projektende wurden die im SET Level Projekt generierten Ergebnisse und Erkenntnisse im Themenkomplex Credible Simulation Prozess Framework an die SmartSE-Arbeitsgruppe übergeben, um die Arbeiten weiterzuführen und dabei neue Use Cases der Zusammenarbeit zu bearbeiten, die Ergebnisse zu versteigen und auch einen Personenkreis außerhalb der Automobilbranche zu adressieren, da in SmartSE auch Partner aus weiteren Branchen vertreten sind.
Zuvor wurde bereits zusammen mit Partnern aus dem TP 3, TP 4 und aus der SmartSE-Gruppe im Juni 2022 ein Workshop auf dem prostep ivip Symposium in Stuttgart durchgeführt. Einerseits war das Ziel die Vorstellung der Ergebnisse in Hinblick auf eine

prozessgestützte, kollaborative Zusammenarbeit unter Einbeziehung von dafür definierten Standards und des im Projekt entwickelten Traceability Demonstrators TRACY. Anderseits lieferte der Workshop und die dabei entstandenen Diskussionen, insbesondere mit Partnern nicht im Projekt SET Level mitwirkten, wertvolle Inputs für die finalen Arbeiten im Projekt und auch im Hinblick auf das Abschlussevent.
Darüber hinaus wurde unter Führung von BMW zusammen mit PROSTEP und dem DLR die Zusammenarbeit und der Austausch mit dem Schwesterprojekt VVMethoden über die Projektlaufzeit stetig vorangetrieben.

3 Literaturverzeichnis

ACOSAR. 2018. ACOSAR. [Online] 2018. [Zitat vom: 18.12.2020] http://www.acosar.eu/overview.php.

Arnold, Martin, Clauß, Christoph und Schierz, Tom. 2013. Error Analysis and Error Estimates forCo-simulation in FMI for Model Exchangeand Co-Simulation v2.0. *Archive of Mechanical Engineering.* 2013, S. 75-94.

BMWi. 2015. *Fachprogramm "Neue Fahrzeug- und Systemtechnologien".* 2015.

BMWi. 2015, geändert 2018 und 2021. *Richtlinie zur Förderung von Forschungs- und Entwicklungsprojekten im Rahmen des BMWi-Programms „Neue Fahrzeug- und Systemtechnologien“.* 2015, geändert 2018 und 2021.

Böde, Eckard, et al. 2017. *Design Paradigms for Multi-Layer Time Coherency in ADAS and Automated Driving (MULTIC).* s.l. : Forschungsvereinigung Automobiltechnik e.V. (FAT), 2017.

Bungartz, HJ., Zimmer, S., Buchholz, M., Pflüger, D. 2013. Mikroskopische Simulation von Straßenverkehr. *Modellbildung und Simulation.* s.l. : eXamen.press, 2013.

CIA. 2019. The World Factbook. 2019.

Clauß, Christoph, et al. 2012. *Master zur Simulatorkopplung via FMI.* Wolfenbüttel : ARGESIM-Verl., 2012. ASIM-Treffen der Fachgruppen "Simulation technischer Systeme" und "Grundlagen und Methoden in Modellbildung und Simulation". S. 57-70.

Eigner, M., Dickopf, T., and Apostolov, H. 2017. *The evolution of the V-model: From VDI 2206 to a system engineering based approach for developing cybertronic systems.* Seville, Spain : 14th IFIP International Conference on Product Lifecycle Management (PLM), 2017. S. 382–393.

Eigner, Martin, et al. 2014. System Lifecycle Management: Initial Approach for a Sustainable Product Development Process Based on Methods of Model Based Systems Engineering. [Hrsg.] Alain Bernard, Balan Gurumoorthy, and Abdelaziz Bouras Shuichi Fukuda. *Product Lifecycle Management for a Global Market.* s.l. : Springer, 2014, Bd. volume 442, S. 287–300.

FMI. 2020. Functional Mock-up Interface for Model Exchange and Co-Simulation. [Online] Dezember 2020. https://fmi-standard.org/.

Galbas, Roland, Neurohr, Christian und Rosenberger, Philipp. 2022. Application of SET Level Results in the VVMethods Project. *setlevel.de.* [Online] 12. 10 2022. [Zitat vom: 14. 02 2023.] https://setlevel.de/en/final-presentation/presentations/application-of-set-level-results-in-the-vvmethods-project.

Geimer, M., Krüger, T. und Linsel, P. 2006. Co-Simulation, gekoppelte Simulation oder Simulatorkopplung? Ein Versuch der Begriffsvereinheitlichung. *O+P Ölhydraulik und Pneumatik.* 2006, S. 572 - 576.

Gomes, Clàudio, et al. 2017. Co-simulation: State of the art. *arXiv.* Februar 2017.

Günther, Felix Christian. 2017. *Beitrag zur Co–Simulation in der Gesamtsystementwicklung des Kraftfahrzeugs.* s.l. : TU München, 2017. Doktorarbeit.

Hungar, Hardi. 2020. *A Concept of Scenario Space Exploration with Criticality Coverage Guarantees.* 2020.

Hungar, Hardi. 2021. *Use Cases for Simulation in the Development of Automated Driving Systems.* [Hrsg.] T., Steffen, B. Margaria. s.l. : Springer, Cham, 2021. Bd. 13036.

IEEE Standards Association. 2010. IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) — Framework and Rules. [Online] 2010. https://standards.ieee.org/standard/1516-2010.html.

IPG. 2019. Vehicle-in-the-Loop Methode. *https://ipg-automotive.com/de/unternehmen/news/vehicle-in-the-loop-methode-erlebt-an-unserem-neuen-demofahrzeug/.* [Online] 27. 07 2019. [Zitat vom: 09. 02 2023.] https://ipg-automotive.com/fileadmin/data/products_and_solutions/test_systems/vehicle-in-the-loop/product_sheets/VIL_Whitepaper_DE.pdf.

ISO. *ISO/AWI TS 5083 - Road vehicles – Safety for automated Driving systems – Design, verification and validation. .*

ISO. *Technical Report ISO/TR 4804:2020 - Road vehicles — Safety and cybersecurity for automated driving systems – Design, verification and validation.*

Jantsch, Axel. 2004. *Modeling Embedded Systems and SoCs: Concurrency and Time in Models of Computation.* s.l. : Morgan Kaufmann Series in Systems on Silicon, 2004.

Junghanns, Andreas und Blochwitz, Torsten. 2018., *10 Years of FMI.* Tokyo : s.n., 2018. Modelica Conference.

Kuefler, Axel, et al. 2017. Imitating Driver Behavior with Generative Adversarial Networks. *CoRR.* 2017.

Kullback, S. und Leibler, R.A. 1951. On information and sufficiency. *Annals of Mathematical Statistics Band 22.* März 1951, S. S. 79–86.

Linnhoff, Clemens, et al. 2021. *Towards Serious Perception Sensor Simulation for Safety Validation of Automated Driving - A Collaborative Method to Specify Sensor Models.* Indianapolis, IN, USA : IEEE International Intelligent Transportation Systems Conference (ITSC), 2021. Bd. 2021 IEEE International Intelligent Transportation Systems Conference (ITSC), S. 2688-2695. ITSC48978.2021.9564661.

Modelica Association. 2019. DISTRIBUTED CO-SIMULATION PROTOCOL (DCP). [Online] März 2019. https://dcp-standard.org/assets/specification/DCP_Specification_v1.0.pdf.

Neurohr, C., et al. 2021. Criticality Analysis for the Verification and Validation of Automated Vehicles. *IEEE Access.* 2021, vol. 9, S. 18016-18041.

SaFAD. 2019. https://www.press.bmwgroup.com. [Online] 2019. [Zitat vom: 09. 02 2023.] https://www.press.bmwgroup.com/deutschland/article/detail/T0298103DE/fuehrende-unternehmen-aus-dem-automobil-und-mobilitaetsbereich-veroeffentlichen-erstmals-rahmenbedingungen-zur-sicherheit-von-automatisierten-fahrzeugen-sae-level-3-und-4.

Schütt, B., et al. 2022. *An Application of Scenario Exploration to Find New Scenarios for the Development and Testing of Automated Driving Systems in Urban Scenarios.* s.l. : 8th International Conference on Vehicle Technology and Intelligent Transport Systems, 2022. 10.5220/0011064600003191.

SET Level. 2022. SET Level Public GitLab TSS. [Online] SET Level, 2022. [Zitat vom 28.11.2022] https://gitlab.setlevel.de/open/traffic-spaces-and-scenarios/tss.

SET Level. 2022. SET Level Website Abschlusspräsentation. [Online] SET Level, 2022. [Zitat vom: 28.11.2022] https://setlevel.de/en/final-presentation.

SET Level. 2022. SET Level Website Abschlusspräsentation UAP 1.2.1. [Online] SET Level, 2022. [Zitat vom: 28.11.2022] https://setlevel.de/en/final-presentation/booths/effective-scenario-definition.

SmartSE project group. 2021. *White paper "Simulation-based decision making and release".* Germany : prostep ivip Association, 2021.

Stanford Artificial Intelligence Laboratory et al. 2018. Robotic Operating System. [Online] 23.05.2018. https://www.ros.org.

Systems Engineering Vision Working Group. 2007. *INCOSE systems engineering vision 2020. Technical Product INCOSE-TP-2004-004-02.* San Diego, CA, USA : s.n., 2007.

TÜV Süd. 2013. Toolqualifizierung nach ISO 26262. *www.validas.de.* [Online] 2013. [Zitat vom: 10. 02 2023.] https://www.validas.de/TQS/2013/abstracts/Slides_Pappler.pdf.

VDAQMC. 2019. AIAG & VDA FMEA-Handbuch. *FMEA-Handbuch.* s.l. : VDAQMC, 2019, S. 122-123f.

Viehof, Michael. 2018. *Objektive Qualitätsbewertung von Fahrdynamiksimulationen durch statistische Validierung.* s.l. : TU Darmstadt, 2018.

VVMethoden. 2022. The VVM Glossary. *VVMethoden-Projekt Website Halbzeitpräsentation.* [Online] 2022. https://www.vvm-projekt.de/fileadmin/user_upload/Mid-Term/VVM_HZE_S1_P6_20220315_VVMGlossary.pdf.

Wachenfeld, W. und Winner, H. 2015. Die Freigabe des autonomen Fahrens. *Autonomes Fahren.* Berlin : Springer, 2015, S. 439-464.

Walden, D. D., Roedler, G. J., Forsberg, K. J., Hamelin, R. D., and Shortell, T. M. 2015. *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities.* 4th edition. s.l. : Wiley, Hoboken, NJ, USA, 2015.

Wikipedia. 2023. Gesetz von Conway. *Wikipedia.* [Online] 09.02.2023. [Zitat vom: 09.02 2023.] https://de.wikipedia.org/wiki/Gesetz_von_Conway.

Wikipedia. 2023. SMART (Projektmanagement). *Wikipedia.* [Online] 30.01.2023. [Zitat vom: 09..02.2023.] https://de.wikipedia.org/wiki/SMART_(Projektmanagement).

Wikipedia. 2023. Solver. *Wikipedia.* [Online] 30. Januar 2023. [Zitat vom: 09. 02 2023.] https://de.wikipedia.org/wiki/Solver.

Youssef Bouanan, Simon Gorecki, Judicaël Ribault, Gregory Zacharewicz, Nicolas Perry. 2018. *Including in HLA federation functional mockup units for supporting interoperability and reusability in Distributed Simulation.* Bordaux, France : s.n., 2018.

4 Abbildungsverzeichnis

5 Tabellenverzeichnis

Berichtsblatt

1. ISBN oder ISSN	2. Berichtsart (Schlussbericht oder Veröffentlichung) Schlussbericht

3. Titel

SET Level 4to5 – Simulationsbasiertes Entwickeln und Testen von Level 4 und 5 Systemen

4. Autor(en) [Name(n), Vorname(n)]	5. Abschlussdatum des Vorhabens 31.10.2022
Rude, Stefan; Köster, Frank; Hungar, Hardi; Sachsenweger, Heinz; Bou, Julien; Heinkel, Hans-Martin; Franke, Carsten; Platzer, Thomas; Fischer, Martin; Meyer, Arndt-Michael; Bühler, Christian; Plötzwich, Florian	6. Veröffentlichungsdatum 30.04.2023
	7. Form der Publikation
8. Durchführende Institution(en) (Name, Adresse) Bayerische Motoren Werke AG, Deutsches Zentrum für Luft- und Raumfahrt e. V., Opel Automobile GmbH, Audi AG, ADC Automotive Distance Control Systems GmbH, MAN Truck & Bus AG, Volkswagen AG, IPG Automotive GmbH, dSPACE GmbH, Robert Bosch GmbH, ZF Friedrichshafen AG, PROSTEP AG, ETAS GmbH, Ford-Werke GmbH, OFFIS – Institut für Informatik e. V.[1], RWTH Aachen, TU Braunschweig, TU Darmstadt, Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung e.V., FZI Forschungszentrum Informatik [1)]OFFIS ist während der Projektlaufzeit ausgeschieden, die von OFFIS geplanten Arbeiten wurden vom DLR übernommen	9. Ber. Nr. Durchführende Institution
	10. Förderkennzeichen
	11. Seitenzahl 315
12. Fördernde Institution (Name, Adresse) Bundesministerium für Wirtschaft und Klimaschutz (BMWK) 11019 Berlin	13. Literaturangaben 46
	14. Tabellen 4
	15. Abbildungen 236
16. Zusätzliche Angaben	
17. Vorgelegt bei (Titel, Ort, Datum)	

18. Kurzfassung

Im Projekt SET Level wurden

- Plattformen zur Anforderungsermittlung für automatisierte und vernetzte Straßenfahrzeuge im urbanen Raum und für das (entwicklungsbegleitende) Testen konzipiert, wobei jeweils auch Einzelergebnisse, die z. B. auf Komponentenebene erzielt wurden, im Hinblick auf eine Gesamtfahrzeugbeurteilung verwendbar sind, um die Homologation und Freigabe des Gesamtfahrzeugs später durch Simulationsanteile mittelbar zu unterstützen.
- Beiträge zur Standardisierung simulationsbasierter Entwicklungs-/Testwerkzeuge erarbeitet - u. a. zur/zum
  - Spezifikation von Szenarien und Anwendungsfällen
  - Spezifikation von Modellbeschreibungen und Modellen (u. a. sind hier Granularität und Integrationsfähigkeit zu nennen)
  - Repräsentation von Modellen und Parametrisierungen in Katalogen
  - Modellintegration und Handhabung von Integrationsproblemen
  - Management von Simulationsdaten
  - Konfigurationsmanagement
- Plattformen instanziiert, die zur Ergebnisdarstellung und -demonstration genutzt und insbesondere auch als Aufsatzpunkt für andere/folgende Projekte wie z. B. VVMethoden oder KI-Absicherung herangezogen werden können.

19. Schlagwörter
Simulation, Entwicklung, Testen, Verifikation, Validierung, Automatisiertes Fahren, Autonomes Fahren, automatisierte Fahrfunktion, Sensoren, Sensormodelle, System under Test, Functional Mockup Unit (FMU), Functional Mockup Interface (FMI), ASAM, OpenSCENARIO, OpenDRIVE, Open Simulation Interface (OSI)

20. Verlag	21. Preis

Document Control Sheet

1. ISBN or ISSN	2. type of document (e.g. report, publication) Final report

3. title

SET Level 4to5 – Simulation-based Engineering and Testing of Automated Driving

4. author(s) (family name, first name(s))	5. end of project 31.10.2022
Rude, Stefan; Köster, Frank; Hungar, Hardi; Sachsenweger, Heinz; Bou, Julien; Heinkel, Hans-Martin; Franke, Carsten; Platzer, Thomas; Fischer, Martin; Meyer, Arndt-Michael; Bühler, Christian; Plötzwich, Florian	6. publication date 30.04.2023
	7. form of publication
8. performing organization(s) (name, address) Bayerische Motoren Werke AG, Deutsches Zentrum für Luft- und Raumfahrt e. V., Opel Automobile GmbH, Audi AG, ADC Automotive Distance Control Systems GmbH, MAN Truck & Bus AG, Volkswagen AG, IPG Automotive GmbH, dSPACE GmbH, Robert Bosch GmbH, ZF Friedrichshafen AG, PROSTEP AG, ETAS GmbH, Ford-Werke GmbH, OFFIS – Institut für Informatik e. V.[1], RWTH Aachen, TU Braunschweig, TU Darmstadt, Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung e.V., FZI Forschungszentrum Informatik [1)] OFFIS left during the project duration, the work that was planned to be performed by OFFIS was taken over by DLR.	9. originator's report no.
	10. reference no.
	11. no. of pages 315
12. sponsoring agency (name, address) Federal Ministry for Economic Affairs and Climate Action 11019 Berlin	13. no. of references 46
	14. no. of tables 4
	15. no. of figures 236
16. supplementary notes	
17. presented at (title, place, date)	

18. abstract

In the SET Level project

- platforms were designed for the definition of requirements for automated and connected vehicles in urban areas and for (development-related) testing, where individual results from the component level can also be used on the overall vehicle level, in order to support homologation and release of the entire vehicle by simulation
- contributions to the standardization of simulation-based development/ test tools are developed - among others for the
  - specification of scenarios and use cases
  - specification of model descriptions and models (e.g. granularity and integration capability)
  - representation of models and parameterizations in catalogs
  - model integration and handling of integration problems
  - management of simulation data
  - configuration management
- platforms are instantiated, which can be used for result presentation and demonstration, and in particular as a starting point for other/ following projects such as VV methods or AI validation.

19. keywords
simulation, development, testing, Verification, Validation, Automated Driving, Autonomous Driving, Automated Driving Function, sensors, sensor models, System under Test, Functional Mockup Unit (FMU), Functional Mockup Interface (FMI), ASAM, OpenSCENARIO, OpenDRIVE, Open Simulation Interface (OSI)

20. publisher	21. price