Ansätze für eine modulare Absicherung hochautomatisierter Fahrzeuge

Am Fachbereich Maschinenbau an der
Technischen Universität Darmstadt
zur Erlangung des Grades eines
Doktor-Ingenieurs (Dr.-Ing.)
genehmigte

Dissertation

vorgelegt von

Björn Klamann M.Sc.
aus Limburg a.d. Lahn

Berichterstatter:	Prof. Dr.-Ing. Steven Peters
Mitberichterstatter:	Assoc.Prof. Dipl.-Ing. Dr.techn. Arno Eichberger

Tag der Einreichung:	05.12.2023
Tag der mündlichen Prüfung:	27.02.2024

Darmstadt 2024
D 17

Vorwort

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Fachgebiet Fahrzeugtechnik (FZD) der Technischen Universität Darmstadt. Im Rahmen des Forschungsprojekts UNICAR*agil* und AUTOtech.*agil* bekam ich die Chance an einem außergewöhnlichen Vorhaben mehrerer Universitäten mitzuarbeiten. Ich danke allen Projektmitarbeitenden für das große Engagement, die Unterstützung und die tolle Arbeitsatmosphäre. Ebenso danke ich allen Mitarbeitenden von FZD, die allesamt eine großartige Gemeinschaft und Zusammenarbeit bei FZD geschaffen haben. Behaltet dies bei und unterstützt euch besonders in den schwierigen Zeiten auf dem Weg zur Dissertation, wie ihr auch mich unterstützt habt.

Ein großer Dank geht an meinen Doktorvater, Prof. Dr.-Ing. Steven Peters, der mich auch in schwierigen Zeiten unter vollem Einsatz und mit viel Motivation zur Finalisierung der vorliegenden Arbeit geführt hat. Trotz des Neueinstiegs in das Thema hast du mich nicht nur bei der Struktur, sondern auch beim inhaltlichen Feinschliff intensiv unterstützen können.

Ich danke auch Herrn Prof. Dr. rer. nat. Hermann Winner, der mich während seiner Zeit bei FZD durch regelmäßige fachlich tiefgehende Rücksprachen intensiv beim Erkenntnisgewinn für die vorliegende Dissertation unterstützt hat. Insbesondere danke ich vielmals für das Feedback zur finalen Ausarbeitung der Dissertation auch nach deiner Pensionierung. Herrn Assoc.Prof. Dipl.-Ing. Dr.techn. Arno Eichberger danke ich vielmals für die Übernahme des Korreferats sowie die hilfreichen und freundlichen Fachgespräche.

Ebenso danke ich meinen Freunden und meiner Familie, die mich in anstrengenden Zeiten immer wieder unterstützt und motiviert haben. Mein größter Dank gebührt meiner Verlobten, Carlotta. Die Kraft und Unterstützung, die du mir immer wieder gegeben hast und immer wieder gibst sind unersetzlich. Ich freue mich sehr auf unsere gemeinsamen glänzenden Wege.

Inhaltsverzeichnis

Abkürzungen

Abkürzung	Beschreibung
A	Assumption (de.: Annahme)
Abk.	Abkürzung
ACC	Adaptive Cruise Control (de.: Adaptive Geschwindigkeitsregelung)
ADS	Automated Driving System
ASIL	Automotive Safety Integrity Level
ASOA	Automotive Service-Oriented Architecture
A-SPICE	Automotive SPICE
BMBF	Bundesministerium für Bildung und Forschung
BSSD	Behavior Semantic Scenery Description (de.: Verhaltenssemantische Szeneriebeschreibung)
C	Context (de.: Kontext)
CPU	Central Processing Unit (de.: Zentrale Recheneinheit)
de.	Deutsch
E/E-Systeme	Elektrische und elektronische Systeme
eng.	Englisch
EVT	Extreme Value Theory (de.: Extremwerttheorie)
FMEA	Failure Mode and Effect Analysis (de.: Fehlermöglichkeits- und Einflussanalyse)
FTR	Fahrdynamik- und Trajektorienregler
FZS	Fahrdynamikzustandsschätzung
G	Goal (de.: Ziel)
GSN	Goal Structuring Notation
HAF	Hochautomatisiertes Fahrzeug
HARA	Hazard Analysis and Risk Assessment (de: Gefährdungs- und Risikoanalyse)
HiL	Hardware-in-the-Loop
HSI	Hardware-Software-Interfaces
IDD	Interface Design Description
IDL	Interface Definition Language
IQ	Informationsqualität
J	Justification (de.: Legitimation)
MD	Manual Driving
MuT	Module under Test
ODD	Operational Design Domain
OEM	Original Equipment Manufacturer (de: Erstausrüster / Fahrzeughersteller)
OuT	Object under Test

Abkürzung	**Beschreibung**
PEGASUS	Project for the establishment of generally accepted quality criteria, tools and methods as well as scenarios and situations for the release of highly-automated driving functions
S	Strategy (de.: Strategie)
S^2I^2	Detaillierte semantische Schnittstellenbeschreibung (S^2I^2 steht für „Syntax, Semantik, Influencing Factors, Impacts“)
SiA	Sicheres Anhalten
SiL	Software-in-the-Loop
Sn	Solution (de.: Lösung)
SOTIF	Safety of the Intended Functionality
SPICE	Software Process Improvement and Capability Determination
STPA	System Theoretic Process Analysis (de.: Systemtheoretische Prozessanalyse)
SuT	System under Test
U	Uncertainty (de.: Ungewissheit)
UML	Unified Modeling Language
ViL	Vehicle-in-the-Loop
VTP	Verhaltens- und Trajektorienplaner
VVM	Verification and Validation Methods
XiL	X-in-the-Loop

Abbildungsverzeichnis

Tabellenverzeichnis

Kurzzusammenfassung

Mit der Einführung hochautomatisierter Fahrzeuge steigt der Freigabeaufwand gegenüber nicht-automatisierten Fahrzeugen erheblich. Bestehende Normen und Prozesse der Automobilindustrie konzentrieren sich bisher auf die Freigabe des Gesamtfahrzeugs. Die Freigabe von Modulen anstelle des Gesamtsystems ermöglicht dagegen die Wiederverwendung von Modulen in verschiedenen Fahrzeugmodellen. Darüber hinaus bewahrt eine entsprechende modulare Absicherung bei Änderungen einzelner Module die Freigabe nicht geänderter Module.

Die vorliegende Arbeit ermittelt neue Ansätze, die eine modulare Absicherung hochautomatisierter Fahrzeuge unterstützen. Hierzu wird eine Argumentationskette in einer Goal Structuring Notation (GSN) präsentiert. Dabei entwickelte Ziele und Lösungen ermöglichen die Absicherung einzelner Module in relativer Unabhängigkeit zu anderen Modulen. Der erste Pfad der GSN argumentiert, dass Module äquivalent zu einem System spezifizierbar sind. Hierbei müssen Abhängigkeiten möglichst exakt identifiziert und beschrieben werden. Die dazu beschriebenen Komplexitätseigenschaften ermöglichen die Reduktion der Ungewissheiten über die Vollständigkeit und die Beschreibung der Abhängigkeiten zwischen Modulen. Der zweite Pfad argumentiert die Vollständigkeit und Validität von Modultests. Die Ziele zur Modulspezifikation werden dabei weiter ergänzt, indem notwendige Informationen zur Identifikation von Modultests hergeleitet werden. In beiden Pfaden verbleiben Ungewissheiten aufgrund der Dekomposition von Informationen der Systemebene zur Modulebene. Daher identifiziert der dritte Pfad der GSN mögliche Irrtümer im Dekompositionsprozess. Die hieraus abgeleiteten Ziele vermeiden diese Irrtümer.

Auf Basis der entwickelten Ziele der Argumentationskette für eine modulare Absicherung werden drei sich ergänzende Lösungen vorgestellt. Zuerst wird gezeigt, dass zwischen den modularen Architektursichten semantische Äquivalenz erreicht werden muss. Aufgrund verbleibender Ungewissheiten bzgl. der Auswahl und Validität von Testumgebungen wird außerdem eine Methode vorgestellt, die Testumgebungen für Module argumentativ auf Basis dekomponierter Testziele definiert. Als dritte Lösung wird die detaillierte semantische Schnittstellenbeschreibung S^2I^2 vorgestellt. Diese ermöglicht anhand vorgegebener Attribute eine detaillierte Verhaltensbeschreibung an den Modulschnittstellen. Darüber hinaus werden Attribute zur Beschreibung der Einflussfaktoren und Auswirkungen innerhalb der Einsatzumgebung eines Moduls vorgestellt. Neben einem Anwendungsbeispiel für S^2I^2 wird anhand des im BMBF-Verbundforschungsprojekts UNICAR*agil* (Förderkennzeichen 16EMO0286) entwickelten hochautomatisierten modularen Fahrzeugs demonstriert, wie der Einsatz von S^2I^2 Fehlerzustände und Irrtümer bereits auf Modulebene vermeidet.

Abstract

Safety approval processes of current standards in the automotive industry consistently focus on the approval of the vehicle as a whole. Even with new approaches currently under research, the approval effort for highly automated vehicles increases significantly compared to non-automated vehicles. Approving modules rather than the overall system provides the opportunity to reuse the modules in different vehicle models. Such a modular safety approval additionally ensures the safety approval of other modules when only one is changed or updated. This thesis covers the derivation of new approaches that contribute to the application of a modular safety approval.

An argumentation chain based on the Goal Structuring Notation (GSN) is presented to argue modular safety. Goals and solutions developed in this process enable the possibility to approve the safety of modules in relative independence to other modules. The first path of the GSN argues that modules can be specified equivalently to a system. In this context, dependencies must be identified and described accurately. The complexity reduction goals developed for this purpose enable a reduction of uncertainties about the completeness and the description of dependencies between modules. The second path argues for the completeness and validity of module tests. The goals for module specification are further complemented by deriving necessary information for the identification of module tests. In both paths, uncertainties remain due to the decomposition of information from the system level to the module level. Therefore, the third path of the GSN identifies possible errors in the decomposition process. The consequently derived goals avoid these errors.

Based on the developed goals of the argumentation chain, three complementary solutions are presented. First, it is shown that semantic equivalence must be achieved between modular architectural views. Due to remaining uncertainties regarding the selection and validity of test environments, a method is presented that defines test environments for modules argumentatively based on decomposed test goals. As a third solution, the detailed semantic interface description S^2I^2 is presented. This enables a detailed behavior description at the module interfaces based on given attributes. Furthermore, attributes for the description of influencing factors and effects within the environment of a module are presented. In addition to an application example for S^2I^2, the highly automated modular vehicle developed in the research project UNICAR*agil* is used to demonstrate how the application of S^2I^2 avoids faults and errors already at module level.

1 Einleitung

Eines der Hauptargumente für die Einführung hochautomatisierter Fahrzeuge (HAF) in den allgemeinen Straßenverkehr ist der potentielle Sicherheitsgewinn durch den Verzicht auf fehlbare menschliche Fahrer. Während ungewiss ist, ob die Einführung tatsächlich die Anzahl tödlicher Verkehrsunfälle verringern kann, ist eine der größten Herausforderungen die Argumentation der Sicherheit hochautomatisierter Fahrzeuge. Die folgende Arbeit widmet sich daher übergeordnet der Entwicklung und Analyse neuer Methoden zur Absicherung hochautomatisierter Fahrzeuge. Im folgenden Kapitel wird daraus die Zielsetzung der Arbeit motiviert, Ansätze zu entwickeln, die es ermöglichen Module eines HAFs individuell abzusichern, ohne auf Integrations- und Systemtests angewiesen zu sein. Daraus wird anschließend die Methodik und der Aufbau der Arbeit erläutert.

1.1 Motivation & Zielsetzung

Die Absicherung hochautomatisierter Fahrzeuge (HAF) gilt als Schlüssel für deren Einführung in den Straßenverkehr. Wissenschaftliche Publikationen (z. B. Wachenfeld und Winner[1]) haben gezeigt, dass eine Absicherung mit bestehenden Methoden der Automobilindustrie wirtschaftlich nicht machbar ist. Die wesentlichen Ursachen hierfür sind die hohe Komplexität der Umgebung, in der das automatisierte Fahrzeug operiert (vgl. z. B. Lippert et al.[2]), und die Komplexität eines HAFs selbst[3]. Eine Vielzahl von Software- und Hardwarekomponenten im Fahrzeug, die Menge an Informationen aus der Umgebung und der Einsatz von Verfahren des maschinellen Lernens erhöhen die Komplexität gegenüber konventionellen Fahrzeugen. Neue Fahrzeuge besitzen je nach Quelle bis zu 100[4] oder 120[5] Steuergeräte, weshalb für Automobilhersteller die Zentralisierung von Steuergeräten in Recheneinheiten eine immer wichtigere Rolle einnimmt.[6] Bei Softwarekomponenten ist eine noch höhere Steigerung der Zahl an Codezeilen zu beobachten. Zwischen 2010 und 2016 stiegen diese bspw. von ca. 10 auf 150 Millionen Zeilen an Softwarecode.[7] Für HAF werden sogar über

[1] Wachenfeld, W.; Winner, H.: The Release of Autonomous Vehicles (2016), S. 437 ff.

[2] Lippert, M. et al.: Behavior-Semantic Scenery Description (2024).

[3] Kröger, W.; Ayoub, A.: Towards «type approval» of automated vehicles (2022).

[4] Burcicki, D.: The E/E architecture and the future of automotive innovation (2022).

[5] Bernhart, W.; Alexander, M.: Computer on wheels (2020).

[6] Bandur, V. et al.: Making the Case for Centralized Automotive E/E Architectures (2021).

[7] Burkacky, O. et al.: Rethinking car software and electronics architecture (2018).

300 Millionen Zeilen Softwarecode erwartet.[8] Die damit einhergehende Anzahl von Schnittstellen und Zuständen vergrößert den Parameterraum und damit den Testaufwand. Während des Entwicklungs- und Absicherungsprozesses steigt mit dem Grad der Integration auch die Anzahl der Parameter und Zustände. Bei Einsatz von maschinellen Lernverfahren existiert darüber hinaus bisher keine akzeptierte Methode zur Einflussanalyse nach Änderungen, sodass Tests nach Änderungen zu wiederholen sind.[9] Zusätzlich ist die Anzahl der möglichen Integrationsstufen nahezu unbegrenzt, da die Integrationsschritte infinitesimal klein sein können. Die Integration und Absicherung komplexer Systeme ist somit herausfordernd.[10] Zur Absicherung hochautomatisierter Fahrzeuge wird mit dem szenariobasierten Ansatz, der im Rahmen des vom Bundesministerium für Wirtschaft und Energie geförderten Forschungsprojekts PEGASUS[11] (eng. Abkürzung: Project for the establishment of generally accepted quality criteria, tools and methods as well as scenarios and situations for the release of highly-automated driving functions) eingehend untersucht wurde, das Gesamtsystem Fahrzeug für die Absicherung in allen Szenarien getestet. Die Szenarien sind jedoch so konzipiert, dass sie mögliche Unzulänglichkeiten testen, die in der Regel nur eine oder mehrere Komponenten betreffen. Selten werden dabei alle Komponenten adressiert.

Zur Beherrschung der Komplexität beinhalten die etablierten Entwicklungsprozesse eine Zerlegung des Systems, um darauffolgend die zerlegten Elemente zu entwickeln, zu analysieren und zu testen. Die fortgeschrittene Entwicklung von Software- und Hardware-in-the-Loop Testumgebungen zur Testung zerlegter Komponenten trägt wesentlich zur Qualitätssicherung und zur Reduktion notwendiger Nachbesserungen nach Systemtests bei. Hierdurch wird ein Großteil von Fehlerzuständen bereits in Komponententests aufgedeckt und korrigiert. Trotz der daraus resultierenden Verbesserungen der Systemzuverlässigkeit ist die endgültige Sicherheitsargumentation nach wie vor nur auf Systemebene zulässig.[12] Daher werden in dieser Arbeit Ansätze zur Umsetzung eines modularen Absicherungskonzepts untersucht. Bei diesem Konzept wird das System in einzeln prüfbare Elemente zerlegt, die im Folgenden als Module bezeichnet.[13]

Die *Hypothese* für die vorliegende Arbeit lautet, dass Module soweit abgesichert werden können, dass auf Integrations- und Systemtests verzichtet werden kann, während das aus Modulen zusammengesetzte System weiterhin das gewünschte Sicherheitsniveau erreicht. Die Zerlegung in Module führt zu einem kleineren und damit übersichtlicheren Parameterraum, sodass ein einzeln betrachtetes Modul weniger komplex ist. Darüber hinaus wird die Annahme getroffen, dass in einer Architektur mit durchgehend individuell abgesicherten

[8] Lazard; Roland Berger: Global Automotive Supplier Study 2018 (2017), S. 49.

[9] Koopman, P.: How safe is safe enough? (2022), S. 83.

[10] Borner, L.: Dissertation, Integrationstest (2010), S. 7.

[11] Mazzega, J. et al.: Pegasus Method (2019).

[12] ISO: ISO 26262 - Road vehicles – Functional safety (2018), Teil 3.

[13] Eine konkrete Definition des Begriffs „Modul“ folgt in Kapitel 2.1.

Modulen Änderungen an einem einzelnen Modul keine Auswirkungen auf die anderen Module haben. Lediglich das geänderte Modul muss erneut abgesichert werden. In diesem Fall können auch verschiedene Kombinationen an Modulen für verschiedene Anwendungsfälle kombiniert werden, ohne dass für jede einzelne Kombination eine Absicherung erforderlich ist. Die modulare Absicherung hätte damit auch potentielle Auswirkungen auf die bestehenden Marktstrukturen. Bei erfolgreicher Umsetzung einer modularen Absicherung könnten Zulieferer Teile der Verantwortung für die Absicherung von den OEMs (Original Equipment Manufacturers) übernehmen. Zulieferer könnten hierdurch ihre Marktmacht stärken.

Zur Erreichung einer möglichst hohen Repräsentativität erfolgt die Demonstration der entwickelten Ansätze in dieser Arbeit an einem möglichst modularen System. Hierzu kommt die Architektur der Prototypen des Forschungsprojekts UNICAR*agil* zum Einsatz. Die Prototypen stellen ein HAF mit SAE Level 4[14] dar. Die Hardwarearchitektur setzt auf eine Reduktion der Anzahl an Steuergeräten, sodass die drei Ebenen einer Fahrfunktion Perzeption, Planung und Aktion weitestgehend voneinander getrennt sind. Die Architektur folgt damit nicht einer konsequenten Zentralisierung von Steuergeräten, betreibt allerdings verschiedene Funktionen teilweise auf denselben Steuergeräten. Für die Software wird eine dienstbasierte Architektur aufgebaut, die Teilfunktionen als unabhängige Dienste darstellt, die gleichzeitig dynamisch rekonfigurierbar sind.[15] Weitere Details zur Architektur folgen in Kapitel 6.1.

1.2 Methodik und Aufbau der Arbeit

Die neuartige Zielsetzung führt zur in Kapitel 1.1 beschriebenen breit gefassten Hypothese, dass Module unabhängig voneinander abgesichert werden können und damit Integrations- und Systemtests reduzierbar sind. Dies erfordert eine in Kapitel 2 vorgestellte ebenso breite Erfassung bestehender Verfahren nach dem Stand der Technik. Hierbei werden einerseits Vorgehen zur Architekturgestaltung und Modularisierung einer Architektur analysiert, andererseits erfolgt eine Betrachtung gängiger Entwicklungsprozesse und deren Methoden zur Absicherung technischer Systeme. Abschließend werden bestehende Lücken im Stand der Technik identifiziert. Diese dienen im darauffolgenden Kapitel 3 der Erfassung des Stands der Wissenschaft zu neu entwickelten Methoden, die Beiträge zum Schließen bestehender Lücken leisten. In Anbetracht des gewünschten Einsatzes der modularen Absicherung für hochautomatisierte Fahrzeuge (HAF), wird ein breiter Überblick vielversprechender Ansätze aus diesem Forschungsbereich gegeben. Darüber hinaus werden alternative Sichtweisen und neue Architektursichten vorgestellt, die zur Beschreibung eines HAFs zum Einsatz kommen. Hierbei wird insbesondere der Einsatz neuer modularer Architekturen und deren Potential für eine modulare Absicherung präsentiert. Als essentielles Merkmal eines Moduls

[14] SAE International: SAE J3016 (2021), S. 28–32.

[15] Rumez, M. et al.: Automotive Service-Oriented Architectures (2020), S. 221853.

werden neuartige erweiterte Schnittstellenbeschreibungen ggü. dem Stand der Technik vorgestellt, die eine modulare Absicherung unterstützen können. Das in Kapitel 1.1 beschriebene Forschungsvorhaben der vorliegenden Arbeit wird daraufhin in bestehende und laufende Forschungsarbeiten eingeordnet. Hieraus werden offene Forschungsfragen hergeleitet, die zur Erreichung einer modularen Absicherung zu beantworten sind.

Zur Beantwortung der ersten Forschungsfrage werden bestehende Unzulänglichkeiten, die eine modulare Absicherung bisher verhindern, konkretisiert. Hierzu wird in Kapitel 4 eine Argumentationskette aufgebaut, die Ziele, Strategien und Lösungen definiert, um das übergeordnete Ziel einer modularen Absicherung zu unterstützen. Die Argumentationskette beschreitet dabei mehrere teilweise redundante Pfade, um eine möglichst hohe Vollständigkeit der damit aufgedeckten und bisher ungelösten Ziele zu erreichen. Hierbei werden Ungewissheiten in der Argumentation und im Dekompositionsprozess zur Entwicklung eines konkreten Moduls berücksichtigt. Die daraus identifizierten offenen Ziele betreffen vorwiegend die Spezifikation der Architektur und der Module. Abschließend werden Lösungen vorgestellt, die zur Schließung dieser Lücken beitragen.

Eine dieser Lücken ist die Beschreibung von Modulschnittstellen als eine von der Implementierung unabhängige Spezifikation zur Standardisierung von Modulen bzw. ihrer Schnittstellen. Daher wird in Kapitel 5 eine neue Beschreibungsform für Modulschnittstellen eingeführt. Diese sammelt Informationen zum Modulverhalten und den Abhängigkeiten zwischen Modulen bis hin zur Systemebene, die dieses Verhalten beeinflussen. Die Beschreibung dient entsprechend zur Vermeidung von Irrtümern in der Spezifikation und zur Aufdeckung von Fehlerzuständen durch Ableitung von Modultests.

Der beschriebene Nutzen der entwickelten detaillierten semantischen Schnittstellenbeschreibung wird in Kapitel 6 auf Basis der Entwicklungs- und Testergebnisse im Forschungsprojekt UNICAR*agil* demonstriert. Hierzu werden zuerst Modul- und Integrationstests beschrieben, die auf Basis des Stands der Technik und Wissenschaft abgeleitet sind. Im Detail werden dann die Testergebnisse analysiert, die Fehlerzustände bzw. Irrtümer aufgedeckt haben. In der Analyse wird jedem Fehlerzustand oder Irrtum mindestens ein Attribut der Schnittstellenbeschreibung zugeordnet. Daraufhin folgt eine Argumentation wie die Information eines oder mehrerer Attribute die Aufdeckung oder Vermeidung von Fehlerzuständen oder Irrtümern unterstützt.

Als Ergebnis der Arbeit liegt eine detaillierte Argumentationskette vor, die die Möglichkeit zur Absicherung von Modulen anstatt des Systems darlegt. Somit ist nachvollziehbar, welche Ziele und Ansätze notwendig sind, um eine modulare Absicherung so umzusetzen, dass sie einer Absicherung mit Integrations- und Systemtests äquivalent ist. Durch Entwicklung einer detaillierten semantischen Schnittstellenbeschreibung wird die Grundlage gelegt, die identifizierten offenen Ziele zu erfüllen. Mit der Berücksichtigung von Ungewissheiten in der Argumentationskette und im Dekompositionsprozess wird das Restrisiko darüber hinaus direkt offengelegt.

2 Grundlagen

Im ersten Abschnitt des Kapitels werden die in dieser Arbeit verwendeten Begrifflichkeiten vorgestellt. Aufgrund der Neuheit der Thematik wird insbesondere der Begriff des Moduls sowie zugehörige Begriffe rund um ein Modul im Rahmen der Absicherung neu definiert. Zu Teilen sind die Begriffe im englischen eindeutiger definiert und gebräuchlicher, weshalb diese den deutschen Begriffen zugeordnet werden.

Der zweite Abschnitt des Kapitels stellt den grundlegenden Zweck von Architekturen vor. Der Fokus liegt hierbei auf der Eigenschaft der Modularität einer Architektur. Es folgt eine Vorstellung der Verwendung von modularen Architekturen nach dem Stand der Technik und Prinzipien zur Gestaltung von Architekturen insbesondere zur Erreichung von modularen Eigenschaften.

Darauf folgt eine grundlegende Einführung von Schnittstellen und der Komplexität eines technischen Systems. Es schließt sich eine Übersicht zu Methoden zur Erreichung einer Sicherheitsfreigabe an und wie diese in Normen und Regularien in der Automobilindustrie vorgegeben werden. Abschließend erfolgt ein Überblick der dabei identifizierten Lücken im Stand der Technik für eine Absicherung hochautomatisierter Fahrzeuge allgemein und für die Erreichung einer modularen Absicherung.

2.1 Terminologie

In einer Sicherheitsargumentation können geringe Missverständnisse bereits schnell zu Personen- und Sachschäden führen. Das folgende Kapitel führt daher wichtige Begriffe ein, die in dieser Arbeit verwendet werden. Entsprechende Begriffe sind in Kursiv hervorgehoben. Die Terminologie dient dem Leser zum eindeutigen Verständnis der Begrifflichkeiten und Argumentationen. Die Begriffe werden zu großen Teilen der angegebenen Literatur entnommen. Allerdings nutzt auch die etablierte Literatur manche Begriffe nicht eindeutig, weshalb in solchen Fällen die Wahl der Terminologie begründet oder eine selbst eingeführte Terminologie erläutert wird.

Begriffe im Bereich der Absicherung

Der Begriff *Sicherheit (eng.: safety)* ist im Deutschen gegenüber dem Englischen weniger eindeutig. Im Englischen wird zwischen „Security“ und „Safety“ unterschieden. Die Security kann bei elektronischen Systemen im Deutschen genauer als Informationssicherheit (IT-Sicherheit) bezeichnet werden. Die Norm IEEE 1012[16] definiert die *Informationssicherheit* mit dem Schutz von Hardware und Software vor fremdem Zugriff, z. B. zum Abruf von Informationen oder zu deren nicht autorisierten Nutzung. Im Fokus der vorliegenden Arbeit

[16] IEEE: IEEE Std 1012 - Verification and Validation (2017), S. 25.

liegt dagegen der Teil der „Safety“ zur Erlangung einer Sicherheit vor Gefahren für Leib und Leben. Sicherheit meint hierbei nicht, dass in jeglichen Situationen die Wahrscheinlichkeit für einen Schaden Null ist. Die Norm ISO 26262 definiert *Sicherheit* stattdessen als die Abwesenheit eines unvertretbaren Risikos. Die Schwelle zur Unvertretbarkeit ist durch Entwickler und Gesellschaft zu ermitteln. Junietz[17] bezieht sich bspw. auf das makroskopische Risiko des Systems Straßenverkehr. Er beschreibt dazu Möglichkeiten und Maße, um einen Wert für ein zumutbares Risiko zur Einführung hochautomatisierter Fahrzeuge zu ermitteln. Zur Bewertung des Restrisikos eines HAFs fordert bspw. die EU-Regulierung 2022/1426[18] bereits die Definition der Abnahmekriterien anhand bestehender Risiken im Straßenverkehr durch manuell betriebene Fahrzeuge. Hierzu wird der Richtwert von 10^{-7} Todesopfern pro Betriebsstunde vorgeschlagen. Gleichzeitig wird darauf hingewiesen, dass ein Hersteller auch andere Richtwerte heranziehen kann, falls die Abwesenheit eines unangemessenen Sicherheitsrisikos nachgewiesen wird.

Die *funktionale Sicherheit* betrachtet genauer mögliche Fehlfunktionen von elektrischen und elektronischen Systemen (E/E-Systeme).[19] Die *Verhaltenssicherheit (eng.: behavioral safety)* bezieht darüber hinaus unabhängig von der technischen Umsetzung das von außen beobachtbare Verhalten mit ein. Die Abweichungen von einem gedachten oder gewünschten Sollverhalten werden dabei nicht zwingend nur hinsichtlich Fehlfunktionen, sondern auch hinsichtlich unvollständiger oder fehlerhafter und ggf. situationsspezifischer Spezifikationen analysiert.[20,21] Die Norm ISO/DIS 21448 bezeichnet dies als Sicherheit der vorgesehenen Funktion (eng.: Safety of the Intended Functionality, Abk.: SOTIF).[22] Die Argumentation der Sicherheit der vorgesehenen Funktion zur Absicherung hochautomatisierter Fahrzeuge wird als wesentliche Herausforderung gesehen, da die Spezifikation der möglichen Situationen und Szenarien, in denen sich ein Fahrzeug bewegt nicht vollständig vorhersehbar ist.[23]

Die *Absicherung*[24] eines technischen Systems wird in dieser Arbeit als übergeordneter Prozess zur Erbringung der Sicherheitsargumentation verstanden. Hierzu zählt neben den Tests zur Verifikation und Validierung auch der Entwicklungsprozess. Im Entwicklungsprozess können zur Absicherung Methoden angewendet werden, die die Erreichung bestimmter

[17] Junietz, P. M.: Dissertation, Microscopic and Macroscopic Risk Metrics (2019), S. 49–62.

[18] Europäische Kommission: Durchführungsverordnung (EU) 2022/1426 (2022), S. 14.

[19] ISO: ISO 26262 - Road vehicles – Functional safety (2018), Teil 1, S. 14.

[20] Klamann, B. et al.: Pass-/Fail-Criteria for Particular Tests (2019).

[21] WAYMO: Waymo Safety Report (2021).

[22] ISO: ISO/DIS 21448: Safety of the intended functionality (2022), S. 8.

[23] PEGASUS Projekt: PEGASUS Abschlussbericht (2020), S. 29–30.

[24] Der deutsche Begriff „Absicherung“ findet im englischen selten in direkter Übersetzung (eng.: safeguarding) Anwendung (vgl. Erläuterungen von Bagschik[27]). Stattdessen wird der Begriff Sicherheitsfreigabe (eng.: safety approval) als Ziel der Absicherung verwendet.

Ziele (z. B. die Vermeidung unvertretbaren Risikos) unterstützen. Hierzu liefern insbesondere Methoden zur Spezifikation oder zur Ermittlung von Anforderungen eine entscheidende Grundlage für die Absicherung.[25]

Die Literatur unterscheidet in der Entwicklung technischer Systeme zwischen Verifikation und Validierung eines technischen Systems oder einzelner Komponenten. Die *Verifikation* überprüft, ob die spezifizierten Anforderungen erreicht werden. Die *Validierung* prüft dagegen, ob der Zweck des eigentlichen Produkts oder bestimmte Ziele des Kunden erreicht werden. Die Validierung ist in diesem Fall auch eine Überprüfung der Spezifikation und Anforderungen.

Beide Begriffe werden aber auch in vergleichbarer Form zur Überprüfung von Simulationsmodellen angewendet. Diese dienen in der Absicherung zur frühen und kostengünstigen Prüfung des sich in Entwicklung befindlichen Systems. In der Verwendung der Begriffe wird das Modell in der vorigen Terminologie entsprechend zum technischen System. Die *Verifikation von Modellen* meint daher die Überprüfung, ob das implementierte Modell „formal richtig“ und dementsprechend die Anforderungen an das Modell erfüllt. Die *Validierung von Modellen* ist dagegen die Überprüfung, ob das Modell für seine gewünschte Anwendung geeignet ist. Auch hier ist daher das wesentliche Ziel zu prüfen, ob die Spezifikation und die Anforderungen korrekt sind.

Die Verifikation und Validierung kann also für das übergeordnete System erfolgen oder für ein Modell, das ein Abbild oder einen Teil des Systems oder der Systemumgebung darstellt. Daran lässt sich bereits der Einfluss verschiedener Hierarchieebenen auf die Verwendung der Begriffe erkennen. Dies ist übertragbar auf Teilsysteme, Module oder Komponenten eines Systems. Anforderungen zur Gestaltung und anschließenden Verifikation werden bspw. von den höheren Hierarchieebenen (z. B. von der System- auf die Modulebene) heruntergebrochen. Die Validierung eines Elements erfolgt dagegen durch den Einsatz in der gewünschten Anwendung oder wenigstens in einer dafür gestalteten repräsentativen Umgebung. Das bedeutet, Elemente aus niedrigeren Hierarchieebenen werden durch deren Einsatz in einer höheren validiert.

Auf Basis der bisherigen Begriffe finden auch Wortkombinationen im Bereich der Absicherung Verwendung. Die *Sicherheitsvalidierung (eng.: safety validation)* oder die *Sicherheitsargumentation (eng.: safety case)* stehen für den Prozess zur Erbringung von Argumenten, um die Abwesenheit unvertretbaren Risikos nachzuweisen.[26] Bagschik[27] betont, dass die Übersetzung des in der Norm ISO 26262 verwendeten englischen Begriffs „safety case“ in „Sicherheitsargumentation“ zutreffender als die verbreitete Übersetzung „Sicherheitsnachweis“ ist, da ein Nachweis für Sicherheit nicht vollständig erbracht werden kann. Mit Hilfe

[25] Henzel, M.: Dissertation, Generalisierbarkeit von maschinell gelernten Algorithmen (2019), S. 44–45.

[26] ISO: ISO 26262 - Road vehicles – Functional safety (2018), Teil 10, S. 12-13.

[27] Bagschik, G.: Dissertation, Systematischer Einsatz von Szenarien (2022), S. 15.

einer Sicherheitsargumentation kann eine *Sicherheitsfreigabe (eng.: safety approval)* erteilt bzw. erreicht werden. Die Literatur und der alltägliche Sprachgebrauch grenzen an dieser Stelle die Sicherheitsargumentation nicht klar von der Absicherung ab. In der vorliegenden Arbeit wird dagegen, wie von Weitzel[28] beschrieben, die Absicherung als ganzheitlicher Prozess bis zur Sicherheitsfreigabe betrachtet. Der Begriff Absicherung wird in dieser Arbeit daher bevorzugt verwendet.

Mit dem Ziel der Absicherung ein unvertretbares Risiko zu vermeiden, werden Gefährdungs- und Risikoanalysen durchgeführt. Das *Risiko* beschreibt die „Kombination aus der Wahrscheinlichkeit, mit der ein Schaden auftritt, und dem Ausmaß dieses Schadens“. Hierbei sind *Gefahren* konkrete Situationen, die unter Abwesenheit entsprechende Gegenmaßnahmen zu einem Schaden führen.[29] Das Risiko ist entsprechend größer als das *Grenzrisiko*, definiert als höchstes noch zumutbares Risiko.[30] Eine *Gefährdung* ist dagegen lediglich eine potentielle Schadensquelle[31], aber noch keine konkrete Situation.

Neben dem akzeptierten noch vertretbaren (Rest-)Risiko existiert bei Nutzung aller technischer Systeme ein inhärentes Risiko. Nach Maurer[32] ist das inhärente Risiko bei Einführung automatisierter Fahrzeuge nicht vollständig zu eliminieren. Das inhärente Risiko lässt sich lediglich reduzieren.

Da der Risikobegriff an eine Wahrscheinlichkeit und an einen Schaden gekoppelt ist, existiert der hiervon unabhängige englische Begriff „uncertainty“. Dessen übliche Übersetzung ins Deutsche „Unsicherheit“ impliziert allerdings, dass ein Zustand „unsicher“ ist und daher risikobehaftet ist. Daher wird stattdessen der deutsche Begriff Ungewissheit eingeführt. *Ungewissheit (eng.: uncertainty)* bedeutet, dass keine Gewissheit über den Zustand eines Systems oder über das vermeintlich bekannte Wissen besteht. Ungewissheit kann bspw. durch getroffene Annahmen oder fehlendes Grundlagenwissen entstehen. Die Definition entspricht der von Kiureghin and Ditlevsen für den englischen Begriff „epistemic uncertainty“: „…epistemische Ungewissheit ist eine Ungewissheit, bei der angenommen wird, dass sie durch mangelndes Wissen (oder Daten) verursacht wird.“[33] Unsicherheit wird in der vorliegenden Arbeit nur als Begriff verwendet, wenn eine Ungewissheit auch ein unvertretbares Risiko hervorruft.

Ungewissheiten in der Spezifikation oder der Implementierung können verschiedene Fehlerarten bzw. Anomalien hervorrufen. Der Begriff Anomalie bezeichnet "jede Abnormalität,

[28] Weitzel, A. et al.: Absicherungsstrategien für Fahrerassistenzsysteme (2014), S. 7.

[29] DIN: DIN VDE 31000-2 - Begriffe der Sicherheitstechnik (1987).

[30] Börcsök, J.: Funktionale Sicherheit (2011), S. 57.

[31] IEC: IEC 61508 (1997), Teil 4.

[32] Maurer, M.: Das inhärente Risiko autonomer Straßenfahrzeuge (2018).

[33] Eigene Übersetzung nach Kiureghian, A. D.; Ditlevsen, O.: Aleatory or epistemic? (2009), S. 106.

Irregularität, Inkonsistenz oder Abweichung von den Erwartungen".[34] Der Stand der Technik definiert diese Fehlerarten unterschiedlich. Für ein eindeutiges Verständnis der Begriffe, die jeweils auf die Spezifikations-, Implementierungs- und Betriebsphase eines Entwicklungsprozesses anwendbar sind, werden im Folgenden die englischen Definitionen für *fault*, *error* und *failure* nach Stolte et al.[35] und die deutschen Übersetzungen *Fehlerzustand*, *Irrtum* und *Versagen* nach Klamann und Winner[36] verwendet, sodass sich folgende Definitionen ergeben:

Fehlerzustand (eng.: fault): „Abnormaler interner Zustand, der dazu führen kann, dass ein Element ein unbeabsichtigtes Verhalten erzeugt“.[37]

Irrtum (eng.: error): „Abweichung von einem korrekten Wert oder externen Zustand, verursacht durch einen Fehlerzustand.“[38]

Versagen (eng.: failure): „Beendigung eines beabsichtigen Verhaltens eines Elements oder Gegenstands aufgrund der Manifestation eines Fehlerzustands.“[35]

Abbildung 2-1 der Norm ISO 26262 veranschaulicht die Verwendung der definierten Begriffe für zwei Ebenen einer Systemarchitektur, die Komponenten- und die Elementebene.

---

[34] IEEE Computer Society: IEEE Std 1044 - Software Anomalies (2010), S. iv.

[35] Stolte, T. et al.: Taxonomy to Unify Fault Tolerance Regimes (2021).

[36] Klamann, B.; Winner, H.: Analyse von Dekompositionsprozessen zur modularen Absicherung (2022).

[37] ISO: ISO 26262 - Road vehicles – Functional safety (2018), Teil 1, S. 11.

[38] Avizienis, A. et al.: Taxonomy of dependable and secure computing (2004), S. 4.

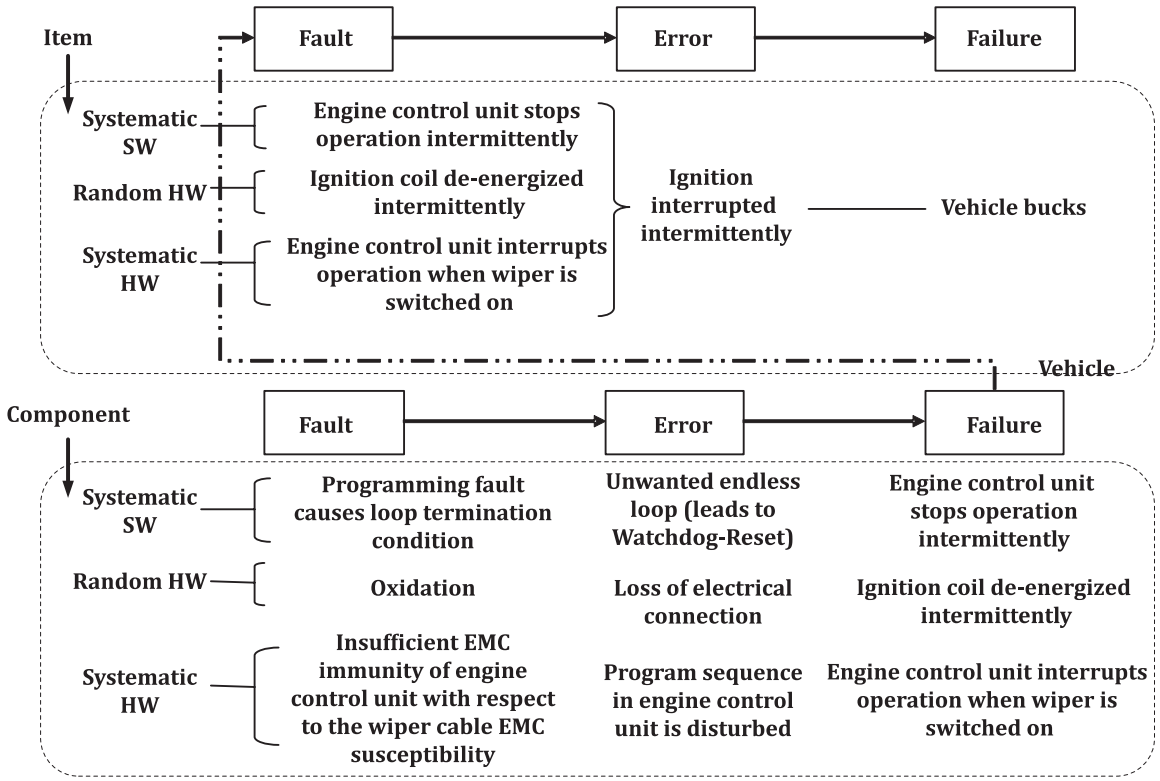


Abbildung 2-1: Beispiele für die Begriffe fault, error und failure nach ISO 26262[39] für zwei verschiedene Hierarchieebenen.

Abbildung 2-1 beschreibt jedoch nur die Betriebsphase, nicht aber die Berücksichtigung von Anomalien bei der Implementierung oder dem Spezifikationsprozess. Im Bereich der Softwareentwicklung wird der Begriff Irrtum (eng.: error) zum Ausdruck "einer menschlichen Handlung, die zu einem inkorrekten Ergebnis führt"[40] verwendet. Auf dieser Grundlage wird in dieser Arbeit mit Abbildung 2-2 eine zusätzliche Sichtweise eingeführt, die beispielhaft einen initialen Fehlerzustand in der Implementierungsphase berücksichtigt, der zu einem Versagen während des Betriebs führt.

[39] ISO: ISO 26262 - Road vehicles – Functional safety (2018), Teil 10, S. 6.

[40] Eigene Übersetzung nach IEEE Computer Society: IEEE Std 1044 - Software Anomalies (2010), S. 5.

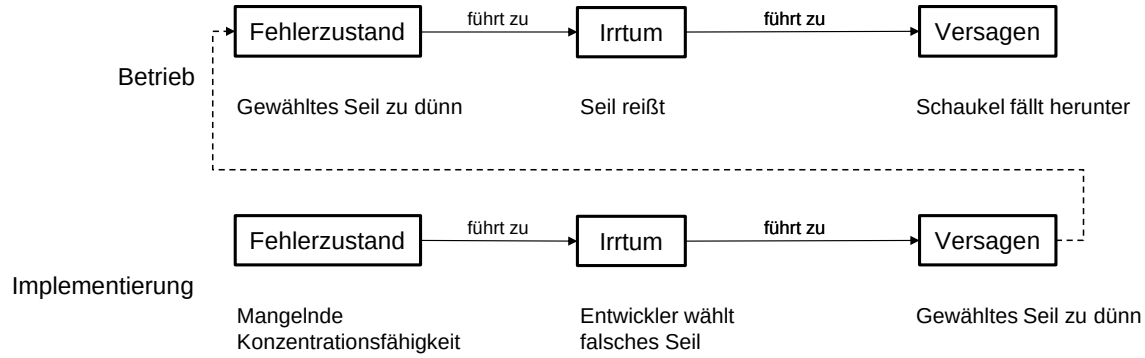


Abbildung 2-2: Beispiele für Fehlerzustand, Irrtum und Versagen in der Implementierungs- und in der Betriebsphase.

Abbildung 2-3 zeigt ein weiteres Beispiel für Anomalien in der Spezifikationsphase wie diese in der vorliegenden Arbeit definiert werden. Der initiale Fehlerzustand in der Spezifikation führt wiederum zu einem Fehlerzustand, Irrtum und anschließendem Versagen bei der Implementierung.

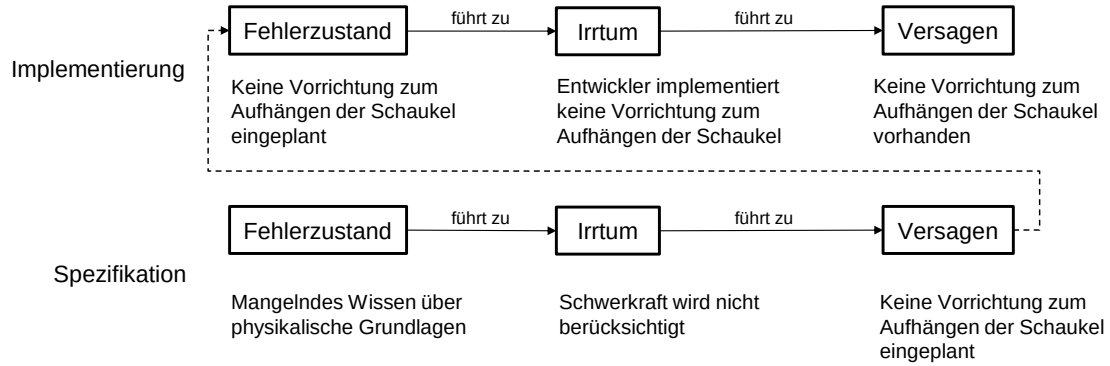


Abbildung 2-3: Beispiele für Fehlerzustand, Irrtum und Versagen in der Spezifikations- und in der Implementierungsphase.

Begriffe der Systemtechnik

Die vorliegende Arbeit beschäftigt sich mit Möglichkeiten zur Absicherung von Modulen, das heißt von Teilen eines Modulverbunds. Daher ist eine eindeutige Terminologie für diese verschiedenen Aufteilungen für das Verständnis der vorgebrachten Argumentationen wichtig.

Allgemein wird ein beliebiger, abstrakter oder konkreter Gegenstand als *Entität* bezeichnet. Die Entität ist daher ein Sammelbegriff für jede Art von geordneten Einheiten. Die Regeln zur Ordnung sind dabei erst für die jeweilige Anwendung festgelegt.[41]

Ein *System* ist üblicherweise die hauptsächlich betrachtete Entität in einer technischen Entwicklung. Ein System stellt eine gekapselte Funktionalität bereit bzw. erfüllt eine isolierte Menge an Anforderungen. Die Kapselung zu einem System kann üblicherweise als an einen

[41] IEC: International Electrotechnical Vocabulary (2015), IEV ref 741-01-18.

Endkunden verkäufliche Einheit betrachtet werden. Ein Kunde kann hierbei bspw. ein Nutzer des Systems Fahrzeug sein. Avizieniz et al. definieren ein System wie folgt:

„Ein System […] ist eine Entität, die mit anderen Entitäten, d.h. mit anderen Systemen, einschließlich Hardware, Software, Menschen und der physischen Welt, interagiert.“[42]

Die *Systemumgebung* umfasst alle Entitäten, die nicht innerhalb der definierten Systemgrenze liegen. Die Systemumgebung eines hochautomatisierten Fahrzeugs beinhaltet bspw. andere Verkehrsteilnehmer, die Straßeninfrastruktur oder die Umweltbedingungen. Scholtes et al.[43] präsentieren einen strukturierten Überblick an Elementen, die zur Beschreibung von Szenarien, die ein hochautomatisiertes Fahrzeug bewältigen muss, notwendig sind. Die *Szenarien* beschreiben dazu eine zeitliche Abfolge an *Szenen*, die durch verschiedene Parameter der Systemumgebung charakterisiert sind. Alle möglichen Elemente und Parameter der Systemumgebung, aus denen die Szenen und Szenarien erstellbar sind, werden z. B. in einem Dokument zur Beschreibung der *Operational Design Domain (ODD)* aufgenommen.[44]

Die Norm ISO 26262 unterscheidet vier verschiedene Hierarchieebenen.[45a] Folgende Beschreibungen sind nach ihrer Hierarchieebene von der höchsten zur niedrigsten angeordnet:

Fahrzeug: Die Norm ISO 26262 definiert lediglich die Fahrzeugfunktion, die ein Verhalten induziert, das vom Kunden beobachtbar ist. Das Fahrzeug kann demnach als das Gesamtsystem zusammengefasst werden, das von Kunden erwerbbar und nutzbar ist.

Item: Ein oder mehrere Systeme, das/die eine Funktion oder Teilfunktion auf der Fahrzeugebene darstellt und für die Sicherheitsanalyse (nach der Norm ISO 26262) betrachtet wird/werden.[46]

System: Zusammenstellung mehrerer Komponenten oder Subsysteme, die mindestens einen Sensor, einen Regler und einen Aktor miteinander verbinden.

Komponente: Technisch und logisch separierbares Element, das noch kein System darstellt und aus mehreren Hardwarebauteilen oder Softwareeinheiten besteht. Beispiele: Gehäuse, Mikrocontroller.

Hardwarebauteil (eng.: Hardware part): Teil einer Hardwarekomponente, das ggf. aus weiteren Hardwaresubbauteilen zusammengesetzt ist. Beispiele: CPU, Widerstand, Schraube.

Softwareeinheit (eng.: Software unit): Atomare Ebene einer Softwarekomponente, die noch eigenständig testbar ist. Dies können bspw. einzelne Softwareteile sein, die ausführbar sind, aber alleine noch keine nutzbare Funktion darstellen.

---

[42] Eigene Übersetzung nach Avizienis, A. et al.: Taxonomy of dependable and secure computing (2004), S. 3.

[43] Scholtes, M. et al.: 6-Layer Model for a Description of Urban Environment (2021).

[44] Czarnecki, K.: Operational Design Domain for Automated Driving Systems (2018).

[45] ISO: ISO 26262 - Road vehicles – Functional safety (2018), a: Teil 10, S. 4, b: Teil 1, S. 44.

[46] Eigene Übersetzung nach: ISO: ISO 26262 - Road vehicles – Functional safety (2018), Teil 1, S. 14–25.

Die Norm ISO 26262[45a] verlangt also, dass ein System mindestens einen Sensor, einen Regler und einen Aktor beinhaltet. Diese Definition scheitert jedoch bei Anwendung auf rein mechanische Systeme oder reine Computersysteme. Stolte et al.[47] erläutern, dass daher die Definition für ein System von Avizienis et al.[48] aufgrund ihrer breiteren Perspektive zu bevorzugen ist. Es sollte jedoch beachtet werden, dass Systeme in der Regel in kleinere Einheiten, wie Subsysteme oder Komponenten zerlegt werden können, wie in der Norm ISO 26262[45b] beschrieben. Die weiteren Kapitel der vorliegenden Arbeit werden zeigen, dass nicht nur das System, sondern auch dessen Umgebung und damit Ebenen oberhalb der Systemebene sicherheitsrelevant sein können. Teilweise wird daher von der *höchsten sicherheitsrelevanten Ebene* gesprochen, falls dies hervorgehoben werden soll. In der vorliegenden Arbeit wird davon ausgegangen, dass dies die Systemebene ist, sodass die Begriffe synonym verwendet werden.

Die Zerlegung einer übergeordneten Entität kann theoretisch auf beliebig vielen Ebenen, in Entitäten beliebiger Größe bis zu einer atomaren Ebene erfolgen. Sowohl Leveson[49] als auch Schäuffele et al.[50] beschränken sich für das System Engineering auf drei Hierarchieebenen, die System-, Subsystem- und Komponentenebene. Die Definition eines Systems gilt auch für ein Subsystem innerhalb eines Systems und ebenso für eine Komponente innerhalb eines Subsystems.

Ein Subsystem, das ein dekomponierter Teil eines Systems mit spezifischen modularen Eigenschaften ist, wird als Modul bezeichnet und ist für diese Arbeit wie folgt definiert:

„Ein *Modul* ist eine Gruppierung von Komponenten mit höherer Kopplung untereinander als zu Komponenten anderer Module in Bezug auf den Zweck der jeweiligen modularen Architektur. Ein Modul ist daraus folgend relativ unabhängig von anderen Modulen in Bezug auf den Zweck der jeweiligen modularen Architektur, besitzt aber wohldefinierte Schnittstellen zu diesen."

Der verfolgte Zweck in dieser Arbeit ist die individuelle Absicherung von Modulen. Die Modularchitektur für die modulare Absicherung hängt jedoch auch von anderen Zwecken ab (z. B. wirtschaftliche, organisatorische oder fertigungstechnische Zwecke). Daher kann es für verschiedene Zwecke desselben Systems unterschiedliche Modularchitekturen geben. Abbildung 2-4 veranschaulicht die Beziehung zwischen den verschiedenen hierarchischen

[47] Stolte, T. et al.: Taxonomy to Unify Fault Tolerance Regimes (2021).

[48] Avizienis, A. et al.: Taxonomy of dependable and secure computing (2004).

[49] Leveson, N.: Engineering a safer world (2012), S. 187.

[50] Schäuffele, J. et al.: Automotive software engineering (2016), S. 125.

Ebenen in Anlehnung an die Norm ISO 26262[51] und an die Definitionen von Steimle et al.[52] von Komponente, Hardware- und Softwareteil.

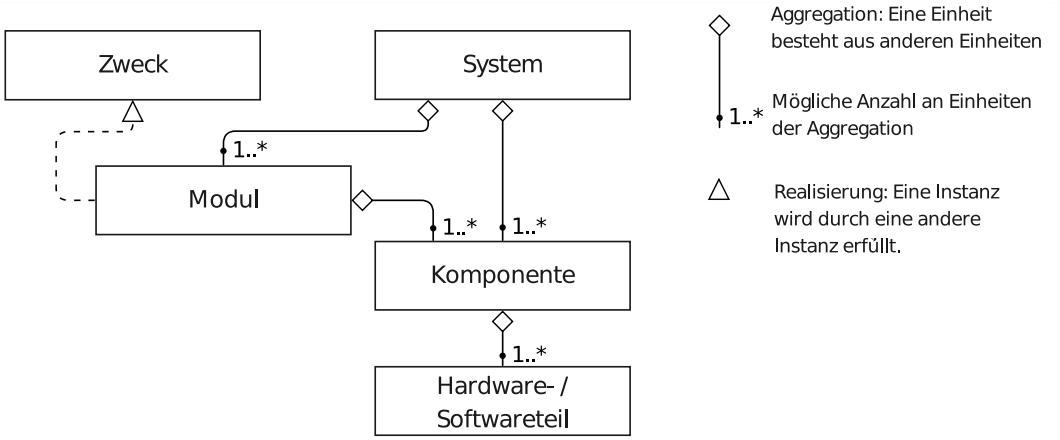


Abbildung 2-4: Betrachtete Hierarchieebenen und Zusammenhänge in Bezug auf Module, dargestellt als UML-Klassendiagramm.[53]

In einer Modularchitektur besteht die *Modulumgebung* aus Modulen oder der Systemumgebung. Entitäten, die ggü. anderen Modulen klein und ggf. eher als Komponente identifizierbar wären, werden ebenfalls als Modul definiert. Für ein konsequent modulares System ist dies dadurch gegeben, dass der Ausschluss einer Komponente von anderen Modulen automatisch bedeutet, dass die Komponente zu diesen Modulen relativ unabhängig ist. Dies ist gleichzeitig die Definition eines Moduls, sodass solch eine Komponente ein Modul darstellt.

In dienstbasierten Softwarearchitekturen haben sich zusätzlich Begriffe für Eingaben, Ausgaben sowie Empfänger und Sender von Daten etabliert. Empfänger einer Nachricht werden *Abonnenten (eng.: Subscriber)* genannt, die *Anforderungen (eng.: Requirements)* stellen, welche Informationen, in welcher Form benötigt werden. Sender einer Nachricht werden *Herausgeber (eng.: Publisher)* genannt, die *Garantien (eng.: Guarantees)* auf die gesendeten Informationen geben. Die Terminologie ergibt sich daraus, dass der Qualität der Nachrichten ein höherer Stellenwert beigepflichtet wird.[54]

Begriffe im Bereich des Testens

Die beschriebenen Hierarchieebenen werden auch zur Beschreibung einer Testebene genutzt. So beschreiben Komponenten-, Modul- und Systemtests jeweils einen Aufbau in denen jeweils Komponente, Modul oder System im Fokus der Testziele steht, d.h. das sogenannte *Object under Test (OuT)*.[55] Die Umgebung des OuT kann dabei simuliert oder

[51] ISO: ISO 26262 - Road vehicles – Functional safety (2018), Teil 10, S. 4.

[52] Steimle, M. et al.: Terminology for Scenario-Based Development and Test.

[53] Klamann, B.; Winner, H.: Analyse von Dekompositionsprozessen zur modularen Absicherung (2022).

[54] Kampmann, A. et al.: Dynamic Service-Oriented Software Architecture (2019).

[55] ISO: ISO/TR 4804 - Safety and cybersecurity for automated driving systems (2020), S. 6.

emuliert sein, sofern diese als relevant für den gewünschten Test eingestuft wird. Als Alternative zum allgemeinen OuT findet auch der Begriff *System under Test (SuT)* Anwendung.[56] In der vorliegenden Arbeit wird darüber hinaus der Begriff *Module under Test (MuT)* verwendet, um zu verdeutlichen, welches Modul innerhalb einer Architektur aus Modulen gerade betrachtet wird. Der Begriff MuT wird auch verwendet, wenn das Modul nicht getestet, aber bspw. analysiert oder spezifiziert wird.

Einen Zwischenschritt stellen *Integrationstests* dar. Integrationstests testen die Interaktion der Module untereinander nach dem Modultest. Laut IEEE handelt es sich bei Integrationstests um Tests, „bei denen Softwarekomponenten, Hardwarekomponenten oder beides kombiniert und getestet werden, um die Interaktion zwischen ihnen zu bewerten.“[57] In Integrationstests wird also mehr als ein Modul in realer Form getestet, während ein Teil der Umgebung weiterhin simuliert oder emuliert wird. Der *Systemtest* stellt die höchste Stufe der Integrationsteststufen dar. In diesem sind alle Module oder Komponenten des Systems in realer Form vorhanden. Die Umgebung kann dagegen weiterhin simuliert, emuliert oder real sein. In der vorliegenden Arbeit wird zur Vereinfachung bei Verwendung des Begriffs Integrationstest auch der Systemtest als höchste Stufe der Integrationstest miteingeschlossen. Auf Abweichungen von dieser Verwendung wird an der jeweiligen Textstelle hingewiesen.

Entgegen der vorigen Definition können reale Entitäten der Modulumgebung auch in *Modultests* für eine modulare Absicherung verwendet werden. Hierzu ist jedoch eine Argumentation darüber notwendig, dass die gegenseitigen Einflüsse ausreichend bekannt oder unabhängig voneinander sind. Die vorliegende Arbeit wird in den Kapiteln 4 und 5 erläutern, welche Methoden hierfür verwendet werden können.

Im realen System, wie in einer simulierten Testumgebung, erhalten Softwaremodule *Eingangsgrößen* bzw. *Eingaben* und erzeugen *Ausgangsgrößen* bzw. *Ausgaben*. Eingaben und Ausgaben können dabei gewollt, teilweise aber auch ungewollt sein und in verschiedener Form auftreten. Mögliche Formen werden in der Vorstellung des Stands der Technik von Schnittstellen erläutert (vgl. Kapitel 2.3). In einer Testumgebung spricht man darüber hinaus auch von einer gewünschten *Stimulation*, die ausreichend repräsentativ für die Eingaben im realen System in realer Umgebung ist. Die hierzu erzeugten Eingaben werden als *Stimuli* bezeichnet, die auf Basis von aufgezeichneten Daten oder mit Hilfe von Modellen erzeugt werden.

Die ausreichende Repräsentativität der Stimuli bzw. der gesamten Testumgebung wird als *Validität* bezeichnet. Die Validität einer Simulations- oder Testumgebung superpositioniert sich im Wesentlichen aus der *Modellvalidität* und der *Datenvalidität*.[58] Die Modellvalidität

[56] ASAM e.V.: ASAM Test Specification Study Group Report 2022 (2022), S. 69.

[57] Eigene Übersetzung nach IEEE: IEEE Std 1012 - Verification and Validation (2017), S. 24.

[58] Viehof, M.: Dissertation, Objektive Qualitätsbewertung durch statistische Validierung (2018), S. 12 ff.

meint hierbei die Validität des erzeugten Verhaltens der darzustellenden Objekte. Die Datenvalidität betrifft die Validität der Daten, die zur Modellparametrierung verwendet werden. Validität ist kein absolutes Maß, sondern immer abhängig vom gewünschten Zweck der Testumgebung. Zur grundlegenden Überprüfung, ob eine Softwarekomponente ausführbar ist, genügt bspw. eine übliche Entwicklungsumgebung mit der Möglichkeit einzelne Eingaben einzuspielen und Ausgaben zu erhalten. Zur Prüfung eines Reglers auf dessen Stabilität wird dagegen bspw. eine genauere Modellierung der Regelstrecke benötigt.

2.2 Architektur & Modularität

Modularität ist als spezifische Eigenschaft einer Architektur definiert. Daher ist zuerst der Blick auf den allgemeinen Zweck einer Architektur entscheidend, um den Zweck und die Eigenschaften einer modularen Architektur klar darzustellen. Per Definition beschreibt eine Systemarchitektur die Eigenschaften und Beziehungen ihrer einzelnen Bausteine.[59a] Die Bausteine werden durch Dekomposition des Systems definiert oder das System wird durch zuvor definierte Bausteine aus einer „Bottom-up“ Perspektive aggregiert. Der als Dekomposition bezeichnete Prozess ist allgemein als die Zerlegung oder Verfeinerung einer übergeordneten Entität in kleinere Entitäten definiert. Die kleineren Entitäten können wiederum zur größeren Entität aggregiert werden.[60a] Der Zweck dieser Dekomposition ist die Beherrschung der Komplexität eines Systems, da die zerlegten weniger komplexen Bausteine einzeln betrachtet und verstanden werden können.[60b] Bausteine können hierbei unterschiedlich hohe Granularität annehmen, d.h. es können darunter z. B. Subsysteme, Komponenten oder kleinere Bauteile verstanden werden.

Theoretisch nimmt die Komplexität der Bausteine mit jeder weiteren Zerlegung zu niedrigeren Granularitäten ab.[60b] Nach Vogel[59b] besitzen übergeordnete Hierarchieebenen jedoch emergente Eigenschaften, die erst durch das Zusammenspiel der zerlegten Bausteine entstehen und daher nicht vorhersehbar seien. Die gewünschten Eigenschaften eines technischen Systems werden zu Beginn festgelegt, um z. B. bestimmte Anwendungsfälle darzustellen. Hieraus lassen sich Eigenschaften der einzelnen Bausteine ableiten. Allerdings lässt sich mit der Annahme der Existenz emergenter Eigenschaften auf verschiedenen Hierarchieebenen nicht zuverlässig vorhersagen, ob die Eigenschaften der Bausteine auch die gewünschten Eigenschaften des Systems erzeugen.

Die Beschreibung und der Detailgrad der Systemarchitektur richtet sich nach deren Zweck. Aus diesem Grund werden verschiedene Sichtweisen von Systemarchitekturen unterschieden. Eine *Sichtweise* ist dabei der übergeordnete Blickwinkel, aus dem ein System oder eine

[59] Vogel, O.: Software-Architektur (2009), a: S. 58, b: S. 59.

[60] Schäuffele, J.; Zurawka, T.: Automotive Software Engineering (2016), a: S. 128–129, b: S. 21.

Problemstellung betrachtet wird. Der Sichtweise untergeordnet ist die Sicht bzw. *Architektursicht*, die ein System repräsentiert. Sichtweisen können daher generischer formuliert und ggf. wiederverwendet werden, während Architektursichten üblicherweise einem konkreten System zugeordnet sind und dieses darstellen.[61] Die Reduktion der Realität, die jeder Beschreibung einer Sichtweise und Architektursicht unterliegt, ist abhängig von der gewählten Sichtweise. Für einen Softwareentwickler sind beispielsweise Details über die Schnittstellen zwischen Komponenten entscheidend, um die eigene Funktion entwickeln zu können.[62a] Einem Projektmanager genügt dagegen eine gröbere Beschreibung der Schnittstellen, um Aufgaben nach Komponentenverantwortlichkeiten zu verteilen. Die Sichtweise des Projektmanagers ist daher in der Beschreibung der technischen Details der Schnittstellen reduziert gegenüber der Sichtweise des Softwareentwicklers. Alle Architektursichten beeinflussen sich gegenseitig bzw. sind voneinander abhängig. Die Stärke des Einflusses ist abhängig von den horizontalen Abhängigkeiten zwischen den Architektursichten.[63]

Etablierte Normen wie die Norm ISO 26262[64] beziehen sich vorwiegend auf drei Architektursichten. Die funktionale Sicht, die als erstes z. B. aus einer Beschreibung des Anwendungsfall abgeleitet wird, die Sicht der Hardwarekomponenten und die Sicht der Softwarekomponenten, die jeweils von der funktionalen Sicht abgeleitet werden.

Larses[65] betrachtet Einflussfaktoren auf modulare Architekturen und beschreibt dabei zwei Hauptkategorien an Architektursichten: Funktion und Implementierung. Die Implementierung umfasst ferner die drei Sichten Software & Kommunikation, Hardwaretopologie & E/E-Architektur und elektrische Signale.

Darüber hinaus lassen sich für verschiedene Zwecke noch weitere Architektursichten bestimmen. Dajsuren et al.[62b] definieren insgesamt sechs verschiedene Sichtweisen basierend auf Architekturen, die in der Automobilindustrie üblicherweise genutzt werden. Darunter zählen sie wie Larses eine funktionale, eine Software- und eine Hardwarearchitektur. Zusätzlich nennen Dajsuren et al.[63] die folgenden fünf Architektursichten.

Die *Use-Case-Sicht* (de: Anwendungsfallsicht) und die *Feature-Sicht* (de: Merkmalssicht) können als Varianten oder Vorstufen der *funktionalen Sicht* betrachtet werden. In der Use-Case-Sicht wird beschreiben, wie das System vom Nutzer verwendet wird. In der Feature-Sicht werden Funktionsumfänge eines Systems beschrieben, die bereits einen direkten Nutzen für den Kunden darstellen (z. B. eine adaptive Geschwindigkeitsregelung (ACC) oder

[61] The Open Group: Developing Architecture Views (2022).

[62] Dajsuren, Y.: Defining Architecture Framework for Automotive Systems (2019), a: S. 141, b: S. 161.

[63] Dajsuren, Y. et al.: Formalizing correspondence rules for automotive architecture views (2014).

[64] ISO: ISO 26262 - Road vehicles – Functional safety (2018), Teil 2, S. viii.

[65] Larses, O.: Factors influencing dependable modular architectures (2005), S. 5.

ein Navigationssystem). Darüber hinaus können aus beiden Sichten zusätzliche Testfälle abgeleitet werden, die die Nutzung des Systems bzw. enthaltener Teilsysteme direkt adressieren.

Die *Allokationssicht* liefert die Zuordnung zwischen der *Software*- und der *Hardwaresicht*. Für die Absicherung kann sie daher zur Spezifikation von Prüfständen oder zur Ableitung von Anforderungen aus den Zuordnungen (z. B. wie viel Speicherplatz wird von welcher Softwarekomponente auf welcher Hardwarekomponente benötigt) verwendet werden.

Die *Topologiesicht* ist eine spezifischere Sicht auf die Hardwarekomponenten. Sie beschreibt nach Dajsuren et al.[66] die Verbindungen zwischen Hardwarekomponenten in Bezug auf die Transporttechnologie (z. B. CAN oder Flexray) sowie die verwendete Hardware (z. B. Drähte oder Schalter).

Die *Timing-Sicht* ist insbesondere für dynamische Systeme, wie Fahrzeuge entscheidend, um z. B. durch Vorgabe von Stellgrößen zum richtigen Zeitpunkt die gewünschte Stellgröße auch erreichen zu können, das heißt, um Größen des dynamischen Systems zu regeln. Die Timing-Sicht gibt einen Überblick über die Zeit, die jeder Prozess benötigt, um z. B. die erforderliche Reaktion auf dynamisch geänderte Bedingungen zu erreichen.

Prinzipien und Taktiken zur Modularisierung

Nachdem sich die präsentierten Architektursichten nach dem Stand der Technik vorwiegend auf E/E-Komponenten und Softwarekomponenten beziehen, beginnt die folgende Darstellung von Modularität in der Mechanik. In diesem Bereich wurden erste Konzepte zur Gestaltung modularer Komponenten entwickelt und damit die Grundlagen der Modularität geschaffen.

Göpfert[67a] gibt einen Überblick darüber wie eine Modularisierung Einfluss auf Hierarchisierung, Entkopplung, Wiederverwendbarkeit, Austauschbarkeit, Erweiterbarkeit, Standardisierbarkeit, Kontrollierbarkeit, Stabilität und Kombinierbarkeit eines Systems hat. Wiederverwendbarkeit, Austauschbarkeit, Erweiterbarkeit, Standardisierbarkeit und Kombinierbarkeit bewirken vorwiegend wirtschaftliche Vorteile und besitzen das Potential durch gesammelte Erfahrungen die Qualität eines Systems zu steigern. Hierarchisierung, Entkopplung, Kontrollierbarkeit und Stabilität sind direkt mit technischen Eigenschaften einer modularen Architektur verbunden. Die detaillierten Eigenschaften und Vorteile werden im Folgenden näher erläutert.

Das Dekompositionsprinzip impliziert die Bildung einer Hierarchie durch die Strukturierung des Systems in mehrere Ebenen. Die *Hierarchisierung* schafft eine klare Darstellung der Zusammensetzung des Produkts und erleichtert so den Entwicklern das Verständnis des Produkts. Darüber hinaus kann der Produktionsprozess verkürzt werden, da die Herstellung und

[66] Dajsuren, Y.: Defining Architecture Framework for Automotive Systems (2019), S. 165.

[67] Göpfert, J.: Modulare Produktentwicklung (2009), a: p. 121, b: pp. 117-121.

Montage in verschiedenen parallel laufenden Entwicklungs- und Produktionsprozessen auf der Grundlage dieser Hierarchie erfolgen kann.[68]

Die *Entkopplung* der Module verringert die Abhängigkeit zwischen den Modulen. Hieraus kann eine Reduzierung der Anzahl an Schnittstellen erwartet werden, nicht aber unbedingt eine Reduktion der Komplexität dieser Schnittstellen. Die geringere Anzahl und klarere Definition von Schnittstellen begünstigen weiter die Parallelisierung der Entwicklungs- und Produktionsprozesse. Für mechanische Systeme begünstigt die Entkopplung auch die Montage und Demontage des Produkts. Dies ist insbesondere dann von Vorteil, wenn ein bestimmtes Modul repariert oder ausgetauscht werden muss. Die Entkopplung ermöglicht außerdem die Trennung des Lebenszyklus der Module vom Lebenszyklus des Fahrzeugs, was Zulieferern bereits die Entwicklung eigener Teilsystem ohne spezifische Vorgaben der Systemhersteller ermöglicht. Dies ermöglicht erst die Austauschbarkeit von Modulen. Dagegen erfordert die Entkopplung eine aufwendigere Spezifikation der Schnittstellen, um das Zusammenspiel und gleichzeitig die Lösbarkeit der Module zu gewährleisten.

Neben dem Aufwand für die Entwicklung und dem Management der Modularität bringen Modularchitekturen auch Nachteile bei der technischen Gestaltung mit sich. Im Vergleich zu einer integralen Architektur können modulare Architekturen zu einer komplexeren Struktur führen, da jede (Teil-)Funktion ein eigenes entkoppeltes Modul erfordern kann. Die Entkopplung kann dabei mit komplexeren Schnittstellen einhergehen, um eine ausreichende Unabhängigkeit zwischen den Modulen zu erreichen. Komplexere Schnittstellen erfordern dabei mehr Aufwand in der Spezifikation, mehr Konstruktions- oder Implementierungsaufwand oder auch mehr Bauraum. Weitere wirtschaftliche Nachteile von modularen Architekturen werden u. a. von Göpfert[67b] beschrieben.

Als grundlegende Prinzipien zur Erreichung modularer Eigenschaften in einer Architektur nennt Göpfert drei Prinzipien. Das *Dekompositionsprinzip* besagt, dass die Detaillierung des Systems oder der Module durch Zerlegung bedarfsgerecht fortgeführt werden sollte. Eine Zerlegung kann nach Interpretation des Autors derweil bedarfsgerecht sein, solange ein bestimmter Nutzen davon erwartet wird. Das *Unabhängigkeitsprinzip* hat das Ziel möglichst unabhängige Module zu gestalten. Hierzu sollen Anzahl und Intensität der Beziehungen möglichst gering sein. Das *Integritätsprinzip* dient als Korrektiv des Unabhängigkeitsprinzips. Module sollten nach dem Integritätsprinzip immer auch das Zusammenwirken der Module im Sinne des Systems berücksichtigen.[69]

Ähnliche Prinzipien existieren in der Softwaretechnik. Durch Auftrennung eines Problems in unabhängig voneinander behandelbare Teilaspekte wird das *Seperation of Concerns* verfolgt. Informationen über die inneren Prozesse, Zwischenergebnisse oder über die Implementierung eines Moduls sollen unter dem Prinzip des *Information Hiding* verborgen bleiben. Darüber hinaus soll die Modularität durch Abstraktion und die damit einhergehende

[68] Schuh, G.; Riesener, M.: Produktkomplexität managen (2018), S. 95.

[69] Göpfert, J.: Modulare Produktentwicklung (2009), S. 54–55.

Entwicklung von Modellen des Moduls und seiner Schnittstellen unterstützt werden. Nach Vogel impliziert die damit erreichbare Modularität auch die Prinzipien der losen Kopplung und der hohen Kohäsion.[70a] Nach dem Prinzip der *losen Kopplung* sollen keine internen Daten abrufbar und die Anzahl der Schnittstellen möglich gering sein.[71] Dies führt in der Regel zu einer *hohen Kohäsion*, bei der Module so gekapselt sind, dass dessen relevante Eigenschaften ohne Kenntnis anderer Module verstanden und geändert werden können.[70b]

Bass et al.[72] nennen in Ergänzung zu den Prinzipien verschiedene Entwicklungstaktiken, die insbesondere die Änderbarkeit und die Testbarkeit und daraus folgend wiederum die Modularität erhöhen. Dabei soll die *Modulgröße möglichst reduziert* werden, sodass die Anzahl an Funktionen geringer ist und damit der Aufwand für Änderungen und damit verbundenen Tests sinkt. Gleichen Effekt erzeugt auch die *Erhöhung der semantischen Kohäsion* durch die Teile eines Moduls, die verschiedene Zwecke haben, in andere oder neue Modulen ausgelagert werden. Zur *Reduktion der Kopplung* zwischen Modulen sind als neue Taktiken die Refaktorierung und die Abstraktion gemeinsamer Dienste hervorzuheben. *Refaktorierung* beschreibt eine Umstrukturierung des Softwarecodes zur Erhöhung der Verständlichkeit, Übersichtlichkeit oder Wiederverwendbarkeit, ohne dabei das beobachtbare Verhalten der Software zu verändern.[73] Zur Optimierung der Architektur werden insbesondere ähnliche oder redundante Softwareteile identifiziert und ausgelagert. Bei der *Abstraktion gemeinsamer Dienste* werden insbesondere ähnliche Softwareteile identifiziert, die durch eine generalisierte Version ersetzt und ausgelagert werden kann. Ähnliches Vorgehen auf tieferer Ebene verfolgt auch die *verzögerte Festlegung (von Parametern)*. Hierbei werden mathematische Funktionen und Parameter im Quelltext so lange wie möglich variabel gehalten, sodass Änderungen und Erweiterungen schnell umgesetzt werden können, ohne z. B. alle Parameterwerte manuell neu setzen zu müssen.

Konzepte zur Nutzung modularer Architekturen

Die Austauschbarkeit von Modulen bietet die Chance verschiedene Systemvarianten mit den gleichen Modulen zu erstellen. Dies kann sogar bis zur Verwendung derselben Kombination an Modulen für völlig unterschiedliche Anwendungsfälle gehen. Nach Bender und Gericke[74] gibt es drei verschiedene Strategien, die den Lösungsraum für die Verwendung von Modulen oder den Entwurf von modularen Architekturen aufspannen. Zuerst können Basismodule für verschiedene Produkte wiederverwendet werden. Module können außerdem entwickelt und verwendet werden, um verschiedene Produkte oder Anwendungsfälle durch verschiedene

[70] Vogel, O.: Software-Architektur (2009), a: S. 145–148, b: S. 141-145.

[71] Lotz, F. G.: Dissertation, Referenzarchitektur für automatisierte Fahrzeugführung (2017), S. 38.

[72] Bass, L. et al.: Software architecture in practice (2013), S. 141–145.

[73] Fowler, M.: Refactoring (2019), S. 45.

[74] Bender, B.; Gericke, K.: Pahl/Beitz Konstruktionslehre (2021), S. 353–355.

Kombinationen eines Moduls zu erreichen. Modulschnittstellen erfordern daher eine Standardisierung und oft Schnittstellen, die nur in bestimmten Modulkombinationen verwendet werden können. Zusätzlich ist auch die Verwendung einer Plattform als Basis mit standardisierten Schnittstellen möglich, an die verschiedene Module koppelbar sind. Die Plattform wird für eine Produktfamilie verwendet, die durch die hinzugefügten Module variiert.[74] Die Plattform umfasst entsprechend Komponenten eines Systems, die unverändert bleiben können, ohne die gewünschte Produktvielfalt wesentlich zu beeinträchtigen.

Lipsmeier[75] zeichnet einen grundsätzlicheren Blick auf mögliche Produktstrategien und schlägt vier Stufen der Modularisierung vor, wie sie in Abbildung 2-5 dargestellt sind. Angefangen bei der Plattformstrategie bis hin zu Modulen, die offen miteinander kombiniert werden können. Die Wahl der Modularisierungsstufe ist dabei abhängig von den Marktanforderungen bzw. der Produktstrategie. Soll bspw. eine hohe Anzahl verschiedener Varianten angeboten werden und sollen Module flexibel austauschbar sein, wird eher eine offene Modularisierung gewählt. Haben diese Produktstrategien weniger Priorität kann dagegen eine integralere Architektur funktionale Vorteile bieten oder geringeren Bauraum erfordern, da aufwendige Schnittstellen zwischen Modulen ggf. entfallen.

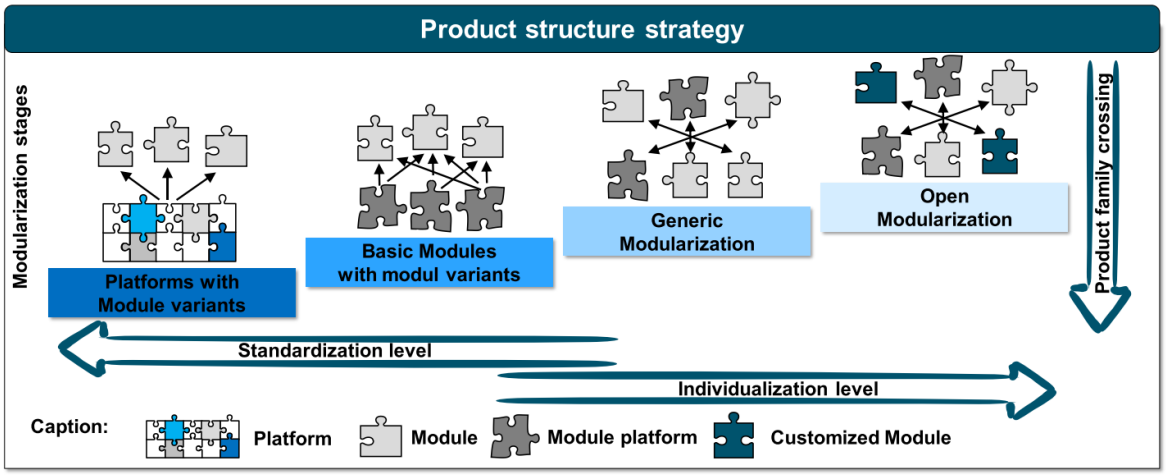


Abbildung 2-5: Mögliche Produktstrategien bei Nutzung einer modularen Architektur.[75]

Baldwin und Clark[76] unterscheiden zwischen "Modularität in der Konstruktion", "Modularität in der Produktion" und "Modularität bei der Nutzung". Die dargestellte Vielfalt an Architektursichten zu Beginn dieses Kapitels kann daher zur Erreichung von Modularität und die dafür notwendigen Sichten sogar noch höher sein. Module sind so definiert, dass sie relativ unabhängig von anderen Modulen innerhalb derselben Architektur sind, sodass Informationen innerhalb einer Architektur eindeutig zugeordnet sein müssen. Diese Unabhängigkeit definiert der Stand der Technik jedoch nicht über verschiedene Architekturen hinweg. Teilweise ergibt sich dies lediglich unbeabsichtigt, wenn verschiedene modulare

[75] Lipsmeier, A. et al.: Mechatronic Modularization of Intelligent Technical Systems (2018).

[76] Baldwin, C. Y.; Clark, K. B.: Managing in an age of modularity (2008).

Architektursichten ähnliche Strukturen aufweisen, da sie auf der gleichen Basis (z. B. der funktionalen Architektur) aufbauen.

Testbarkeit und Testumgebungen

Tests auf niedrigeren Hierarchieebenen erfordern eine für die Modultests geeignete Testumgebung. In der Literatur werden Anforderungen und Prinzipien beschrieben, um Testbarkeit einer Komponente sicherzustellen. Jungmayr[77] stellt u. a. Testbarkeitsanforderungen aus Sicht verschiedener Stakeholder vor. Für einen Testingenieur nennt er insgesamt zehn Anforderungen, die zur Verringerung des Testaufwands erfüllt werden sollten. Im Rahmen einer Absicherung einzelner Module sind dabei insbesondere die Kontrollierbarkeit, Beobachtbarkeit, Isolierbarkeit und Robustheit zu nennen. Die weitere Ableitung dieser Testbarkeitsanforderungen an Module und die modulare Architektur ergeben nach Bass et al.[78] identische Anforderungen bzw. Prinzipien wie zur Gestaltung modularer Architekturen. Als zusätzliche Herausforderung für Testbarkeit ergibt sich jedoch die Erreichung von Validität einer Testumgebung für die modulare Absicherung.

Die Verwendung virtueller Modelle in einer Testumgebung bietet immense Vorteile in Bezug auf flexible und kostengünstige Umgestaltung oder Testaufwand bzw. Testzeit, geht aber mit einer begrenzten Genauigkeit oder erhöhter Ungewissheit einher.[79] Für automatisierte Fahrzeuge herrscht Uneinigkeit über die Validität der Simulationsmodelle von Sensoren sowie der Umgebung. Hierbei fehlen einerseits Methoden und Metriken zur Validierung, andererseits ist unklar, ob existierende Modelle ausreichend valide sind.[80] Daher sind reale Sensordaten, die im realen Fahrbetrieb aufzuzeichnen sind, erforderlich. Diese Problematik besteht unabhängig davon, ob auf Modul- oder Systemebene getestet wird. Der Hauptunterschied zwischen Modul- und Systemtests ist die Darstellung anderer Module statt nur der Umgebung des Systems. Hierzu kommen verschieden Arten von X-in-the-loop (XiL) zum Einsatz. Hakuli und Krug[81] stellen beispielhaft sieben verschiedene mögliche Konfigurationen hiervon vor. Dazu zählen u. a. Model-in-the-Loop (MiL), Simulation-in-the-Loop (SiL) und Hardware-in-the-Loop (HiL) bis hin zu Vehicle-in-the-Loop (ViL) Testumgebungen vor. Sie unterscheiden dabei nur zwischen virtuellen und realen Komponenten.

---

[77] Jungmayr, S.: Testbarkeitsanforderungen an die Software (2008).

[78] Bass, L. et al.: Software Architecture in Practice, 4th Edition (2021), S. 167.

[79] ISO: ISO/TR 4804 - Safety and cybersecurity for automated driving systems (2020), S. 58–61.

[80] Leitner, A.: ENABLE-S3: Project Introduction (2020).

[81] Hakuli, S.; Krug, M.: Virtual Integration in the Development Process (2016), S. 160–161.

2.3 Schnittstellen und Komplexität

Das folgende Unterkapitel beschreibt Schnittstellen als eine der Kernkomponenten modularerer Architekturen als Detaillierung zu den vorgestellten Architektursichten. Aufgrund des hohen Einflusses der Wahl der Schnittstellen einer modularen Architektur auf die Komplexität eines Systems, wird außerdem der Begriff der Komplexität in der Systemtechnik und dessen Einflussfaktoren nach dem Stand der Technik beschrieben.

Schnittstellen

Schnittstellen sind die Verbindung zwischen Modulen, die einen Vertrag über das darstellen, was erforderlich ist, und das, was garantiert wird. Sie stellen Grenzen dar, über die Entitäten miteinander kommunizieren und interagieren.[82] Die strukturierte Kommunikation ist ein Hauptmerkmal von modularen Architekturen und beeinflusst daher stark die Möglichkeit Module individuell abzusichern. Pimmler und Eppinger[83] unterscheiden allgemein zwischen räumlichen, energetischen, informellen und materiellen Interaktionen. Sie beziehen sich dabei vorwiegend auf die Interaktionen bzw. die Schnittstellen eines mechatronischen Systems. Das Hauptproblem bei der Absicherung von automatisierten Fahrzeugen ist die enorme Anzahl an möglichen Variationen in der Informationsverarbeitung. Daher konzentriert sich die weitere Literaturauswahl auf den Bereich der Software- bzw. Informationstechnik.

Bass et al.[84a] charakterisieren in der Software- und Informationstechnik eine Schnittstelle als ein Vertrag zwischen einem Element und seinen Akteuren. Sie beschreiben Prinzipien zur Schnittstellengestaltung, die im Folgenden mit einer kurzen Erläuterung aufgeführt sind: [84b]

Prinzip der geringsten Überraschung: Die Erwartungen der Abonnenten sollten möglichst gut getroffen werden.

Prinzip der kleinen Schnittstellen: Schnittstellen sollten laut Bass so wenige Informationen wie möglich transportieren.

Prinzip des einheitlichen Zugangs: Laut diesem Prinzip sollten möglichst wenige Informationen über die Implementierung der Funktionen notwendig sein, um mit der Schnittstelle zu kommunizieren. Dieses Prinzip ist äquivalent zum Prinzip des *Information Hiding* in der Architekturgestaltung.

Prinzip der Wiederholungsvermeidung: Schnittstellen sollten Informationen möglichst nicht redundant anbieten.

Bass et al.[84a] beschreiben darüber hinaus drei Prinzipien zur Änderung von Schnittstellen: Abschaffung (eng.: depreciation), Versionierung (eng.: versioning) und Erweiterung (eng.:

[82] Broy, M.; Kuhrmann, M.: Einführung in die Softwaretechnik (2021), S. 658.

[83] Pimmler, T. U.; Eppinger, S. D.: Integration Analysis of Product Decompositions (1994).

[84] Bass, L. et al.: Software Architecture in Practice, 4th Edition (2021), a: S. 276, b: S. 278.

extension). Zur Abschaffung und Versionierung (die ebenfalls eine Abschaffung der vorherigen Schnittstelle erfordert) betonen Bass et al., dass alle Akteure darüber zu informieren sind und dies z. B. durch eine Versionsnummer klar zugeordnet wird. Die Erweiterung einer Schnittstelle erfordert laut Bass et al. eine Kompatibilität zur bestehenden Schnittstelle oder eines Mediators zur Übersetzung der Informationen.

Komplexität

Kröger und Ayoub[85] weisen auf die Unvorhersehbarkeit des Systemverhaltens hin, wenn das prädizierte Verhalten nur auf dem Wissen der Eigenschaften oder des Verhaltens der einzelnen Komponenten beruht. Dies ist gleichbedeutend den in Kapitel 2.2 erläuterten emergenten Eigenschaften. Kröger und Nan[86] definieren Eigenschaften komplexer Systeme basierend auf Eigenschaften aus dem Komplexitätsmanagement für Unternehmen und Produkte[87]. Kröger und Ayoub ziehen diese Eigenschaften heran, um zwischen komplexen und komplizierten Systemen zu unterscheiden. Die Society for Risk Analysis (SRA) definiert komplexe Systeme als solche, die sich genau dadurch auszeichnen, dass die Vorhersehbarkeit des Systemverhaltens aus dem Wissen über einzelne Komponenten nicht möglich ist.[88] Kröger und Ayoub implizieren daher, dass eine solche Vorhersage zwar nicht für komplexe, aber für komplizierte Systeme potentiell möglich ist. Als komplexe Systeme bezeichnen sie offene und veränderbare Systeme, wie z.B. Stromnetze oder den Aktienmarkt. In diesen Systemen kann jede Komponente individuell operieren und sich verändern, während sie gleichzeitig in hohem Maße miteinander verbunden sind und voneinander beeinflusst werden. Dies wird auch als hohe Stärke zwischen den Schnittstellen bezeichnet. Außerdem ist das Verhalten der Komponenten eines komplexen Systems teilweise unbekannt und kann sich ändern. Komplizierte Systeme werden dagegen u.a. dadurch charakterisiert, dass das Systemverhalten trotz einer hohen Anzahl an Komponenten vorhersehbar ist und verglichen mit komplexen Systemen wenig Dynamik besitzt.[86]

Nach Maurer[89] ist die Komplexität eines Systems durch viele verschiedene unklare Zuordnungen zwischen verschiedenen Architekturen gekennzeichnet. Zum Beispiel gibt es in einem komplexen System keine Eins-zu-Eins-Abbildung zwischen den beschriebenen Funktionen und den Softwaremodulen. Maurer[89] verweist auf verschiedene Methoden zum Umgang mit Komplexität. Die primäre Domäne hierfür ist das *Systems Engineering*, als ein wesentlicher Teil der Systementwicklung. Die Definition von Systems Engineering ist weit gefasst und beinhaltet den gesamten Entwicklungsprozess (vgl. die Definition von Systems

[85] Kröger, W.; Ayoub, A.: Towards «type approval» of automated vehicles (2022).

[86] Kröger, W.; Nan, C.: Dealing with complexity (2019).

[87] Schoeneberg, K.-P.: Komplexitätsmanagement in Unternehmen (2014), S. 13–19.

[88] SRA: Society for Risk Analysis Glossary (2018).

[89] Maurer, M.: Complexity management in engineering design (2017), S. 92.

Engineering des International Council on Systems Engineering[90]). Für komplexe Systeme führt Sheard[91] die Bereiche *Discovery* und *Program Systems Engineering* ein. *Discovery Systems Engineering* umfasst Aufgaben, die auf die Modellierung und das Verständnis des Problemraums abzielen. *Program Systems Engineering* legt den Schwerpunkt auf die Definition des Lösungsraums sowie die Spezifikation technischer und Mensch-Maschine-Schnittstellen.[91]

2.4 Sicherheitsfreigabe

Wie jedes sicherheitsrelevante System erfordert ein hochautomatisiertes Fahrzeug eine Sicherheitsfreigabe. Mit der Terminologie in Kapitel 2.1 wird bereits erläutert, dass für die Sicherheitsfreigabe eine Sicherheitsargumentation notwendig ist. Im Folgenden wird daher der Stand der Technik zur Erbringung einer Sicherheitsargumentation dargelegt. Zuerst werden Normen und Regularien beschrieben, die für die Automobilindustrie allgemein gültig sind. Die Recherche von Normen und Regularien anderer Branchen (u. a. der Luft- und Raumfahrt) ergibt, dass diese nahezu gleiche Prozesse und Methoden verwenden. Neue Erkenntnisse für die modulare Absicherung hochautomatisierter Fahrzeuge sind darin nicht ersichtlich. Auf deren detaillierte Beschreibung wird in diesem Kapitel daher verzichtet. Eine Übersicht und Analyse zu diesen Normen präsentiert u. a. Reschka[92]. Im zweiten Teil folgt mit der Goal Structuring Notation (GSN) eine spezielle Beschreibungsform der Sicherheitsargumentation, die in der vorliegenden Arbeit Anwendung findet.

2.4.1 Normen und Regularien

Die Absicherung stützt sich in der Automobilindustrie im Wesentlichen auf die Norm ISO 26262 sowie die ISO/DIS 21448. Im Folgenden wird zuerst ein Überblick über die Norm ISO 26262 gegeben, die zur Sicherstellung einer funktionalen Sicherheit Anwendung findet. Dabei wird insbesondere deren Betrachtung von Systemen und Subsystemen bzw. Modulen beschrieben. Daraufhin folgt die Beschreibung der ISO/DIS 21448, die ergänzend zur Norm ISO 26262 für automatisierte Fahrfunktionen zum Einsatz kommt. Darauf folgt eine Übersicht geltender Regularien, die für die Zulassung hochautomatisierter Fahrzeuge und speziell für die Freigabe von Modulen relevant sind.

---

[90] INCOSE: Systems Engineering Definition (2022).

[91] Sheard, S. A.: Three Types of Systems Engineering Implementation (2000).

[92] Reschka, A.: Dissertation, Fertigkeiten- und Fähigkeitengraphen von automatisierten Fahrzeugen (2017), S. 69–88.

ISO 26262

Die funktionale Sicherheit nach ISO 26262 fordert die "Abwesenheit von unvertretbarem Risiko"[93] im Bereich der elektrischen und elektronischen (E/E) Systeme.[94a] Die Norm zielt auf die Identifizierung von Gefahren, die durch das Fehlverhalten des betrachteten Systems verursacht werden. Auf Basis dieser Gefährdungen werden im Rahmen einer Gefährdungsanalyse und Risikobewertung (HARA[95]) potentielle Risiken bestimmt. Aus den Ergebnissen wird ein Automotive Safety Integrity Level (ASIL) ermittelt, der definiert, welche Maßnahmen zur Risikominderung in Entwicklung und Erprobung durchgeführt werden müssen, um ein unvertretbares Risiko zu vermeiden.[94b]

In der Norm ist das betrachtete Objekt (als "Item" bezeichnet) ein Teilsystem eines Fahrzeugs. Die Norm verlangt jedoch, dass „die Sicherheitsziele für das Item in einem repräsentativen Kontext auf Fahrzeugebene validiert werden“[96].

Im Folgenden wird passend zur Definition eines Systems nach Kapitel 2.1 auf die Systemebene statt auf die Fahrzeugebene Bezug genommen. Dies ist notwendig, um Module außerhalb des Fahrzeugs, die Module innerhalb des Fahrzeugs beeinflussen nicht außer Acht zu lassen. Im Forschungsprojekt UNICAR*agil* wird z. B. die sogenannte Leitwarte als eigenständiges Modul betrachtet, die u. a. die manuelle Steuerung des Fahrzeugs in bestimmten Situationen via Funkverbindung übernehmen kann.

Die ISO 26262 erlaubt eine Dekomposition von Sicherheitsanforderungen, um die damit verbundene ASIL durch den Einsatz von Redundanzen zu reduzieren. Die zugeordneten Software- oder Hardwarekomponenten zur Umsetzung der Sicherheitsanforderungen müssen dazu ausreichend unabhängig sein. Diese Unabhängigkeit ist laut der Norm auf Systemebene zu verifizieren, sodass auch Systemtests notwendig sind.[94c] Eine Absicherung einzelner Komponenten ist daher nicht vorgesehen und mit der Forderung nach Tests auf der Fahrzeugebene auch für ein Item nicht zulässig.

Die Norm empfiehlt jedoch bereits auf der Ebene der Software- und Hardwarekomponenten ein umfassendes Spektrum an Analyse- und Testmethoden. Tabelle 2-1 zeigt eine Übersicht aller von der ISO 26262 empfohlenen Tests für eine ASIL D Bewertung.[94e] ASIL D stellt die höchsten Anforderungen an Analyse und Test der zugeordneten Komponenten. Die Testmethoden nutzen als Grundlage die Ergebnisse der Analysemethoden, sodass hier auf die Auflistung der Analysemethoden mit größtenteils äquivalenten Bezeichnungen verzichtet wird.

Die Auflistung der Testmethoden zeigt, dass die Fahrzeugebene nur beim Back-to-Back-Test fehlt. In Back-to-Back-Tests werden die Antworten zweier Systeme oder Modelle auf die

[93] Eng.: „absence of unreasonable risk“[94d]

[94] ISO: ISO 26262 - Road vehicles – Functional safety (2018), a: Teil 1, S.1 b: Teil 1, S.2 c: Teil 9, S.4–9, d: Teil 1, S. 14, e: Teil 4, S. 17.

[95] Abkürzung für den englischen Begriff “Hazard Analysis and Risk Assessment“

[96] Eigene Übersetzung nach ISO: ISO 26262 - Road vehicles – Functional safety (2018), Teil 4, S. 25.

gleichen Stimuli verglichen, bspw. des realen Systems mit denen des entsprechenden Simulationsmodells.[94e] Back-to-Back-Tests werden daher hauptsächlich zur Validierung von Simulationsmodellen oder zum Vergleich von zwei Versionen einer Komponente verwendet. Meistens ist ein Simulationsmodell für das gesamte Fahrzeug nicht vorhanden oder nicht darauf ausgelegt ein valides Systemverhalten zu erzeugen. Daher ist der Back-to-Back-Test auf Fahrzeugebene möglicherweise nicht zielführend.

Auf den niedrigeren Hierarchieebenen fehlen dagegen mehrere Testmethoden. Der Nutzertest unter realen Bedingungen, der ggü. der Fahrzeugebene auf Hardware-/Software- und Systemebene fehlt, kann per Definition nur am Gesamtfahrzeug durchgeführt werden. Allerdings können die Umgebungsbedingungen und das Benutzerverhalten auch für die Simulation modelliert werden. Damit kann zumindest ein Nutzertest mit ggf. ungewisser Aussagekraft auf unteren Ebenen durchgeführt werden. Der Langzeittest kann mit dem Zeitplan des Entwicklungsprozesses kollidieren, kann aber unter Berücksichtigung der Ungewissheiten der dafür notwendigen Simulationsmodelle auch auf niedrigere Ebenen abgeleitet werden.

Tabelle 2-1: Von der ISO 26262 empfohlene Testmethoden (mit "x" markiert) für verschiedene Hierarchieebenen bei ASIL D Bewertung der zugeordneten Sicherheitsziele.[97]

Testmethode	Hardware-/Softwareebene	Systemebene	Fahrzeugebene
Anforderungstest	x	x	x
Fehlerinjektionstest	x	x	x
Back-to-Back Test	x	x	-
Leistungstest	x	x	x
Test externer Schnittstellen	x	x	x
Test interner Schnittstellen	x	x	x
Prüfung der Schnittstellenkonsistenz	x	x	x
Fehlerratetest	x	x	x
Test der Ressourcennutzung	x	x	x
Stresstest	x	x	x
Test auf Belastbarkeit und Robustheit der Schnittstellen unter bestimmten Umweltbedingungen	-	x	x
Test der Interaktion/Kommunikation	-	x	x
Tests abgeleitet aus der Felderfahrung	-	x	x
Langzeittest	-	-	x
Nutzertest unter realen Bedingungen	-	-	x

Zwei Tests, die auf der Hardware-/Softwareebene im Vergleich zur Systemebene fehlen, betreffen die Kommunikation zu anderen Hardware-/Softwarekomponenten („Test auf Belastbarkeit und Robustheit der Schnittstellen unter bestimmten Umweltbedingungen“ und „Test der Interaktion/Kommunikation“). Diese werden nur bei fertig entwickelten Komponenten auf Systemebene für möglich gehalten. Tests aus der Felderfahrung auf der Hardware-/Softwareebene werden von der Norm ebenfalls nicht empfohlen. Dies ist bei einer Neuentwicklung sinnvoll, da Felderfahrungen erst in einem späten Stadium des Entwicklungsprozesses

[97] Übersetzung von Klamann, B.; Winner, H.: Comparing Different Levels of Technical Systems (2021).

möglich sind. Auf die Nutzung der Felderfahrungen aus vergangenen Entwicklungen wird dagegen nicht hingewiesen.[98]

Auffällig ist, dass mit Ausnahme des Back-to-Back Tests alle Testmethoden für die Hardware-/Softwareebene auch auf System- und Fahrzeugebene erfolgen sollen. Die Norm scheint davon auszugehen, dass die Ungewissheiten auf der niedrigeren Ebene immer zu hoch sind. Daher ist (insbesondere für ASIL D klassifizierte Items) eine Überprüfung auf höheren Hierarchieebenen notwendig, die die Abhängigkeiten automatisch mitberücksichtigen.

ISO/DIS 21448

Die in der Norm ISO 26262 beschriebenen Methoden konzentrieren sich auf Gefährdungen durch Unzulänglichkeiten in der Implementierung eines Items oder durch zufällige Hardwarefehler (von Koopman et al.[99] zusammenfassend als "equipment faults" bezeichnet). Eine fehlerhafte Spezifikation der Funktion wird von der Norm ISO 26262 jedoch nicht betrachtet. Ergänzend dazu beschreibt die Norm ISO/DIS 21448 daher Methoden zur Identifizierung und Reduzierung von Unzulänglichkeiten in der Spezifikation der vorgesehenen Funktionalität, die Gefahren in Form von unsicherem Verhalten verursachen. In diesem Umfang wird die Norm als SOTIF bezeichnet, das Akronym für „Safety Of The Intended Functionality“ (de.: Sicherheit der vorgesehenen Funktion). SOTIF wurde im Jahr 2019 eingeführt für automatisierte Fahrfunktionen mit SAE Level 1-2.[100] Mit der Neuveröffentlichung im Jahr 2022 gilt die Norm auch für Systeme bis SAE Level 5. Die Norm bezieht sich auf "Unzulänglichkeiten der Spezifikation oder Leistungseinschränkungen bei der Implementierung von E/E-Elementen" im betrachteten System.[101a] Sie schließt jedoch Gefährdungen aus, die direkt von einer Komponente aufgrund von Abweichungen von der Spezifikation ausgehen, d.h. durch Irrtümer bei der Implementierung. Dies ist Teil der Norm ISO 26262.[101b] Nolte et al.[102] definieren Einflüsse auf die Fahraufgabe mit dem Begriff "externes Verhalten". Externes Verhalten beeinflusst direkt andere Verkehrsteilnehmer und kann daher Gefahren erzeugen. Automatisierte Fahrzeuge erfordern daher neben der funktionalen Sicherheit auch die Berücksichtigung der Verhaltenssicherheit zur Vermeidung gefährlichen Verhaltens.[103] Die Spezifikationen der Komponenten können dabei jedoch initial fehlerhaft sein. Dadurch ausgelöstes unerwünschtes Verhalten adressiert SOTIF weiterhin mit der Durchführung von Systemtests und korrigiert darauffolgend ggf. die Spezifikation der Komponenten.

---

[98] ISO: ISO 26262 - Road vehicles – Functional safety (2018), Teil 4.

[99] Koopman, P. et al.: A Safety Standard Approach for Fully Autonomous Vehicles (2019).

[100] ISO: ISO/PAS 21448: Safety of the intended functionality (2019).

[101] ISO: ISO/DIS 21448: Safety of the intended functionality (2022), a: S. 10 b: S. 7–9, c: S. 149, d: S. 152-153.

[102] Nolte, M. et al.: Skill- And Ability-Based Development Process (2017).

[103] WAYMO: Waymo Safety Report (2021), S. 11.

SOTIF weist allerdings auf die Möglichkeit einer modularen Sicherheitsargumentation hin, um den Aufwand für die Sicherheitsvalidierung zu verringern.[101c] In der Norm wird ein vereinfachtes Beispiel für eine Funktionalität gegeben, die in der Lage ist, innerhalb eines geraden Fahrstreifens zu fahren, ohne mit anderen Objekten zu kollidieren. Es wird ein modularer Ansatz vorgestellt, der die erforderliche Umfeldwahrnehmung auf Komponentenebene analysiert. Die Testaufwandsreduktion wird dabei mit der Reduktion der Testfälle argumentiert, die für diese Komponente relevant sind. Allerdings liefert das Beispiel nur die Berechnung für eine theoretische Teststreckenreduzierung durch Betrachtung der einzelnen Komponenten, nicht aber für eine modulare Sicherheitsargumentation. Auf eine notwendige Analyse der Einflüsse zwischen den Komponenten wird hingewiesen. Die beschriebenen Maßnahmen zur Analyse und Prüfung der entsprechenden Einflüsse zur Erreichung ausreichender Unabhängigkeit zwischen Modulen erfordern jedoch weiterhin den Einbezug des Systems und von Systemtests.[101d]

Geltende Regularien

Regularien zur Zulassung automatisierter Fahrfunktionen sowie hochautomatisierter Fahrzeuge existieren in Deutschland auf nationaler, auf europäischer sowie auf internationaler Ebene. In Deutschland ermöglicht das Straßenverkehrsgesetz nach § 1a bis § 1g[104] seit 2021 eine Zulassung „autonomer Fahrzeuge“ wie in § 1d formuliert. Dies soll Fahrzeuge mit Automatisierung bis Level 5 nach SAE[105] einschließen. Die Europäische Union beschreibt in der Verordnung 2022/1426[106a] Verfahren zur Typgenehmigung automatisierter Fahrsysteme. Diese beinhaltet neben möglichen Elementen der Einsatzumgebung auch mögliche Szenarien und Testverfahren, betont jedoch, dass Hersteller diese für das eigene System und dessen Einsatzumgebung ergänzen müssen.[106b] Laut der Verordnung erfolgt die Prüfung der Ergebnisse der Validierung von Simulationsumgebung am „vollständig integrierten Werkzeug“[106c]. Die „Validierung des integrierten Systems“[106c] wird zwar genannt, allerdings lediglich als möglicher Schritt der Freigabe aufgelistet. Es sind daher weder Systemtests noch die Validierung am vollständig integrierten System vorgeschrieben.

Auf internationaler Ebene existieren von der Vereinten Nationen mehrere verfasste Regelungen für die Zulassung verschiedener Assistenzsysteme wie z. B. Notbremsassistenten (UNECE R152[107]) oder automatische Spurhalteassistenzsysteme (UNECE R157[108a]). Die UNECE R157 beschreibt gegenüber der Verordnung 2022/1426 Szenarien und Prüfkriterien noch detaillierter, allerdings entsprechend nur für ein Level 3 System. Darüber hinaus be-

[104] Bundesrepublik Deutschland: Straßenverkehrsgesetz (StVG) (2023).

[105] SAE International: SAE J3016 (2021).

[106] Europäische Kommission: Durchführungsverordnung (EU) 2022/1426 (2022), a: -, b: S. 24, c: S. 58.

[107] UNECE: UN-Regulation No. 152 - Approval of the Advanced Emergency Braking System (AEBS) (2020).

[108] UNECE: UN-Regelung Nr. 157 - Genehmigung automatischer Spurhalteassistenzsysteme (2022), a: -, b: S.106.

schreibt die Regelung, dass von der Prüfbehörde das Sicherheitskonzept sowie die Verifikations- und Validierungspläne auf Fahrzeug- und Systemebene geprüft werden.[108b] Auch an dieser Stelle werden Fahrzeug- oder Systemtests allerdings nicht vorgeschrieben.

Mit dem Ziel einer modularen Absicherung den Absicherungsaufwand auch bei Änderungen von Modulen zu reduzieren, ist auch die dafür gültige Regelung UNECE R156[109a] zu beachten. Die für Fahrzeuge mit beliebiger Automatisierungslevel gültige UNECE R156 fordert Vorkehrungen zur Genehmigung von Softwareupdates in Steuergeräten von bereits zugelassenen Fahrzeugen. Hierbei wird ein sogenanntes Software Update Management System (SUMS) gefordert.[109b] Das SUMS soll Prozesse beschreiben und dokumentieren, die für ein Softwareupdate durchgeführt werden. Hieraus soll insbesondere hervorgehen, welche funktionalen Änderungen vorgenommen werden und wie diese andere Systeme beeinflussen. Bei Einflüssen auf freigaberelevante Funktionen oder Bauteile ist diese Freigabe ggf. erneut zu erbringen.[109c] Genaue Vorgaben, durch welche Methoden und Kriterien die Änderungen und Einflüsse zu analysieren sind, werden in der Regelung nicht beschrieben.

Die Regularien stellen das Rahmenwerk für die Zulassung automatisierter Fahrzeuge dar und beschreiben Abnahmekriterien, stellen allerdings keine Methoden oder Prozesse bereit, die eine Sicherheitsargumentation unterstützen. Sie fordern zwar bspw. die Beschreibung einer Systemarchitektur[111a], bleiben dabei jedoch möglichst technologieneutral, sodass genauere Vorgehensweisen oder Gestaltungskriterien offenbleiben.[110] Stattdessen wird bspw. auf die zuvor erläuterten Normen ISO 26262 oder SOTIF verwiesen.[111b]

2.4.2 Sicherheitsargumentation mit der Goal Structuring Notation

Die beschriebenen Prozessschritte und Methoden der existierenden Normen werden als ausreichend für eine Sicherheitsargumentation gesehen, wenn sie von Entwicklern entsprechend umgesetzt werden. In der Norm ISO 26262 werden die notwendigen Prozessschritte auch mit der ASIL Klassifizierung erweitert (vgl. Kapitel 2.4.1). Trotzdem sind die Prozessschritte der Normen lediglich Empfehlungen.[112] Die endgültige Sicherheitsargumentation ist von den Entwicklern selbst aufzubauen. Hierbei spielt sowohl die Menge als auch die Qua-

[109] UNECE: UN Regulation No. 156 - Provisions concerning software updates (2021), a: -, b: S. 4 c: S. 8-10.

[110] Gesmann-Nuissl, D.; Tacke, I.: Funktionale Sicherheit KI-basierter Systeme im Automobilsektor (2022).

[111] UNECE: UN-Regelung Nr. 157 - Genehmigung automatischer Spurhalteassistenzsysteme (2022), a: S. 104-105, b: S. 109.

[112] ISO: ISO 26262 - Road vehicles – Functional safety (2018), Teil 4.

lität der durchgeführten Prozessschritte eine wesentliche Rolle für eine erfolgreiche Absicherung.[113] Kelly und Weaver[114] zeigen, dass die dabei übliche Vorgehensweise, die Sicherheitsargumentation in natürlicher Sprache aufzusetzen, Missverständnisse durch mehrdeutige oder unklare Beschreibungen verursacht.

Zur klareren Kommunikation von Argumenten für die Sicherheit technischer Systeme bietet die Goal Structuring Notation (GSN) die Möglichkeit die Sicherheitsargumentation in einer formalisierten Sprache grafisch darzustellen.[115] Die Norm SOTIF schlägt die GSN ebenfalls als Möglichkeit zur Sicherheitsargumentation vor und nutzt diese bereits, um die in der Norm vorgestellten Schritte logisch in einer Argumentationskette zu verknüpfen.[116]

In einer GSN werden übergeordnete Ziele (eng.: Goal, Abk.: G) in Subziele dekomponiert, die zur Argumentation beitragen, so dass das jeweils übergeordnete Ziel erfüllt wird. Ziele werden bis zu einer Ebene dekomponiert, auf der die Erreichung eines Ziels durch eine Lösung (eng.: Solution, Abk.: Sn) bestätigt werden kann. Lösungen stellen üblicherweise die Durchführung etablierter Methoden (bspw. eine Fehlerbaumanalyse) dar, können aber individuell gewählt werden. Zur Erhöhung der Verständlichkeit wird die Dekomposition der Ziele bei Bedarf durch die Angabe von Strategien (eng.: Strategy, Abk.: S) unterstützt, die die Ableitung der jeweiligen Ziele näher erläutern. Ebenso kann bei Bedarf der Kontext (eng.: Context, Abk.: C) oder die Annahmen (eng.: Assumption, Abk.: A) angegeben werden, unter denen eine Strategie oder ein abgeleitetes Ziel gültig ist. Wird ein Ziel im vorliegenden GSN nicht weiterentwickelt, wird unterhalb des Ziels eine Raute angefügt. Ziele werden bspw. nicht weiterentwickelt, wenn noch keine weiteren Ziele oder Lösungen ermittelbar sind oder wenn auf bereits existierende Lösungen verwiesen wird. Die zugehörigen Symbole zur Darstellung der Elemente sind in Abbildung 2-6 dargestellt. Innerhalb der Symbole werden die ID und die Beschreibung des Elements eingefügt.

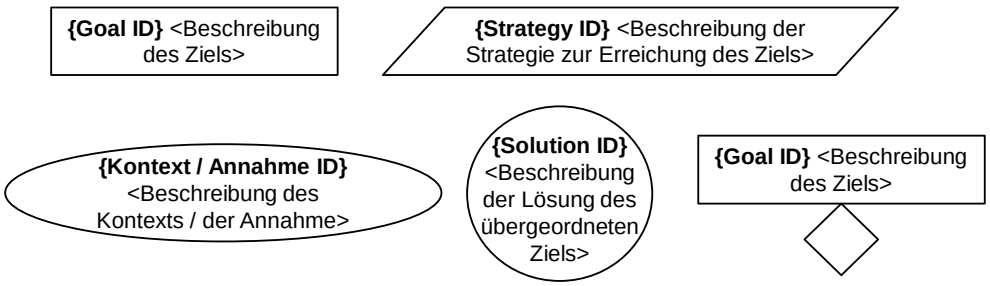


Abbildung 2-6: Notationselemente der Goal Structuring Notation (GSN).

Die Verbindung zwischen Zielen und Subzielen erfolgt mit gefüllten Pfeilen Richtung Subziel. Die Verbindung zwischen Strategien oder Zielen zu Kontext oder Annahmen wird mit

[113] Despotou, G.; Kelly, T.: Argument modularity for safety assurance (2008).

[114] Kelly, T.; Weaver, R.: The Goal Structuring Notation (2004).

[115] SCSC: Goal Structuring Notation Community Standard (2021).

[116] ISO: ISO/DIS 21448: Safety of the intended functionality (2022), S.25, S. 64 ff.

nicht gefüllten Pfeilen Richtung Kontext oder Annahme dargestellt.[117] Ein beispielhafter GSN Graph hiermit ist in Abbildung 2-7 dargestellt. Der dargestellte GSN Graph zeigt einen Ausschnitt des Beispiels nach Kelly und Weaver[118] zur Sicherheitsargumentation des Kontrollsystems einer Hydraulikpresse. Hierbei wird zunächst auf generischer Basis argumentiert, dass einerseits alle Sicherheitsanforderungen erfüllt (S1) und identifizierte Gefährdungen vermieden werden (S2). Spezifischer soll ersteres dadurch erreicht werden, dass eine Zustandsmaschine bei Loslassen der Steuerelement den Pressvorgang abbricht (vgl. G3, G7 und Sn3) sowie durch sicheres Anhalten bei Komponentenausfällen (G4). Das Ziel G4 wird dabei im Graph nicht weiterentwickelt und stattdessen z.B. in der Sicherheitsargumentation der einzelnen Komponenten betrachtet. Für die Strategie S2 werden die Ziele G8 und G9 verfolgt, die ein unbeabsichtigtes Öffnen oder Schließen der Presse nur durch Bauteilversagen erreichen sollen. Hierzu wird die Lösung Sn3 zur Entwicklung eines Fehlerbaums vorgeschlagen. Zusätzlich wird die Lösung Sn4 angegeben, die alle identifizierten Gefährdungen in Tests des Kontrollsystems adressiert.

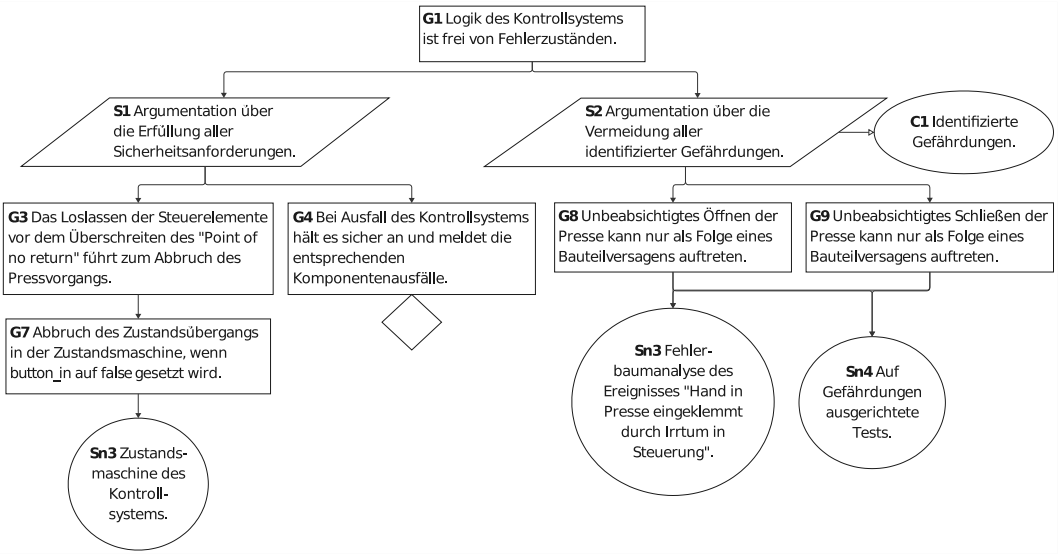


Abbildung 2-7: Beispiel für einen GSN Graph zur Argumentation der Sicherheit des Kontrollsystems (eng.: Control System, Abk.: C/S) einer Hydraulikpresse. Beispiel nach Kelly und Weaver[118] übersetzt und gekürzt dargestellt.

2.5 Defizite im Stand der Technik

Der Stand der Technik weist mehrere Schwachstellen und Lücken auf, um das Ziel einer modularen Absicherung hochautomatisierter Fahrzeuge zu erreichen. Unter Anwendung konventioneller Methoden der Absicherung kann es weiterhin „zu einem potentiell gefähr-

[117] SCSC: Goal Structuring Notation Community Standard (2021), S. 36.

[118] Kelly, T.; Weaver, R.: The Goal Structuring Notation (2004).

lichen Verhalten kommen, das durch die vorgesehene Funktionalität verursacht wird, obwohl diese Systeme frei von den in der Norm ISO 26262 angesprochenen Fehlern sind“[119]. Die Norm SOTIF adressiert daher bereits mögliche Unzulänglichkeiten in der Spezifikation, die das externe Verhalten eines automatisierten Fahrzeugs beeinflussen. Allerdings werden dabei lediglich Empfehlungen beschrieben. Eine Auswahl konkreter Entwicklungs- und Testschritte fehlt bisher. In den Normen präsentierte Beispiele beziehen sich auf einzelne Ansätze, die vorwiegend in Forschungsprojekten Anwendung finden. Bei den Vorschlägen zu Tests einzelner Komponenten oder Module bleiben die Normen vage bzgl. deren Rolle für die Absicherung, sodass weiterhin Systemtests als notwendig definiert werden.[120, 121] Darüber hinaus sind die Begriffe der verschiedenen Hierarchieebenen unscharf definiert, sodass eine Einordung der notwendigen Prozessschritte auf verschiedenen Hierarchieebenen nicht eindeutig ist. Die Betrachtung eines Items nach Norm ISO 26262 als Modul ist bspw. nicht möglich, da für dieses die Betrachtung eines vollständigen Regelkreises sowie die finale Sicherheitsvalidierung auf Fahrzeugebene gefordert wird.[122a] Bestehende Regularien ermöglichen zwar die Zulassung hochautomatisierter Fahrzeuge und stellen teilweise Freigabekriterien auf, für die Umsetzung der dazu notwendigen Methoden verweisen die Regularien jedoch auf die beschriebenen Normen. Die Regularien schränken dabei die Möglichkeit einer modularen Absicherung nicht ein, da die Betrachtung der Systemebene in der Entwicklung zwar begutachtet, aber die Durchführung bspw. Von Systemtests nicht explizit gefordert wird. Die UNECE R156 fordert indes eine Analyse der Einflüsse nach Änderungen an Softwarekomponenten durch Updates und motiviert damit ebenfalls eine modularen Absicherung, um diesen Analyseaufwand möglichst gering zu halten.

Modulare Architekturen finden bereits breite Anwendung im Stand der Technik. Die Vorteile für die Entwicklung und den Nutzer eines technischen Systems werden u. a. in Normen hervorgehoben und Zielvorgaben zur Erreichung von Modularität gegeben.[122b] Allerdings bleiben diese Vorgaben und die Charakteristika einer modularen Architektur offen. Ebenso werden Vorteile für die Testbarkeit bei Einsatz einer modularen Architektur genannt, ein Verzicht auf Integrationstests wird jedoch nicht betrachtet. Die Entwicklung von Anforderungen und Methoden zur Absicherung einzelner Module ist in der Literatur daher nicht auffindbar. Existierende Testbarkeitsanforderungen zielen vorwiegend auf die Gestaltung der Architektur ab. Als wesentliche Herausforderung wird dabei die Validität der Testumgebungen identifiziert, es existieren jedoch keine etablierten strukturierten Vorgehensweisen zur Auswahl und Gestaltung von Testumgebungen speziell für die Absicherung einzelner Module.

---

[119] Eigene Übersetzung nach ISO: ISO/DIS 21448: Safety of the intended functionality (2022), S. 7.

[120] ISO: ISO 26262 - Road vehicles – Functional safety (2018), Teil 4, S. 25.

[121] ISO: ISO/DIS 21448: Safety of the intended functionality (2022), S. 54–56.

[122] ISO: ISO 26262 - Road vehicles – Functional safety (2018), a: Teil 1: S. 16, S. 25, b: Teil 5, S. 18.

In Bezug auf die Absicherung lässt sich allgemein eine grundsätzliche Skepsis gegenüber einer reinen Betrachtung der Modulebene erkennen. Nach dem Stand der Technik werden bereits hohe Aufwände in Form einer kaum überschaubaren Bandbreite an Test- und Entwicklungsmethoden zur Erreichung eines hohen Qualitätsniveaus von Komponenten oder Modulen betrieben. Trotzdem schreibt die ISO 26262 eine Sicherheitsvalidierung auf Fahrzeug- oder Systemebene vor. Das Bestreben eines erhöhten Qualitätsniveaus dient bisher lediglich zur Vermeidung aufwendiger Korrekturen zu einer späten Phase im Entwicklungsprozess. Ein Beitrag zur Absicherung erfolgt nur implizit. Die vage Beschreibung insbesondere zur Ableitung von Integrationstests lässt darüber hinaus unklar, was der reinen Betrachtung der Modulebene fehlt, um auf Integrationstests verzichten zu können. Eine Erweiterung der Prozessschritte unter Berücksichtigung der Komplexität eines Systems, wie es bspw. die Norm UK MoD Def Stan 00-56[123] fordert, erfolgt in der Norm ISO 26262 lediglich implizit über eine potentiell größere Menge an Sicherheitsanforderungen mit potentiell höheren ASIL Einstufungen.

[123] UK MoD: Defence Standard 00-56 - Safety Management Requirements (2017).

3 Stand der Wissenschaft

Das folgende Kapitel präsentiert aktuelle Forschungsarbeiten zur Absicherung automatisierter Fahrzeuge und zu Ansätzen der Modularisierung insbesondere von Softwarekomponenten. Im Gegensatz zum Stand der Technik werden in diesem Kapitel Arbeiten vorgestellt, deren Methoden nicht in Lehrbüchern oder Normen aufgenommen sind und von denen kein Einsatz in der Industrie bekannt ist. Dies bedeutet, dass der jeweilige Nutzen der vorgestellten Ansätze noch nicht unter Beweis gestellt werden konnte.

Die Kapitel sind themenspezifisch aufgeteilt und greifen Ansätze aus verschiedenen Forschungsprojekten auf, die nach den Kriterien zur Literaturauswahl relevant für die vorliegende Arbeit sind. Gleichzeitig arbeiten größere Forschungsprojekte (wie bspw. das Projekt „Verification and Validation Methods“) an verschiedenen Fragestellungen zur Absicherung von HAF, die nicht Teil dieser Arbeit sind. Am Ende des Kapitels wird das in dieser Arbeit verfolgte Konzept in die vorgestellten Forschungsarbeiten eingeordnet, um daraus die weiterhin offenen Forschungsfragen abzuleiten.

3.1 Kriterien zur Literaturauswahl

Mit der in dieser Arbeit verfolgten Hauptproblematik der Sicherheitsargumentation hochautomatisierter Fahrzeuge ergeben sich zwei übergeordnete Bereiche für die folgende Literaturauswahl. Zuerst ist es wichtig, Ansätze der Forschung für eine Sicherheitsargumentation hochautomatisierter Fahrzeuge aus der laufenden Forschung nachzuvollziehen. Die Auswahl in den folgenden Unterkapiteln greift insbesondere Absicherungsansätze auf, die einerseits Probleme bei Betrachtung der Systemebene aufzeigen, andererseits solche, die bereits Charakteristika besitzen, die eine modulare Betrachtung begünstigen. Hierbei finden sich Ansätze, die unter ausreichender Modularität bspw. eine Testaufwandsreduktion zeigen. Zusätzlich werden in einem deutlich breiteren Bereich und in verschiedenen Domänen veröffentlichte Ansätze vorgestellt, die vergleichbare Ansätze einer modularen Absicherung verfolgen. Wie bereits im vorigen Kapitel dargestellt, existiert nach dem Stand der Technik bereits eine hohe Anzahl verschiedener Methoden zum Test von Komponenten. Es zeigt sich jedoch, dass diese Methoden keine klare Berücksichtigung in der Sicherheitsargumentation hochautomatisierter Fahrzeuge finden. In den folgenden Unterkapiteln werden daher zusätzlich Forschungsarbeiten zur Modellierung hochautomatisierter Fahrzeuge aus verschiedenen Perspektiven bzw. anhand von Architektursichten vorgestellt, die auch das Verhalten eines Fahrzeugs darstellen. Ebenso werden neue Konzepte zur Erreichung einer modularen Architektur und entsprechend einer modularen Sicherheitsargumentation vorgestellt.

3.2 Absicherung hochautomatisierter Fahrzeuge

Wachenfeld und Winner[124a] berechnen am Beispiel einer automatisierten Fahrfunktion für die Autobahn, dass deren Absicherung mit bestehenden Methoden der Automobilindustrie mehr als 6 Milliarden Testkilometer erfordern würde.[125] Kalra und Paddock[126] ermitteln unter verschiedenen Annahmen ebenfalls, dass der Nachweis, dass ein automatisiertes Fahrzeug weniger tödliche Unfälle als ein Mensch verursacht, mehrerer Milliarden Testkilometer erfordert. Die Forschung entwickelt daher neue Ansätze zur Reduktion des Freigabeaufwands automatisierter Fahrzeuge.

Bereits entwickelte Ansätze zu Absicherungsstrategien werden von Junietz et al. [127] zusammengefasst. Sie gehen von drei Kategorien an Strategien aus: Tests in der realen Welt, szenariobasierte Tests und formale Verifikation. Tests in der realen Welt erfordern dabei die Fahrt einer ausreichenden Anzahl von Kilometern in einer repräsentativen Umgebung, um einen statistischen Sicherheitsnachweis zu erbringen. Dies entspricht den Annahmen von Wachenfeld und Winner[124b] zur Berechnung der notwendigen 6 Milliarden Testkilometer für die Freigabe einer Automation für Autobahnen. Junietz et al. schlagen vor, die Extremwerttheorie (EVT) zu verwenden, um die gefahrenen Testkilometer zu extrapolieren. Dieser Ansatz setzt jedoch das Vorhandensein ausreichender Metriken voraus, "die von kritischen Situationen auf Unfälle einer bestimmten Kategorie hinweisen".[127] Zwar gibt es bereits eine Vielzahl an Metriken zur Bewertung der Kritikalität eines Szenarios im Bereich des automatisierten Fahrens (vgl. bspw. Westhofen et al.[128]), doch ist unklar, welche Metriken geeignet sind, um alle kritischen Szenarien zu identifizieren.

Im Projekt PEGASUS wurde ein Prozess für die Freigabe entwickelt, der sich im Wesentlichen auf die Identifikation und Durchführung von Szenarien stützt, die ein automatisiertes Fahrzeug bewältigen muss. Dieses szenariobasierte Testen zielt auf die Identifizierung äquivalenter Testfälle ab, die bei Durchführung des statistischen Testansatzes bzw. im realen Betrieb auftreten. Die als äquivalent identifizierten Testfälle werden als Szenarien bezeichnet, bei denen es sich um Sequenzen von zusammenhängenden Szenen handelt.[129] Die Beschreibung der Szenarien enthält Elemente der Umgebung, von denen angenommen wird,

---

[124] Wachenfeld, W.; Winner, H.: The Release of Autonomous Vehicles (2016), a: S. 442, b: S. 439-441.

[125] Die Autoren verfolgen die Annahme, dass ein hochautomatisiertes Fahrzeug nur halb so viele Unfälle mit Todesfolge wie ein Mensch auf deutschen Autobahnen verursachen sollte. Hieraus wird unter der Annahme einer Poisson-Verteilung von tödlichen Unfällen mit einem Signifikanzniveau von 5 % die angegebene Distanz von ca. 6 Milliarden Testkilometern ermittelt.[124b]

[126] Kalra, N.; Paddock, S. M.: Driving to safety (2016).

[127] Junietz, P. et al.: Approaches to Adress Safety Validation of Automated Driving (2018).

[128] Westhofen, L. et al.: Criticality Metrics for Automated Driving (2022).

[129] Ulbrich, S. et al.: Definition Szene, Situation und Szenario (2015).

dass sie für die Fahraufgabe relevant sind oder das Verhalten eines hochautomatisierten Fahrzeugs (HAF) beeinflussen können. Das szenariobasierte Testen ähnelt dem bereits etablierten Test von Anwendungsfällen (eng.: Use Cases). Allerdings berücksichtigt das szenariobasierte Testen eine höhere Anzahl und Variation von Parametern und zielt auf eine strukturiertere und standardisierte Ableitung von Testfällen für HAF ab. Dennoch ist die Identifikation der Szenarien, die für die Absicherung tatsächlich relevant sind, herausfordernd. Szenarien müssen identifiziert werden, die Kombinationen von Parametern enthalten, die das Verhalten des automatisierten Fahrzeugs maßgeblich beeinflussen. Die Ergebnisse des Forschungsprojekts PEGASUS[130a] stellen Quellen zur Identifikation von Szenarien für die Sicherheitsvalidierung von HAF vor. Diese können auf Wissen und auf Daten basieren. Die folgende Aufzählung nennt mögliche Quellen zur wissens- und datengestützten Szenarioidentifikation gemäß den angegebenen Publikationen:

Wissen

- Gesetze, Normen, Richtlinien[131]
- Beschreibung der Operational Design Domain (ODD)[132]
- Systematische Identifikation von Szenarien[133,134]

Daten

- Verkehrssimulation[135, 136]
- Feldbetrieb oder Verkehrsbeobachtung[137, 138]
- Simulatorstudien[130b, 139]
- Unfalldaten[140]

Bestehende Forschungsarbeiten heben hervor, dass auch mit Anwendung aller existierenden Verfahren zur Szenarioidentifikation nicht sichergestellt ist, sogenannte „Edge Cases“, d.h. selten vorkommende Szenarien, die zu einer kritischen Situation führen, aufzudecken.[141] Daher wird eine allgemeine Datenbank für Testszenarien als potentielle Lösung angesehen,

[130] Mazzega, J. et al.: Pegasus Method (2019), a: -, b: p. 13.

[131] Glatzki, F. et al.: Behavioral Attributes for a Behavior-Semantic Scenery Description (2021).

[132] Czarnecki, K.: Operational Design Domain for Automated Driving Systems (2018).

[133] Bagschik, G. et al.: Wissensbasierte Szenariengenerierung (2018).

[134] Weber, H. et al.: A framework for definition of logical scenarios (2019).

[135] Weber, N. et al.: Statistical approach for the derivation of scenarios (2020).

[136] Nalic, D. et al.: Software Framework for Testing of ADS (2021).

[137] Christian Rösener et al.: A Comprehensive Evaluation Approach for Highly Automated Driving (2017).

[138] Schachner, M. et al.: Observation-Based Pedestrian Scenario Extraction (2023).

[139] Li, H. et al.: Simulation with Vehicle-in-the-Loop Testbed (2023).

[140] Aydin, M.; Akbas, M. I.: Identification of Test Scenarios Using Accident Data (2021).

[141] Koopman, P. et al.: A Safety Standard Approach for Fully Autonomous Vehicles (2019).

um die Ungewissheit über unbekannte Szenarien wenigstens zu verringern.[142] Die Beziffe-rung über die Vollständigkeit einer Szenariendatenbank und die damit verbundene Abschät-zung des Restrisikos ist jedoch weiterhin eine offene Forschungsfrage.

Ein weiterer Ansatz ist die formale Verifikation, die einen formalen Beweis erbringen soll, der die Implementierung auf deren Einhaltung der Spezifikation prüft. Auf der Grundlage einer mathematischen Beschreibung bietet dieser Ansatz die Möglichkeit die Funktionsfä-higkeit z. B. für alle Werte und Wertekombination nachzuweisen. Aufgrund des hohen Auf-wands wird dies in der Regel nur für einfache Funktionen durchgeführt.[143] Shalev-Shwartz et al. [144] stellen dagegen einen Ansatz zur formalen Verifikation eines HAFs vor. Dieser beinhaltet die mathematische Beschreibung des grundlegenden Verhaltens, die ein Planer eines automatisierten Fahrzeugs immer befolgen soll. Bspw. soll zu jeder Zeit ein Sicher-heitsabstand zu anderen Objekten gehalten werden, sodass ein Manöver zur Kollisionsver-meidung immer möglich ist. Hierbei wird die Annahme getroffen, dass der Planer fähig ist diese Regeln immer zu befolgen. Darüber hinaus impliziert der Ansatz, dass alle relevanten Objekte in der Umgebung vom HAF korrekt wahrgenommen werden. Junietz et al.[145] kom-men zu dem Schluss, dass dieser Ansatz vor einer möglichen Einführung weitere Forschung erfordert. Auch Stahl[146] legt dar, dass die formale Verifikation bei Einsatz komplexer Funk-tionen oder von Algorithmen auf Basis maschineller Lernverfahren beschränkt einsetzbar ist. Pek[147] hingegen nutzt die formale Verifikation, um die Sicherheit einer Trajektorie nach-zuweisen. Dieser Ansatz stützt sich jedoch ebenfalls auf die Annahme, dass die Umfelder-kennung korrekte und vollständige Informationen liefert.

Der Ansatz von Pek kann auch beim sogenannten Silent Testing verwendet werden. Wang[148a] definiert Silent Testing als den Test einer Funktion in einem Fahrzeug, das "Sen-soreingaben erhält und dementsprechend Entscheidungen trifft, aber nicht in das Fahren des realen Fahrzeugs eingreift"[149]. Stattdessen wird das Fahrzeug beim Silent Testing von einem menschlichen Fahrer gesteuert. In seiner Dissertation entwickelt er einen Ansatz zum Ein-satz des Silent Testing zur Bewertung einer hochautomatisierten Fahrfunktion. Wang nutzt dabei virtuelle Instanzen der getesteten Planungsalgorithmen in den Fahrzeugen, um das fol-gende Verhalten auf Basis der realen Sensordaten zu simulieren.[148b] Dieser Ansatz verringert die Ungewissheit ggü. einem System, das mit jedem Zeitschritt eine Neuplanung auf Basis

[142] PEGASUS Projekt: PEGASUS Abschlussbericht (2020), S. 155.

[143] Luckcuck, M. et al.: Formal Specification and Verification of Autonomous Robotic Systems (2020).

[144] Shalev-Shwartz, S. et al.: On a Formal Model of Safe and Scalable Self-driving Cars (21.08.17).

[145] Junietz, P. et al.: Approaches to Adress Safety Validation of Automated Driving (2018).

[146] Stahl, T. N.: Dissertation, Safeguarding via online verification (2022), S. 19.

[147] Pek, C. et al.: Using online verification to prevent autonomous vehicles from causing accidents (2020).

[148] Wang, C.: Dissertation, Silent Testing for Safety Validation of Automated Driving (2021), a: p.31, b: p. 36.

[149] Eigene Übersetzung nach Wang[148a].

der aktuellen Position durchführt. Der Einfluss der Eingriffe des menschlichen Fahrers, der die tatsächliche Bewegung des Testfahrzeugs kontrolliert, wird somit verringert.

Sicherheitsargumentation

Das Projekt PEGASUS präsentiert als Übersicht der Sicherheitsargumentation eine Darstellung in der Goal Structuring Notation (GSN).[150] Hierbei werden Ziele basierend auf Vorschlägen der NHTSA zu wichtigen Sicherheitselementen (eng.: priority safety deisgn elements) als Basis der Sicherheitsargumentation hochautomatisierter Fahrzeuge übernommen, daraus Sicherheitsziele abgeleitet und in Teilziele heruntergebrochen. Diese werden mit den Teilschritten der PEGASUS Gesamtmethode zusammengeführt, d.h. einerseits werden weitere Teilziele abgeleitet, andererseits Lösungen (bzw. Nachweise) für die Ziele basierend auf Teilmethoden der PEGASUS Gesamtmethode vorgeschlagen. Die GSN ist lediglich als Beispiel für eine mögliche Sicherheitsargumentation deklariert. Die vorgestellte GSN ist daher unvollständig und kann nicht direkt für eine Sicherheitsargumentation für ein hochautomatisiertes Fahrzeug verwendet werden. Das Beispiel präsentiert jedoch den Nutzen der GSN, um die Zusammenhänge einer Argumentationskette übersichtlich darzustellen. Die Ziele sind jedoch noch auf hoher Ebene formuliert, sodass insbesondere die Detaillierung der Ziele und mögliche Lösungen noch zu entwickeln sind.

Eine solche Sicherheitsargumentation kann im besten Fall ein unvertretbares Risiko vermeiden. Allerdings verbleibt immer ein *Restrisiko*, dessen Quantifizierung mit Hilfe einer Argumentationskette, die zunächst aus qualitativen Argumenten besteht, herausfordernd bleibt. Die Bestimmung des Restrisikos ist nur retrospektiv durch statistische Verfahren möglich. Allerdings existiert bei Nutzung aller technischer Systeme ein *inhärentes Risiko*, das Teil des Restrisikos ist. Nach Maurer[151] ist das inhärente Risiko bei Einführung automatisierter Fahrzeuge nicht vollständig zu eliminieren. Das inhärente Risiko lässt sich durch eine Sicherheitsargumentation lediglich reduzieren. Wachenfeld[152] stellt als möglichen Ausweg die stufenweise Einführung automatisierter Fahrzeuge vor. Hierbei wird das Gesamtrisiko des Straßenverkehrs z. B. durch Beschränkung des Funktionsumfangs und der Limitierung der Stückzahlen automatisierter Fahrzeuge unter einer bestimmten Risikoschwelle gehalten.

Schnittstelle hochautomatisierter Fahrzeuge zur Umgebung

Die Schnittstelle eines Systems ist die zu seiner unmittelbaren Umgebung, die auf das System einwirkt. Aufgrund der Komplexität der relevanten Umgebung eines hochautomatisierten Fahrzeugs ist deren Spezifikation gegenüber Fahrzeugen ohne automatisierte Fahrfunktionen wesentlich umfangreicher. Als relevant werden all jene Faktoren betrachtet, die das Verhalten des automatisierten Fahrzeugs beeinflussen und solche die zur Bewertung notwendig sind, ob dieses Verhalten ausreichend sicher und regelkonform ist. Eine Validierung

[150] PEGASUS Projekt: PEGASUS Sicherheitsargumentation (2018).

[151] Maurer, M.: Das inhärente Risiko autonomer Straßenfahrzeuge (2018).

[152] Wachenfeld, W.: Dissertation, How Stochastic can Help to Introduce AD (2016), S. 163–164.

der Umgebungsspezifikation kann durch Systemtests (auf Fahrzeugebene) erfolgen, erfordert jedoch hohe Aufwände, da diese im realen Straßenverkehr durchzuführen sind. Daher werden neue Methoden entwickelt, um diese Tests zu reduzieren. SOTIF adressiert dies insbesondere mit der Bewertung des Restrisikos aufgrund unbekannter kritischer Szenarien.[153a] Zuvor fordert die Norm eine umfangreiche Analyse jeglicher Informationsquellen zur Identifikation der relevanten Einflussfaktoren, die mögliche Unzulänglichkeiten in der Spezifikation und auslösende Bedingungen für ein gefährliches Verhalten aufdecken. In den vorgeschlagenen Methoden ist ein starker Fokus auf die Umgebung erkennbar.[153b]

Für automatisierte Fahrzeuge wird der Begriff Operational Design Domain (ODD) für die Beschreibung der Umgebung verwendet. Nach ISO/FDIS 34503[154] definiert die ODD die Betriebsbedingungen, für die ein hochautomatisiertes Fahrzeug konstruiert ist. Gemäß PAS 1883:2020 beinhaltet die ODD Umweltbedingungen, Szenarien und dynamische Elemente.[155] Die möglichen Szenarien nach Ulbrich et al.[156] hängen daher von der Operational Design Domain (ODD) ab, für die das automatisierte Fahrzeug konzipiert ist. Czarnecki[157] beschreibt die ODD als Beschränkung des Umgebungsraums, der möglichen Zustände des Ego-Fahrzeugs sowie dessen Verhaltensraum. Als wesentlichen Teil der ODD beziehen sich Schuldt et al.[158] auf die Beschreibung von Szenarien mit Hilfe eines 4-Ebenen Modells. Bagschik et al. [159] erweitern dieses auf 5 Ebenen und von Scholtes et al. [160] auf 6 Ebenen. Zur Beschreibung der Zusammenhänge der Elemente einer ODD existiert bisher lediglich ein Konzeptpapier der Association for Standardisation of Automation and Measuring Systems (ASAM) zur Entwicklung des Standards OpenODD. Das Papier schlägt eine mögliche Sprache und ein Format inklusive einer Syntax zur Beschreibung der Zusammenhänge der ODD vor. Die enthaltenen Attribute sind vorwiegend der Norm PAS 1883:2020 und den Beschreibungen der Society of Automotive Engineers (SAE)[161] entnommen.[162]

---

[153] ISO: ISO/DIS 21448: Safety of the intended functionality (2022), a: S. 50–52, b: S. 31-36.

[154] ISO: ISO/FDIS 34503 - Specification for operational design domain (2023).

[155] BSI: PAS 1883:2020 (2020), S. 5.

[156] Ulbrich, S. et al.: Defining Scene, Situation, and Scenario (2015).

[157] Czarnecki, K.: Operational Design Domain for Automated Driving Systems (2018), S. 6.

[158] Schuldt, F. et al.: Systematische Testgenerierung für Fahrerassistenzsysteme (2013).

[159] Bagschik, G. et al.: Wissensbasierte Szenariengenerierung (2018).

[160] Scholtes, M. et al.: 6-Layer Model for a Description of Urban Environment (2021).

[161] SAE International: Best Practice for Describing an ODD (2020).

[162] ASAM e.V.: OpenODD: Concept Paper (2023).

Entwicklung neuer Normen

Zusätzlich zu den Normen ISO 26262 und SOTIF sind technische Berichte und Normen, die sich explizit mit der Sicherheit von hochautomatisierten Fahrzeugen (HAF) befassen, in Arbeit oder seit Kurzem veröffentlicht. So geben der technische Bericht ISO/TR 4804[163] und die Norm UL 4600[164] jeweils Empfehlungen und präsentieren mögliche Methoden zur Absicherung eines HAFs. Beide können zum Stand der Wissenschaft gezählt werden, da sie eher eine Sammlung bisheriger Erkenntnisse aus der Forschung in Form einer Norm darstellen. Beide Normen sind jedoch in stetiger Überarbeitung. Darüber hinaus ist nicht bekannt, inwieweit diese bisher zu einer erfolgreichen Freigabe eines hochautomatisierten Fahrzeugs beigetragen haben. Honda hat Anfang 2021 das Fahrzeug mit einer Level 3 Funktion für Teile von Japan freigeben können. Veröffentlichungen zu Freigabemechanismen beschränken sich auf die Zahl an realen Testkilometern (1,3 Millionen) oder die Anzahl simulierter Szenarien (10 Millionen).[165] Zusätzlich zur Einschränkung der ODD u. a. auf Japan, eine maximale Geschwindigkeit von 50 km/h und bestimmte Witterungsbedingungen hat Honda die Stückzahl anfangs jedoch auf 100 Exemplare beschränkt.[166] Dies folgt dem Konzept einer risikobeschränkten Einführung automatisierter Fahrzeuge nach Wachenfeld[167]. Das zweite bisher erfolgreich in Serie freigegebene System oberhalb von Level 2 ist der Drive Pilot (ein Level 3 System) von Mercedes, von dem ebenfalls keine detaillierten Informationen über den Freigabeprozess veröffentlicht sind. Die Freigabe des Systems folgt u. a. der UNECE R157[168] sowie SOTIF[169], die jedoch Teile des Freigabeprozesses dem Hersteller überlässt. Bekannt ist auch bei diesem System die Einschränkung der ODD u. a. auf deutschen Fernstraßen, eine maximale Geschwindigkeit von 60 km/h, das Bewegen innerhalb eines Fahrstreifens und bestimmte Witterungsbedingungen.[170, 171] BMW bietet ein vergleichbares Level 3 System zum Ende des Jahres 2023 zum Verkauf an. Die Zulassung des Systems wurde im September 2023 erlangt.[172] Darüber hinaus bieten bspw. die Automobilhersteller BMW und Ford Systeme in Serie an, die es ermöglichen die Hände vom Lenkrad zu

[163] ISO: ISO/TR 4804 - Safety and cybersecurity for automated driving systems (2020).

[164] UL: ANSI/UL 4600 - Voting Version.

[165] Conrad, B.: Marktstart für Honda Legend Hybrid EX (2020).

[166] Greimel, H.: Honda's Level 3 system for automated driving has limits (2022).

[167] Wachenfeld, W.: Dissertation, How Stochastic can Help to Introduce AD (2016), S. 102 ff.

[168] UNECE: UN-Regelung Nr. 157 - Genehmigung automatischer Spurhalteassistenzsysteme (2022).

[169] Mercedes-Benz Group AG: Introducing DRIVE PILOT (2023).

[170] van der Aalst, W.: Six Levels of Autonomous Process Execution Management (APEM) (2022).

[171] Carney, D.: New Mercedes self-driving system (2022).

[172] Greis, F.: Level 3 im Stau: BMW erhält Zulassung für hochautomatisiertes Fahren - Golem.de (2023).

nehmen, während das Fahrverhalten jedoch weiterhin vom menschlichen Fahrer zu überwachen ist. Solche Fahrfunktionen werden meist als Level 2+ bezeichnet.[173, 174]

Der technische Bericht ISO/TR 4804 verweist auf die Herausforderung, verschiedene Systemkonfigurationen und -varianten zu validieren. Dies ist vergleichbar mit einem modularen Ansatz zur Sicherheitsvalidierung. Allerdings beschränkt sich das dargestellte Verfahren auf den Vorschlag der Rückverfolgbarkeit. Regressionstests sind daher auf der Grundlage der Änderungen, z. B. veränderte Software-Codezeilen, zu konzipieren, die mit potentiell betroffenen Fähigkeiten verbunden sind.[175a] UL 4600 schreibt in ähnlicher Weise Regressionstests nach Änderungen vor, indem die Rückverfolgbarkeit genutzt wird, um die Anzahl der Tests zu reduzieren.[176] ISO/TR 4804 beschreibt außerdem Verifikations- und Validierungsverfahren für verschiedene Teilsysteme eines hochautomatisierten Fahrzeugs. Die Vorschläge tragen weitgehend zur Reduzierung von Fehlern bei der Implementierung dieser Teilsysteme bei. Sie verringern auch die Ungewissheit bezüglich ihrer Spezifikation, indem sie eine Grundspezifikation bereitstellen, die bei Entwicklung eines HAFs ergänzt werden kann. Allerdings sind die Ansätze nach wie vor auf Systemtests angewiesen. Es wird sogar empfohlen, Parameter (von der Systemebene) "auf höherer Systemebene zu testen, die einen sicherheitsrelevanten Einfluss auf die Ausgabe der Elemente haben".[177]

3.3 Alternative Perspektiven und Architektursichten

Während entsprechend dem Stand der Technik die Absicherung auf Systemebene erfolgen soll, zeigen aktuelle Forschungsarbeiten auch die Herausforderungen der Absicherung eines Systems innerhalb eines übergeordneten Systems als System-of-Systems. Eine Übertragbarkeit der dazu entwickelten Methoden auf Module scheint vielversprechend. Ein wesentlicher Unterschied zwischen Modulen und einem System besteht jedoch darin, dass die Abhängigkeiten zwischen Modulen größer sind als zwischen Systemen. Ein Modul benötigt in der Regel Informationen von einem anderen Modul, um seine Funktion bereitzustellen und die Anforderungen zu erfüllen. Daher schlussfolgert Grube Doiz[178], dass Module im Vergleich zu Systemen nicht unabhängig sind. Dies entspricht auch der Definition eines Moduls, dass

[173] La Rocco, N.: BMW 5er: Kraftfahrtbundesamt erteilt Genehmigungen für Level 2+ (2023).

[174] La Rocco, N.: BlueCruise: Ford bringt freihändiges Level 2+ nach Deutschland (2023).

[175] ISO: ISO/TR 4804 - Safety and cybersecurity for automated driving systems (2020), a: p. 55, b: p. 62.

[176] UL: ANSI/UL 4600 - Voting Version, S. 211–212.

[177] Eigene Übersetzung nach ISO: ISO/TR 4804 - Safety and cybersecurity for automated driving systems (2020), S. 62.

[178] Grube Doiz, N.: Bachelor Thesis, Modularisierung in der Automobilindustrie (2021), S. 53–56.

dieses „relativ“ unabhängig von anderen Modulen ist. Es verbleibt also immer eine Abhängigkeit, die im Vergleich lediglich geringer ist als zwischen nicht modular gestalteten Subsystemen. Bis auf die Unabhängigkeit erfüllen Module alle System-of-Systems Eigenschaften[178], wie sie z. B. von Nielsen et al. [179] beschrieben werden. Eine teilweise Übernahme von Methoden zur Absicherung von System-of-Systems erscheint daher möglich. Allerdings durchläuft ein übergeordnetes System-of-Systems in der Regel nicht so detaillierte Entwicklungsschritte und Analysen wie die einzelnen Systeme. Daher sind Abhängigkeiten zwischen den Systemen ggf. unbekannt. Anzahl und Stärke der Abhängigkeiten zwischen den Systemen sind allerdings auch geringer als zwischen Modulen eines Systems. Trotz der daraus resultierenden Ungewissheit des System-of-Systems werden bisherige Entwicklungsprozesse ohne eine solche detaillierte Betrachtung der Abhängigkeiten zwischen Systemen offenbar akzeptiert.

Forschungs- und Entwicklungsarbeiten zur Absicherung automatisierter Fahrzeuge zeigen, dass die Betrachtung des Verhaltens eines automatisierten Fahrzeugs notwendig ist.[180] Hieraus wird das Ziel einer Verhaltenssicherheit motiviert, die über die funktionale Sicherheit hinausgeht, da ein sicheres Verhalten auch die Prüfung der Spezifikation selbst erfordert. Die funktionale Sicherheit versucht dagegen vorwiegend die implementierte Funktion ggü. ihrer Spezifikation zu prüfen (vgl. Kapitel 2.4.1 zur SOTIF). Bagschik et al.[181] stellen daher die Fähigkeitssicht als zusätzliche Architektursicht vor. Diese beinhaltet einen Fertigkeitengraph, wie von Reschka et al.[182] beschrieben. Fertigkeiten dienen zur Beschreibung notwendiger Aktivitäten zur Erledigung einer Aufgabe eines technischen Systems.[183a] Über die Fertigkeiten lässt sich das Verhalten eines automatisierten Fahrzeugs mit Funktionen sowie Software- und Hardwarekomponenten verknüpfen bzw. rückverfolgen. Fertigkeiten lassen sich laut Reschka[183b] in Fähigkeiten eines automatisierten Fahrzeugs übersetzen. Fähigkeiten repräsentieren die Qualität einer Aktivität und sind ggü. Fertigkeiten bereits an die technische Umsetzung angepasst, damit ein Gütemaß für die Fähigkeit definiert werden kann.[183c] Fähigkeiten lassen sich daher ebenfalls Software- und Hardwarekomponenten sowie Funktionen zuordnen. Die Rückverfolgbarkeit bis hin zur Verhaltenssicht ist damit auch für unterschiedliche Güten an Fähigkeiten z. B. durch Verwendung unterschiedlicher Komponenten oder aufgrund von Degradationen möglich. Module, die direkt oder indirekt das Verhalten des automatisierten Fahrzeugs beeinflussen, müssen daher mit Fähigkeiten belegt

[179] Nielsen, C. B. et al.: Systems of Systems Engineering (2015), S. 10–12.

[180] WAYMO: Waymo Safety Report (2021).

[181] Bagschik, G. et al.: Architecture Framework for Safe Automated Vehicles (2018).

[182] Reschka, A. et al.: Ability and skill graphs for system modeling (2015).

[183] Reschka, A.: Dissertation, Fertigkeiten- und Fähigkeitengraphen von automatisierten Fahrzeugen (2017), a: S. 66, b: S. 67, c: S. 185.

und auf deren Erfüllung geprüft werden. Koopman[184] ergänzt dazu, dass nicht nur das Verhalten auf Basis der Bewegung des Fahrzeugs mit in die Betrachtung der Verhaltenssicherheit einzubeziehen ist. Beispielsweise kann die Kommunikation mit anderen Verkehrsteilnehmern (sei es über Funktechnologien oder externe Anzeigen am Fahrzeug) oder der reine Standort des Fahrzeugs, das damit ggf. Rettungsfahrzeuge blockiert, als sicherheitsrelevantes Verhalten betrachtet werden. Trotz der Hinweise auf notwendige Rückverfolgbarkeit zwischen den Architektursichten für eine Absicherung wird eine Zuordnung der einzelnen Sichten zu Modulen in bisherigen Forschungsarbeiten nicht berücksichtigt.

Das vom Bundesministerium für Wirtschaft und Klimaschutz geförderte Forschungsprojekt Verification and Validation Methods (VVM) setzt die in Unterkapitel 3.2 vorgestellten Forschungsarbeiten aus dem Projekt PEGASUS fort. Die bestehenden Methoden werden insbesondere in der detaillierten Umsetzung weiterentwickelt und auf eine urbane und damit komplexere ODD angewendet. Hierbei soll eine Sicherheitsargumentation basierend auf der Vollständigkeit der Spezifikation der ODD und deren Abdeckung in Tests erfolgen. Die Sicherheitsargumentation erfolgt dabei auf drei Ebenen, die in drei Entwicklungsschritten betrachtet werden. In der Konzepterstellung wird die Ebene der Fähigkeiten (eng: Capability Layer) betrachtet, die das Verhalten beschreibt, das zum Betrieb des automatisierten Fahrzeugs unter Abwesenheit unvertretbaren Risikos notwendig ist. Auf Basis der beschriebenen Fähigkeiten erfolgt während der Konstruktion und Realisierung die konkrete Spezifikation des Verhaltens auf der ingenieurstechnischen Ebene (eng: Engineering Layer). Hierbei wird die funktionale Architektur und die Software- und Hardwarearchitektur entwickelt. Die Realisierung wird darüber hinaus mit der physikalischen Konstruktion auf der dritten Ebene, der realen Welt (eng: Real World Layer) betrachtet. Während der Verifikation und Validierung werden ebenfalls alle drei Ebenen mit einbezogen. Auf der Fähigkeitsebene wird das Konzept zur Verifikation und Validierung definiert. Daraus folgt auf ingenieurstechnischer Ebene unter Einbezug der funktionalen und technischen Architektur die Definition von Testplänen. Auf der dritten Ebene der realen Welt schließt sich daraufhin die Testdurchführung an.[185] Abbildung 3-1 visualisiert den beschriebenen Prozess schematisch.

[184] Koopman, P.: How safe is safe enough? (2022), S. 74–79.

[185] Galbas, R.: VV-Methods comprehensive framework for AD safety assurance (2022).

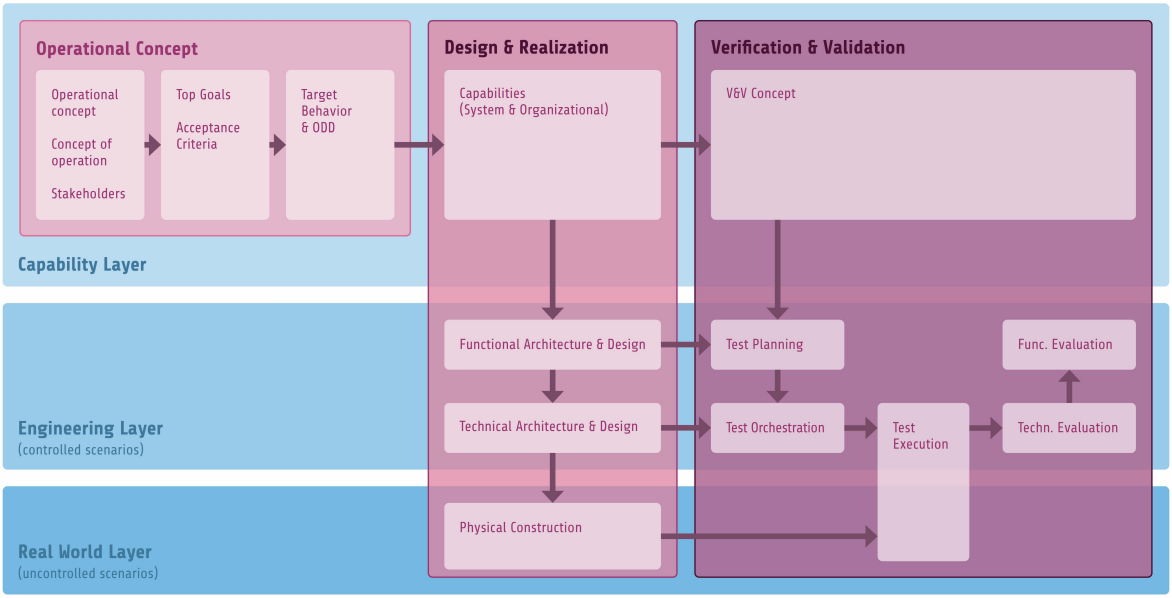


Abbildung 3-1: VVM Assurance Framework.[188]

Die in VVM beschriebenen Ebenen (eng: Layers) stellen jedoch keine Hierarchieebenen dar. Die Ebenen werden in derselben Veröffentlichung auch mit dem eindeutigeren Begriff Perspektiven bezeichnet (eng: Perspectives).[186] Perspektiven sind aufgrund ihrer ganzheitlichen Betrachtung des Entwicklungsprozesses für einen bestimmten Zweck wiederum von Sichtweisen abzugrenzen. Bagschik[187] verwendet den Begriff Sichtweisen bspw. zur Beschreibung von Architektursichten (eng: Architectural Viewpoints). Kapitel 2.2 beschreibt die eigentlich herrschende Unterscheidung zwischen einer Sichtweise und einer Architektursicht. Eine Sichtweise ist demnach eine der Architektursicht übergeordnete Betrachtung eines Systems.

Das Projekt VVM schlägt außerdem vor, für jede Ebene und jeden Entwicklungsschritt eine eigene Sicherheitsargumentation durchzuführen. Hierdurch soll die Übersichtlichkeit und damit auch die Vertrauenswürdigkeit verbessert werden.[188] Der Ansatz ist daher teilweise vergleichbar mit der Sicherheitsargumentation einzelner Module. Allerdings bleiben nach aktuellem Stand die Abhängigkeiten zwischen den individuellen Sicherheitsargumentationen unberücksichtigt.

Die Einführung der Perspektiven in VVM wird mit der notwendigen Rückverfolgbarkeit von verwendeten Methoden oder Nachweisen zu definierten Zielen in der Sicherheitsargumentation begründet.[186] Zu deren Umsetzung wird u. a. das Phänomen-Signal-Modell vorgestellt, das auf Basis von wahrnehmbaren Informationen aus der Systemumgebung, von Straßenverkehrsregeln sowie von Verhaltensregeln den weiteren Informationsfluss bis zur

[186] Galbas, R.: VV-Methods comprehensive framework for AD safety assurance (2022), S. 5.

[187] Bagschik, G. et al.: Architecture Framework for Safe Automated Vehicles (2018).

[188] Reich, J.: Assurance Argumentation Framework (2023).

Verhaltensentscheidung modelliert. Die damit einhergehende Formalisierung des Verhaltens unterstützt die Verhaltensentscheidung und die Bewertung des beobachtbaren Verhaltens bei der Verifikation und Validierung.[189] Trotz der starken Formalisierung und des direkten Bezugs auf wahrnehmbare Informationen sowie die Verhaltensentscheidung wird nur die Systemebene eines hochautomatisierten Fahrzeugs betrachtet. Dies hat den Vorteil das Verhalten unabhängig von technischen Umsetzungen ist. Allerdings bietet der derzeitige Stand nicht die Möglichkeit das Sollverhalten einzelner Module zu modellieren. Bisherige Veröffentlichungen beschreiben keine detaillierten Methoden, sodass die konkrete Umsetzung und Evaluation des Prozesses weiterhin Teil der Forschung sind.

3.4 Modulare Architekturen

Kapitel 2.2 stellt bereits grundlegende Eigenschaften modularer Architekturen vor und beschreibt Prinzipien, mit denen diese Eigenschaften erreichbar sind. Ergänzend dazu werden im folgenden Abschnitt Forschungsarbeiten zur Analyse der Nutzung modularer Architekturen aus ökonomischer Sicht dargestellt. Hierbei werden neue oder ungenutzte ökonomische Potentiale aufgezeigt, die sich durch eine konsequente Modularisierung ergeben. Anschließend werden Forschungsarbeiten analysiert, die bisherige Schwächen modularer Architekturen identifizieren und technische Lösungen zur Nutzung der offenen Potentiale vorstellen.

Ökonomische Sichtweise

Nach Baldwin und Clark[190] sind modulare Architekturen aufgrund der überschaubaren Komplexität ein beliebter Weg, um ein Produkt zu entwickeln. Weiter ist die parallele Entwicklung verschiedener Module möglich und die Ungewissheiten sind für zukünftige Entwicklungen geringer, da Module gegenüber Subsystemen ohne modulare Eigenschaften leichter optimiert, erweitert oder ersetzt werden können. Sie beziehen sich dabei auf die entwickelte Modularität in der Computerindustrie, die zu ihrer schnellen Entwicklung seit den 1990er Jahren beitrugen. Helper et al.[191a] weisen darauf hin, dass in der Automobilindustrie die Auslagerung von Modulen an spezialisierte Zulieferer ein Hauptantrieb für die Modularisierung ist. Sie stellen jedoch fest, dass modulare Architekturen auch Vorteile in Bezug auf Ergonomie sowie bei der Produktion bieten. Durch die verbesserte Möglichkeit individueller Tests der Module vor ihrer Integration sei außerdem ein Qualitätsvorteil erzielbar.[191b]

Doran[192] vertritt die Ansicht, dass Zulieferer als Modullieferanten selbst zu Fahrzeugherstellern werden können. Module müssen dafür relativ unabhängig von anderen Modulen und

[189] Beck, H. N. et al.: Phänomen-Signal-Modell: Formalismus, Graph und Anwendung (31.07.21).

[190] Baldwin, C. Y.; Clark, K. B.: Managing in an age of modularity (2008).

[191] Helper, S. et al.: Modularization and Outsourcing, a: - , b: p. 6.

[192] Doran, D.: Supply chain implications of modularization (2003).

ohne Unterstützung eines OEM sicher sein. Mondragon et al. [193] stellen fest, dass dies ihren Untersuchungen zufolge nur bei wenigen komplexen Technologien möglich sein könnte. In Bezug auf ihr Beispiel eines Brake-by-Wire-Moduls erklären sie, dass insbesondere komplexe Systeme OEMs als Haftungsträger benötigen. Sie untermauern diese Aussage damit, dass OEMs weniger bereit sind, Module in ihr System zu implementieren, in die sie selbst keinen Einblick haben.

Sanchez[194] präsentiert einen Überblick, wie die Automobilindustrie Modularität einsetzt. Dabei zeigt er auf, dass die eingesetzten Module die Anforderungen an eine modulare Architektur nicht erfüllen. Die Erkenntnisse von Sanchez zeigen, dass die Modularisierung in der Industrie nicht konsequent verfolgt wird. Integrations- und Systemtests sind somit zwingend erforderlich, da ohne eine konsequente Modularisierung, die Module nicht ausreichend unabhängig voneinander sein können. Sanchez weist insbesondere darauf hin, dass Schnittstellen im Entwicklungsprozess zu spät eingefroren werden (d.h. erst während der detaillierten Entwicklung der Module) und zu vage spezifiziert sind. Außerdem weist er darauf hin, dass eine Verbesserung der Schnittstellenspezifikation in neuen Produktgenerationen nicht durchgeführt wird. Zusätzlich fehle den Unternehmen eine geeignete Modularisierung der Organisationseinheiten und eine klare Strategie für deren Umsetzung.

Modularisierung von Software

Die zuvor beschriebenen Eigenschaften von Modularchitekturen und die Modulstrategien basieren auf der Perspektive mechanischer und mechatronischer Systeme (vgl. Kapitel 2.2). Die Eigenschaften und Strategien sind allerdings so beschrieben, dass sie auch auf modulare Systeme anderer Domänen anwendbar sind. Lipsmeier[195] und Göpfert[196] weisen z. B. darauf hin, dass ihre Beschreibungen von modularen Architekturen auch auf Softwarearchitekturen anwendbar sind. In der objektorientierten Programmierung werden Objekte als "operative Entität, die sowohl spezifische Datenwerte als auch den Code kapselt, der diese Werte manipuliert"[197] definiert. Lackes und Siepermann[198] definieren ein Modul im Kontext von Software als eine funktional geschlossene Einheit, die einen bestimmten Dienst bereitstellt. Dienstorientierte Architekturen greifen auf dieser Basis die offene Modularisierungsstrategie von Lipsmeier auf. Deren Dienste sind Softwarekomponenten, die die gleichen Eigenschaften wie Objekte bzw. Module haben. Dienste sind darüber hinaus lose gekoppelte Softwareteile und so konzipiert, dass sie während der Laufzeit integrierbar sind. Dies bietet eine noch höhere Flexibilität gegenüber der Objektorientierung. Darüber hinaus sind Dienste eng an

[193] Mondragon, C. C. et al.: Managing technology for complex systems (2009).

[194] Sanchez, R.: Building real modularity competence (2013), S. 205.

[195] Lipsmeier, A. et al.: Mechatronic Modularization of Intelligent Technical Systems (2018).

[196] Göpfert, J.: Modulare Produktentwicklung (2009), S. 192–203.

[197] Eigene Übersetzung von McGregor, J. D.; Sykes, D. A.: Testing object-oriented software (2001), S. 18.

[198] Lackes, R.; Siepermann, M.: Gabler Wirtschaftslexikon (2018).

die funktionale Architektur gekoppelt, da sie jeweils eine geschlossene Funktion bzw. einen Anwendungsfall darstellen.[199] Sie sind so gestaltet, dass andere Dienste einen Dienst bei Bedarf anfordern bzw. auf dessen Informationen zugreifen. Dabei können die Dienste in unterschiedlichen Programmiersprachen geschrieben werden und auf unterschiedlichen Hardwarekomponenten laufen. Die Kommunikation findet nur über die Schnittstellen statt, andere Daten und Zugriffsmöglichkeiten bleiben verborgen.[200]

Larses[201a] präsentiert ein Vorgehen zur Gruppierung von Modulen unter Berücksichtigung verschiedener Architektursichten. Er weist darauf hin, dass zur Modularisierung Komponenten mit starken Abhängigkeiten oder vergleichbaren Eigenschaften zu Modulen gruppiert werden sollten.[201b] Allerdings liefert er keine genauere Definition für solche Abhängigkeiten und Eigenschaften. Systemingenieure müssen zur Definition einer modularen Architektur entsprechend entscheiden, welche Abhängigkeiten und Eigenschaften stark oder ähnlich genug sind. Es wird sich daher auf Expertenwissen gestützt, das größeren Ungewissheiten und Schwankungen gegenüber einem vorgegebenen standardisierten Vorgehen unterliegt.

Die im Forschungsprojekt UNICAR*agil* entwickelte Automotive Service-Oriented Architecture (ASOA) nutzt eine dienstbasierte Architektur für den Einsatz in Automobilen sowie speziell in hochautomatisierten Fahrzeugen. ASOA besitzt die Eigenschaft, dass auch während der Laufzeit Dienste durch einen Orchestrator flexibel miteinander verknüpft werden können. Darüber hinaus können verschiedene Dienste auf unterschiedlicher Hardware mit unterschiedlichen Betriebssystemen und in unterschiedlicher Programmiersprache betrieben werden. Jeder Dienst wird außerdem so gestaltet, dass systemspezifische Informationen möglichst nicht benötigt werden. Stattdessen übernimmt ein Betriebsmodusmanagement anhand der Dienst-Informationen und einer Selbstwahrnehmung Entscheidungen, die abhängig von spezifischen Kombinationen verschiedener Dienste sind.[202] Dies erhöht die Modularität der Dienste bzw. ihre Flexibilität für den Einsatz in verschiedenen Systemen. Sicherheitsrelevante Entscheidungen und das daraus hervorgerufene Verhalten werden damit den Diensten abgenommen und in einer zentralen Instanz gesammelt. Die Dienste geben dabei jeweils Garantien, dass sie bestimmte Informationen zur Verfügung stellen. Im Forschungsprojekt UNICAR*agil* wird dies noch durch Qualität der Ausgaben eines Dienstes ergänzt, um dem Begriff einer Garantie gerecht zu werden. Falls ein Dienst eine Garantie nicht mehr oder zumindest nicht mehr im vollen Umfang leisten kann, wird dies über die Ausgabe in einem Qualitätsvektor kommuniziert. Die Selbstwahrnehmung aggregiert die Qualitätsdaten zu einer gesamten Leistungsfähigkeit des Systems.[203] Dies unterstützt ebenfalls die Modularität, da ein einzelner Dienst keine Kenntnisse über die Auswirkungen von Degradationen auf das

[199] The Open Group: Service-Oriented Architecture (SOA) (2007), S. 6.

[200] Bass, L. et al.: Software Architecture in Practice, 4th Edition (2021), S. 151-152.

[201] Larses, O.: Factors influencing dependable modular architectures (2005), a: -, b: S. 10.

[202] Jatzkowski, I. et al.: Vehicle Operating Mode Management (2021).

[203] Nolte, M. et al.: Summary and Taxonomy of Self-Representation Concepts (2020).

Systemverhalten haben muss. Allerdings ist dies weiterhin für die Selbstwahrnehmung und das Betriebsmodusmanagement erforderlich, sodass diese Kenntnisse über das Verhalten der Module unter verschiedenen Zuständen (wie bspw. Degradationen) benötigen.

3.5 Modulare Sicherheitsargumentation

Trotz der grundlegenden Neuheit eines ganzheitlich modularen Ansatzes in der Absicherung beschäftigt sich die Wissenschaft bereits mit Teillösungen, um die Verifikation und Sicherheitsargumentation modular zu gestalten.

Těsanović et al.[204] präsentieren ein Vorgehen, um die Gültigkeit der Verifikation einer Softwarekomponente trotz Änderungen zu erhalten. Die Autoren legen dar, wie z. B. das Zeitverhalten zwischen vorheriger und geänderter Komponente verglichen werden kann. Dementsprechend beschränkt sich das Verfahren auf messbare Größen, die korrekt spezifiziert wurden. Dies schließt eine Validierung und damit eine vollständige Sicherheitsargumentation nicht ein.

Rushby und Miner[205a] erörtern einen ähnlichen Ansatz zur modularen Zertifizierung im Bereich der Luftfahrt. Sie stellen eine formale Methode vor, um die Annahme-Garantie-Argumentation (eng: Assume-Guarantee-Reasoning) in Software zu beschreiben. Assume-Guarantee-Reasoning wird auch in dienstorientierten Architekturen verwendet. Dies bedeutet, dass ein Dienst A garantiert, Informationen mit einer bestimmten Qualität zu liefern. Dienst B arbeitet mit der Annahme, dass diese Garantie eingehalten wird, indem jeweils eine Annahme A(X2) bzw. A(X1) anstatt des eigentlichen Moduls eingesetzt wird. Das Prinzip ist in Abbildung 3-2 schematisch dargestellt. Die Ergebnisse der Einzelabnahmen werden superpositioniert, sodass eine Zertifizierung des Ganzen nicht mehr notwendig ist. Der Ansatz wird jedoch nur zur Verifikation, nicht aber zur Zertifizierung bzw. Validierung verwendet.[205b] Rushby und Miner weisen selbst darauf hin, dass weder ein Sicherheitsnachweis noch eine vollständige Argumentation der Sicherheit der einzelnen Funktionen, der Garantien und des fehlertoleranten Verhaltens (nach Stolte et al. [206] daher mindestens fail-safe) der Module ausreichend für eine Zertifizierung sind. Hierzu sei zusätzlich zwischen normalen und abnormalen Eigenschaften zu unterscheiden. Auf der Grundlage der abgeleiteten Maßnahmen zur Erreichung einer modularen Zertifizierung kommen Rushby und Miner[205c] zu dem Schluss, dass Komponenten weder eine starke Kopplung noch komplexe Abhängigkeiten aufweisen dürfen. Sie liefern jedoch keine weiteren Ansätze, um dieses Ziel zu erreichen.

---

[204] Těsanović, A. et al.: Modular Verification of Reconfigurable Components (2005).

[205] Rushby, J.; Miner, P. S.: Modular Certification (2002), a: -, b: p. 5, c: p. 20.

[206] Stolte, T. et al.: Taxonomy to Unify Fault Tolerance Regimes (2021).

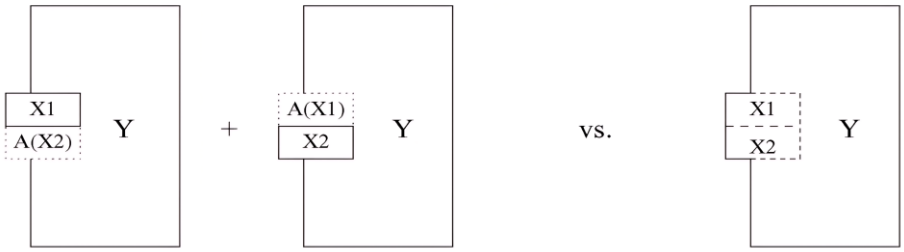


Abbildung 3-2: Prinzipdarstellung des Assume-Guarantee-Reasoning zur modularen Zertifizierung.

Driessen[207a] stellt eine Methode vor, in welcher er eine Modellierungssprache zur Modularisierung von Software beschreibt. Er betrachtet auch die Zertifizierbarkeit durch Einsatz von Verträgen, d.h. äquivalent zu der z. B. von Rushby und Miner genutzten Assume-Guarantee-Reasoning. Driessens Ziel ist ebenfalls Module innerhalb eines Softwaresystems zu ändern oder zu aktualisieren, ohne weitere Teile des Systems zu beeinträchtigen. Driessen beschreibt hierfür ein Vorgehen, um zu zeigen, dass Syntax und Semantik des Softwarecodes nach Änderungen des Moduls (d.h. z. B. des Softwarecodes) äquivalent sind. Die Betrachtung einer potentiell fehlerhaften oder unvollständigen Spezifikation fällt also nicht in den Anwendungsbereich dieses Ansatzes. Außerdem hängt der Ansatz von der vorherigen Zertifizierung des Gesamtsystems ab und erlaubt keine offene Modularisierung, ohne das Gesamtsystem erneut zu zertifizieren.[207b]

Neben der Verifikation einzelner Module oder der Kombination von Modulen unter Verwendung von Methoden der Compositional Safety (vgl. u. a. Sharvia[208]) existieren auch Ansätze zur modularen Argumentation von Sicherheit. Das Ziel ist bei Änderungen an einem System die Argumente zu identifizieren, die von den Änderungen betroffen sind. Hiermit lässt sich ggf. der Aufwand für Lösungen bzw. Nachweise der nicht betroffenen Argumente reduzieren. Ein mögliches Vorgehen zur Wiederherstellung der Sicherheitsargumentation nach Änderungen wird von Despotou und Kelly[209] vorgestellt. Sie zeigen, dass ein modularer Aufbau der Sicherheitsargumentation das Potential bietet, nach Änderungen am System nur die betroffenen Teile der Sicherheitsargumentation zu identifizieren und unabhängig von der restlichen Argumentation wiederherzustellen. Als Wiederherstellungsschritte schlagen sie die erneute Durchführung bestehender und betroffener Prozessschritte (z. B. die Durchführung von Tests) oder die Ergänzung der Sicherheitsargumentation durch zusätzliche Argumente an den betroffenen Stellen vor. Voraussetzung hierfür ist die mögliche Verknüpfung zwischen der Systemarchitektur und der Sicherheitsargumentation. Änderungen an einzelnen Modulen müssen äquivalent Argumente der Sicherheitsargumentation betreffen.

[207] Driessen, T.: Dissertation, Modularity by Design (2019), a: -, b: pp. 160-196.

[208] Sharvia, S.: Dissertation, Integrated Application of Compositional and Behavioural Safety Analysis (2011), S. 33–40.

[209] Despotou, G.; Kelly, T.: Argument modularity for safety assurance (2008).

Abbildung 3-3 zeigt schematisch eine Änderung am System, das einem Teil der Sicherheitsargumentation zugeordnet werden kann. Hierauf folgen Prozessschritte zur Analyse von Möglichkeiten die Argumentation wiederherzustellen, z. B. durch zusätzliche unterstützende Argumente. Die Autoren weisen jedoch darauf hin, dass Verknüpfungen zwischen niedrigeren Ebenen eines Systems zu Gefahren auf Systemebene oft unklar sind. Eine Nachverfolgung der Auswirkung von Änderungen auf die Sicherheitsargumentation wäre in diesem Fall nicht möglich. Für automatisierte Fahrzeuge kann ergänzend die Problematik genannt werden, dass die Sicherheitsargumentation derzeit stark auf Nachweise durch Systemtests basiert. Die Norm SOTIF[210] demonstriert, wie sich theoretisch der Testaufwand durch Aufteilung der Funktionen in Module auch für einen statistischen Testansatz reduziert. Allerdings wird als notwendige Voraussetzung eine ausreichende Unabhängigkeit zwischen den Modulen genannt. Konkretere Anforderungen an die Module oder Methoden, um diese Unabhängigkeit zu erreichen oder zu argumentieren fehlen. Der statistische Ansatz bietet daher bisher keine Möglichkeit zur Verknüpfung zwischen dem Argument eines statistischen Nachweises der Sicherheit und Komponenten des Systems. Eine ähnliche Problematik ergibt sich auch beim szenariobasierten Ansatz zur Absicherung automatisierter Fahrzeuge. Teils können Szenarien expliziter Nachweis von Argumenten sein, die sich auf einzelne Module beziehen. Allerdings werden bisher überwiegend Szenariendatenbanken entwickelt, die eine Sicherheitsargumentation zum Verhalten des Fahrzeugs und damit auf Systemebene erbringen.[211] Eine Verknüpfung zu einzelnen Modulen fehlt hierbei. Dies bedeutet, dass die Sicherheitsargumentation mit den bisherigen Ansätzen zur Absicherung automatisierter Fahrzeuge nicht vollständig modular gestaltet werden kann, wie von Despotou und Kelly vorgeschlagen.

[210] ISO: ISO/DIS 21448: Safety of the intended functionality (2022), S. 149–153.

[211] PEGASUS Projekt: PEGASUS Abschlussbericht (2020), S. 32.

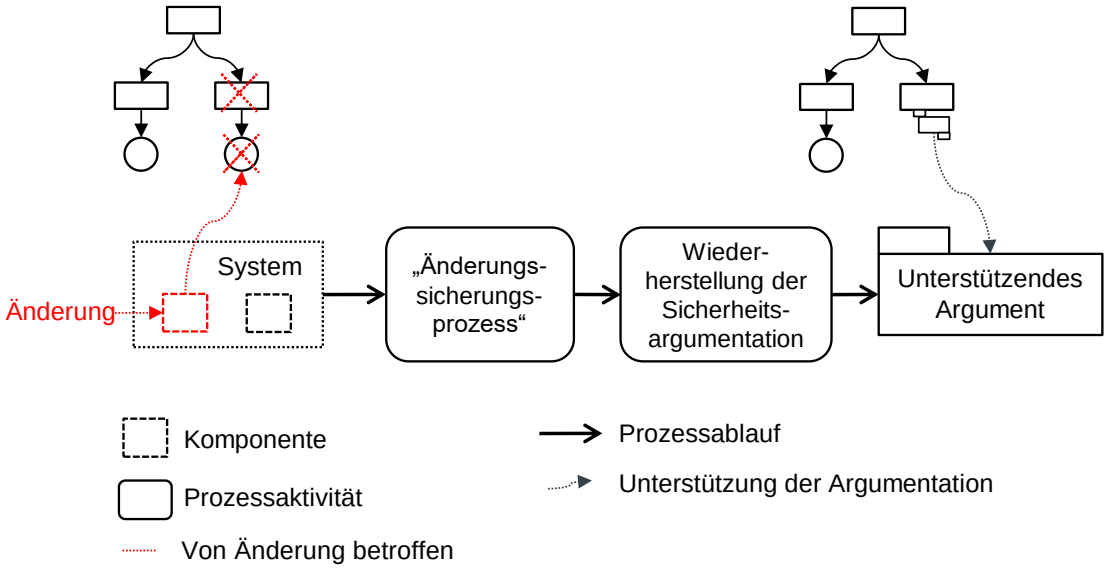


Abbildung 3-3: Wiederherstellung der Sicherheitsargumentation nach Änderung am System nach Despotou und Kelly.[212]

Amersbach und Winner[213] stellen den Ansatz einer funktionalen Dekomposition vor. Sie präsentieren eine Zerlegung der Fahrfunktion in sechs funktionale Dekompositionsebenen, die detaillierter ist als die weit verbreitete Zerlegung in „Sense, Plan und Act“[214]. Die Dekomposition der Fahraufgabe wird verwendet, um den Parameterraum für die Absicherung der einzelnen funktionaler Ebenen zu reduzieren, anstatt den gesamten Parameterraum für alle Funktionsschichten zusammen zu betrachten und zu variieren. Parametervariationen, die ursprünglich zu Testfällen führen würden, bei denen sich die variierten Parameter nicht auf die betrachtete funktionale Schicht auswirken, werden durch den Ansatz der funktionalen Dekomposition eliminiert. Die Autoren zeigen, dass sich dadurch auch der Gesamtaufwand der Absicherung reduziert. Der Ansatz erfordert jedoch die Bewertung des Einflusses von Parametern aus Systemszenarien (im Falle der Verwendung eines szenariobasierten Ansatzes) auf die funktionalen Ebenen. Bislang stützt sich diese Bewertung auf Expertenwissen.[215] Die Untersuchung einer regelbasierten Bewertung der Einflussparameter hat sich jedoch als nicht erfolgversprechend in Bezug auf eine signifikante Reduktion des Parameterraums erwiesen.[216]

[212] Eigene Abbildung übersetzt nach Despotou, G.; Kelly, T.: Argument modularity for safety assurance (2008).

[213] Amersbach, C.; Winner, H.: Functional Decomposition to Reduce Approval Effort (2017).

[214] ISO: ISO/TR 4804 - Safety and cybersecurity for automated driving systems (2020), S. 22.

[215] Amersbach, C.: Dissertation, Functional Decomposition Approach (2020), S. 72.

[216] Bickel, J.: Masterthesis, Identifikation und Zuordnung von Einflussparametern (2020), S. 79–81.

Die Zerlegung des Systems in Module impliziert ebenfalls eine funktionale Dekomposition. Allerdings beinhaltet ein Modul nicht unbedingt nur Funktionen einer einzelnen funktionalen Ebene. Die bspw. von Göpfert[217] beschriebenen Anforderungen an eine modulare Architektur führen zu Modulen, die mehr als eine funktionale Ebene abdecken. Der Stand der Technik bietet zum Beispiel keine Metriken zur Bewertung der Korrektheit von Sensor-Rohdaten, d.h. den Ausgaben der funktionalen Dekompositionsebene Informationsaufnahme. Die Absicherung eines Sensormoduls erfordert daher mindestens den Einbezug der Ebene Informationsverarbeitung, sodass deren Ausgabe z.B. in Form von Objektlisten bewertbar ist. Ein Sensormodul repräsentiert also bereits die ersten beiden funktionalen Dekompositionsebenen. Darüber hinaus beinhalten diese Ebenen nicht nur ein Sensormodul für die Umgebungswahrnehmung, sondern auch von Sensoren anderer Module. Ein automatisiertes Fahrzeug benötigt z. B. eine Schätzung seines Fahrzustands, z. B. durch ein Lokalisierungsmodul, während die Motoren (als potentielle Module) die Messung ihrer aktuellen Drehzahl erfordern.

3.6 Einordnung des Forschungsvorhabens in die Literatur

Aktuelle Forschungsarbeiten zeigen mögliche Methoden zur Absicherung hochautomatisierte Fahrzeuge. Neue Normen greifen diese Methoden auf, befinden sich aber noch in einer frühen Phase, sodass deren vollständige Anwendung und Bewährung in der Praxis bisher nicht bekannt ist. Die bisher weiterhin verfolgte Absicherungsmethode für automatisierte Fahrfunktionen ist der statistische Test bzw. Testfahrten über Millionen von Kilometern mit dem Zielsystem.[218] Mit der Erforschung neuer Methoden wird dieser aktuell versucht durch einen szenariobasierten Testansatz zu ersetzen. Es zeigt sich, dass sowohl die wissensbasierten als auch die datenbasierten Absicherungsmethoden zur Ermittlung von Testszenarien ebenso Teilfunktionen eines hochautomatisierten Fahrzeugs betrachten. Amersbach und Winner[219] stellen ergänzend die funktionale Dekomposition vor, die die automatisierte Fahrfunktion in Teilfunktionen (sogenannte funktionale Dekompositionsebenen) zerlegt, mit der sich Szenarien von der Systemebene vereinfachen lassen. Allerdings fehlt es dabei an einer Argumentation zur Darlegung der Unabhängigkeit dieser funktionalen Ebenen bzw. Module. Das Verhalten anderer Module kann das zu testende Modul auf unerwartete Weise stimulieren. Diese Stimulation muss aufgrund von Abhängigkeiten zwischen Modulen für den Ansatz einer modularen Absicherung genauer analysiert werden. Der Ansatz einer expertenbasierten Bewertung der Abhängigkeiten zwischen funktionalen Ebenen ergibt nur sehr

[217] Göpfert, J.: Modulare Produktentwicklung (2009), S. 38.

[218] Mercedes-Benz Group AG: Introducing DRIVE PILOT (2023), S. 36–44.

[219] Amersbach, C.; Winner, H.: Functional Decomposition to Reduce Approval Effort (2017).

beschränkte Vorteile für die funktionale Dekomposition bei weiterhin vergleichsweiser hoher Ungewissheit.[220]

Noch herausfordernder kann die Sicherstellung ausreichender Unabhängigkeit bei der technischen Umsetzung sein, da Komponenten untereinander teilweise stark vernetzt sind. Hierbei helfen die Ansätze neuer Entwicklungen modularer Architekturen, die eine relative Unabhängigkeit von Softwarekomponenten in Form von Diensten anstreben. Die grundlegende Funktionalität dieser Architekturen zur flexiblen Vernetzung und Kombination von Diensten wird in Forschungsarbeiten adressiert. Eine mögliche Absicherung einzelner Dienste oder Module wird dabei jedoch nicht betrachtet. Absicherungsvorhaben beschreiben daher zwar ebenfalls Tests der einzelnen Funktionen oder Dienste, analysieren jedoch nicht die Auswirkungen der Abhängigkeiten in einer dienstbasierten Architektur. Bestehende Verfahren verfolgen lediglich die Verifikation einzelner Module, wodurch die Validierung für die Sicherheitsargumentation weiterhin unberücksichtigt bleibt.

Die notwendige Betrachtung von Abhängigkeiten findet dagegen in Forschungsprojekten zur Absicherung hochautomatisierter Fahrzeuge wie bspw. dem Projekt VVM Anwendung.[221] Hierbei werden insbesondere neue Architektursichten zur Verhaltensbeschreibung in Form von Fähigkeiten eines hochautomatisierten Fahrzeugs eingeführt. Hieraus sollen sich induktiv sichere Verhaltensweisen und deduktiv Anforderungen an die Komponenten ableiten lassen. Einflüsse der technischen Implementierung und den daraus implizierten Abhängigkeiten zwischen Modulen sind in den bestehenden Forschungsarbeiten jedoch trotz systematischer Modellierung automatisierter Fahrfunktionen nicht adressiert. Stattdessen liegt der Fokus auf der Betrachtung der Systemschnittstelle. Für diese wird in einer ODD die Einsatzumgebung eines automatisierten Fahrzeugs beschrieben. Diese Informationen werden zusammen mit wissens- und datenbasierten Methoden genutzt, um Szenarien auf der Systemebene abzuleiten.

3.7 Resultierende Forschungsfragen

Die Arbeiten aus Forschungsarbeiten zur Absicherung hochautomatisierter Fahrzeuge zeigen, dass die Betrachtung einzelner Module zwar stattfindet, aber weiterhin von einer Absicherung auf Systemebene ausgegangen wird. Amersbach und Winner[222] zeigen jedoch, dass die Betrachtung einzelner Entitäten eines Systems enormes Potential für die Testaufwandsreduktion bietet. Zudem werden die Entwicklungen und Forschungsarbeiten zu dienstbasierten Architekturen ebenfalls dadurch motiviert, dass für immer stärker vernetzte hochautomatisierte Fahrzeuge eine Updatefähigkeit und Wiederverwendbarkeit von Diensten bzw.

[220] Bickel, J.: Masterthesis, Identifikation und Zuordnung von Einflussparametern (2020), S. 79–81.

[221] Reich, J.; Nolte, M.: VVM Assurance Argumentation (2022).

[222] Amersbach, C.; Winner, H.: Functional Decomposition to Reduce Approval Effort (2017).

Modulen sichergestellt werden muss. Hiermit wird den Anforderungen an die Sicherheit und dem enormen Kostendruck bei der Entwicklung solch komplexer Systeme begegnet.

Während die Industrie mit Normen und etablierten Prozessen die Abhängigkeiten in einem System analysieren und Komponenten bereits früh ausführlich testen, zeigt Sanchez[223], dass dessen Umsetzung nicht für eine konsequent modulare Architektur ausreicht. Existierende Forschungsarbeiten zur Absicherung automatisierter Fahrzeuge widmen sich darüber hinaus zwar intensiv der Verhaltenssicherheit des Fahrzeugsystems, adressieren jedoch nicht das Verhalten einzelner Module. Gleichzeitig existieren bei Anwendung bestehender Entwicklungsprozesse Ungewissheiten über die Abhängigkeiten zwischen Systemen in einem System-of-Systems. Ein daraus resultierendes Restrisiko wird allerdings akzeptiert.

Daher werden in der folgenden Arbeit Möglichkeiten analysiert die Sicherheit eines hochautomatisierten Fahrzeugs durch Betrachtung der einzelnen Module zu argumentieren. Die Sicherheitsargumentation soll dabei mindestens das Niveau der konventionellen Absicherung erreichen, d.h. kein höheres Restrisiko verursachen. Hieraus leitet sich die folgende erste Forschungsfrage ab:

1. Wie lässt sich die Sicherheit einzelner Module eines hochautomatisierten Fahrzeugs äquivalent zum System argumentieren?

Zur Beantwortung der ersten Forschungsfrage werden Schwächen bestehender Prozesse und Methoden identifiziert, die eine modulare Absicherung behindern. Daraus folgt die zweite Forschungsfrage, wie sich diese Schwächen mit neuen Ansätzen überwinden lassen:

2. Wie können neue Ansätze die Sicherheitsargumentation hochautomatisierter Fahrzeuge auf Modulebene unterstützen?

Auf Basis der Forschungsfragen wird in Abbildung 3-4 die Struktur der Arbeit dargestellt. Zur Beantwortung der ersten Forschungsfrage dient zunächst die bereits vorgestellte Recherche und Analyse bestehender Methoden nach dem Stand der Technik und Wissenschaft zur Absicherung hochautomatisierter Fahrzeuge und Verwendung modularer Systeme. Darauffolgend wird in Kapitel 4 eine Argumentationskette aufgebaut, die Strategien präsentiert, die eine modulare Absicherung ermöglichen. Hieraus ergeben sich Ziele, die zur Umsetzung einer modularen Absicherung zu erreichen sind. Den Zielen werden daraufhin Lösungen nach dem Stand der Technik und Forschung zugeordnet. Zur Beantwortung der zweiten Forschungsfrage werden daraufhin in Kapitel 4.6 neue Lösungsvorschläge entwickelt, die noch ungelöste Ziele der Argumentationskette unterstützen. In Kapitel 5 wird daraufhin als Kernlösung der vorliegenden Arbeit die detaillierte semantische Schnittstellenbeschreibung zur Unterstützung einer evolutionären Entwicklung vorgestellt. Kapitel 6 demonstriert die vorgestellten Lösungen an einem Prototyp eines hochautomatisierten Fahrzeugs aus dem Forschungsprojekt UNICAR*agil*.

[223] Sanchez, R.: Building real modularity competence (2013).

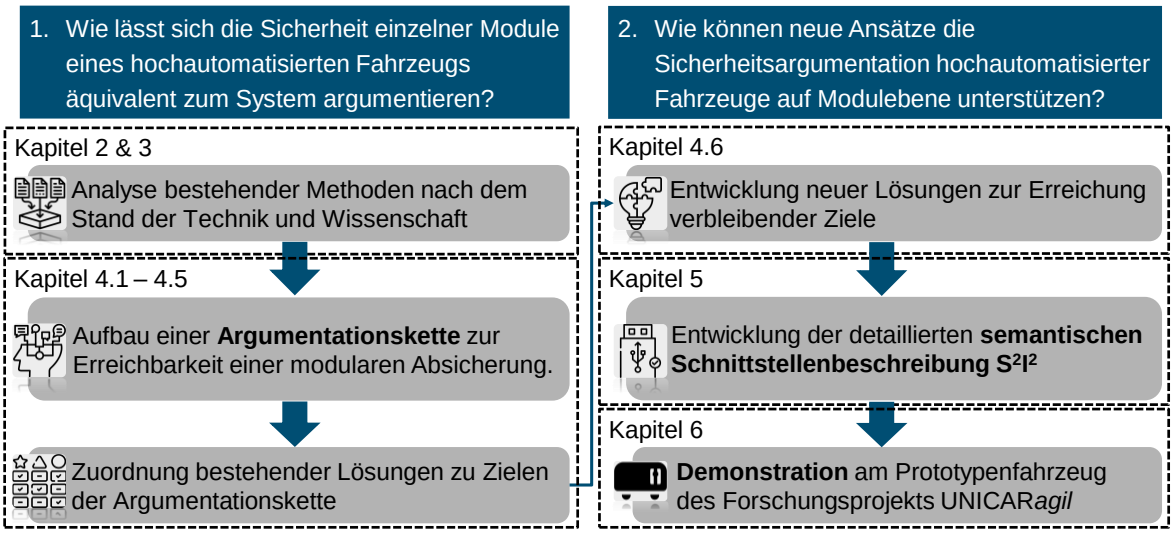


Abbildung 3-4: Forschungsfragen zugeordnet zu Schritten der Methodik der vorliegenden Arbeit unter Angabe der jeweiligen Kapitel. Gestrichelte Linien kennzeichnen die Teile der Methodik, die zu angegebenen Kapiteln gehören. Pfeile stellen die inhaltliche Abfolge dar.

4 Argumentation für eine modulare Absicherung

Ausgehend vom Ziel Module individuell abzusichern, sodass ein hochautomatisiertes Fahrzeug insgesamt abgesichert ist, wird im folgenden Kapitel eine Argumentationskette hergeleitet. Die Herleitung erfolgt auf Basis der identifizierten Lücken im Stand der Technik und bestehender Ansätze der Forschung zu hochautomatisierten Fahrzeugen, modularere Architekturen sowie deren Absicherung. Die Argumentationskette enthält daher nur zu Teilen bereits bestehende Methoden und Prozesse, die zur Absicherung von Modulen beitragen. Die entwickelte Argumentationskette stellt keine Vorlage für die vollständige Umsetzung einer modularen Absicherung dar, kann jedoch als Grundlage für eine solche Sicherheitsargumentation dienen.

Aufgrund der Vielfalt an möglichen Lösungen und Informationsquellen verfolgt die Argumentation einen wissensbasierten Ansatz zur Absicherung von Modulen. Hinsichtlich der Komplexität eines hochautomatisierten Fahrzeugs und dessen Umgebung ist grundsätzlich eine kaum quantifizierbare Unvollständigkeit dieses Ansatzes zu erwarten. Ein datenbasierter Ansatz kann dies ergänzen, benötigt jedoch den wissensbasierten Ansatz als Grundlage zur Identifikation der hierzu erforderlichen Informationen.

Das erste Unterkapitel 4.1 leitet zunächst die genutzte Darstellung der Argumentationskette in einem wissenschaftlichen Kontext her. Auf Basis einer Argumentation unter Verwendung der Goal Structuring Notation (GSN) wird dargelegt, dass es einer zusätzlichen Betrachtung von Ungewissheiten in einer Argumentationskette bedarf. Daraufhin entwickelt Kapitel 4.2 eine Argumentation, dass die Umgebung von Modulen und die Interaktionen mit dieser äquivalent zu Systemen beschreibbar ist. Hierzu dient insbesondere die Strategie zur Beherrschung der Systemkomplexität mit Hilfe der hergeleiteten Komplexitätseigenschaften und daraus folgenden Maßnahmen bzw. Zielen zur Reduktion der Komplexität. Kapitel 4.3 erläutert Argumente zur Validität und Vollständigkeit bestandener Modultests. Für das Bestehen von Modultests wird darin auf etablierte Entwicklungsprozesse und zur Argumentation der Validität von Testumgebungen für hochautomatisierte Fahrzeuge auf aktuelle Forschungsprojekte verwiesen. Zur Argumentation der Vollständigkeit von Modultests wird einerseits die Übertragbarkeit von Elementen einer Item Definition für hochautomatisierte Fahrzeuge auf Module erläutert und andererseits Informationen identifiziert, die zur Generierung von Modultests mit Methoden nach dem Stand der Wissenschaft und Technik notwendig sind. Die Pfade der Argumentationskette nach Kapitel 4.2 und Kapitel 4.3 offenbaren beide an mehreren Stellen weiterhin bestehende Ungewissheiten über die Vollständigkeit und Korrektheit von Modulspezifikationen, Modulumgebung und der Modellierung von Abhängigkeiten zwischen Modulen auf Basis von Informationen, die von der Systemebene abgeleitet werden. Daher werden in Kapitel 4.4 Argumente zur Vermeidung von Irrtümern beim dafür notwendigen Dekompositionsprozess entwickelt. Hierzu werden potentielle Irr-

tümer durch eine rückwärtsgerichtete Betrachtung, ausgehend von nicht abgesicherten Modulen identifiziert. Diese dienen zur Definition von Zielen zur Vermeidung solcher Irrtümer. Kapitel 4.5 liefert daraufhin eine zusammenfassende Übersicht über existierende Lösungen des Stands der Wissenschaft und Technik für die in den vorigen Kapiteln identifizierten Ziele der Argumentationskette. Dabei werden Ziele aufgezeigt, die mit den existierenden Methoden nicht vollständig lösbar sind. In Kapitel 4.6 werden neue Ansätze entwickelt, die diese ungelösten Ziele unterstützen. Zuerst wird dabei zur eindeutigen Zuordnung von Informationen zu Modulen das Gestaltungsprinzip der semantischen Äquivalenz zwischen Architektursichten vorgestellt. Als mögliche Maßnahme zur Unterstützung der Definition von Testumgebungen für Module wird eine Methode zur argumentativen Testumgebungsreduktion beschrieben. Der Großteil ungelöster Ziele der Argumentationskette betrifft die Vermeidung von Irrtümern im Dekompositionsprozess. Darauf aufbauend wird das Potential einer evolutionären Entwicklung vorgestellt und die damit einhergehende Notwendigkeit von standardisierten detailliert beschriebenen Schnittstellen dargelegt. Dies bildet die Grundlage für die detaillierte semantische Schnittstellenbeschreibung, die in Kapitel 5 entwickelt wird. Das beschriebene Vorgehen zur Herleitung der Argumentationskette für eine modulare Absicherung in Kapitel 4 ist in Abbildung 4-1 grafisch dargestellt.

Die Strategien der Argumentationskette werden kontinuierlich durch Beispiele aus dem Bereich hochautomatisierter Fahrzeuge unterstützt. Zur klaren Trennung der Beispiele von der eigentlichen Argumentationskette sind die Beispiele durch graue Kästen hervorgehoben und zur Referenzierung durchnummeriert.

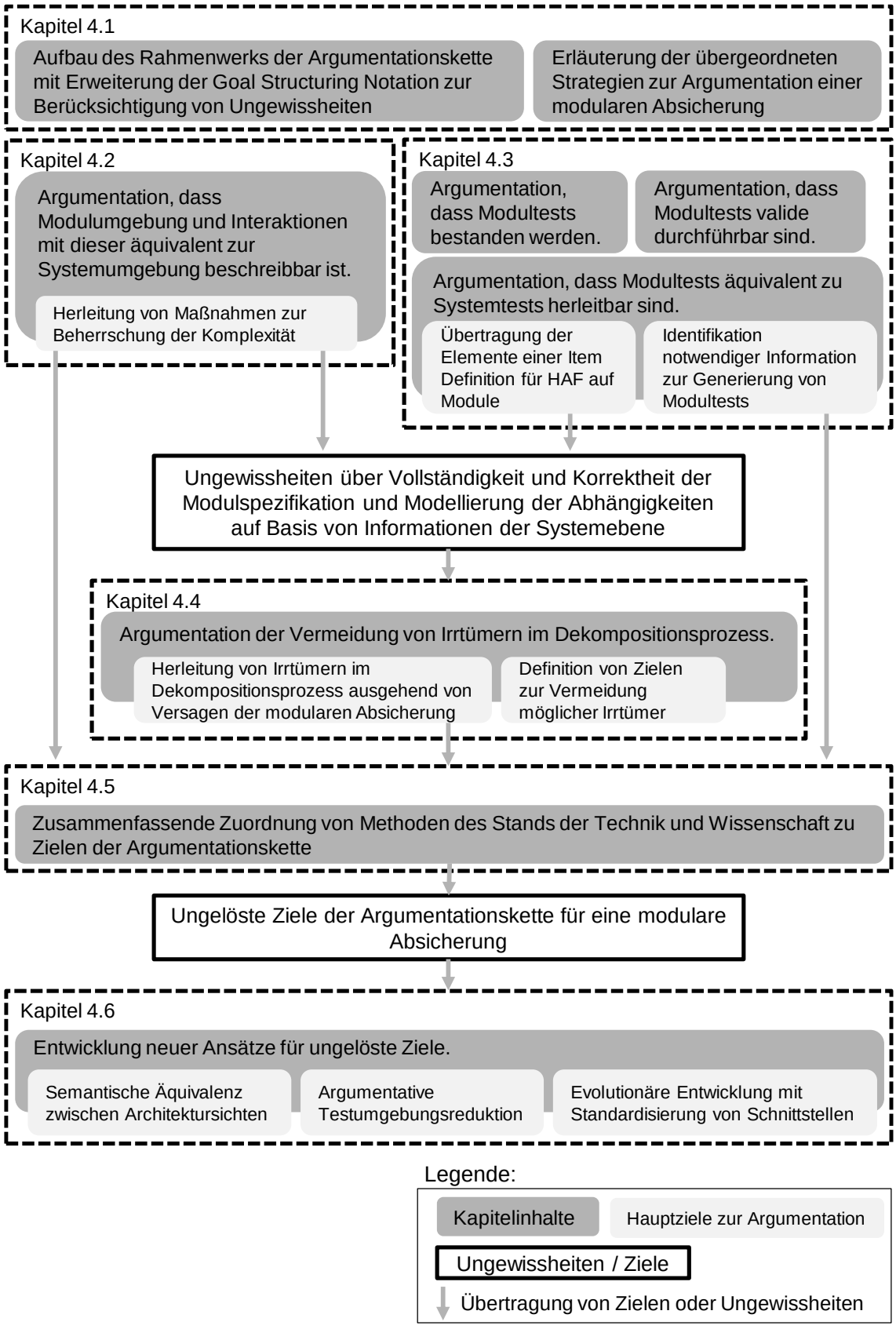


Abbildung 4-1: Struktur der Inhalte in Kapitel 4 auf Basis der Argumentationskette.

4.1 Aufbau der Argumentationskette

Entlang der Argumentationskette werden Strategien und Ziele identifiziert, die zur Erreichung des jeweils übergeordneten Ziels notwendig sind. Die Argumentationskette wird mit Hilfe der in Kapitel 2.4.2 vorgestellten Goal Structuring Notation (GSN) grafisch und logisch unterstützt. Die GSN hat in bisheriger Form das Ziel eine (Sicherheits-)Argumentation zu erbringen. Dies ist auch in der Notation der GSN manifestiert, die lediglich Elemente zur Bekräftigung der Erreichung einzelner Ziele beinhaltet. Popper[224a] kritisiert in seiner Arbeit „Logik der Forschung“ dagegen die zur Zeit der Veröffentlichung in der Wissenschaft vorherrschende Sicht des logischen Positivismus. Die Wissenschaftstheorie nach Popper[224b] versucht stattdessen Ziele bzw. Hypothesen zu entkräften bzw. zu falsifizieren, da die empirische Wissenschaft nicht die Möglichkeit hat, Nachweise für die Gültigkeit einer Hypothese zu erbringen. Grund hierfür ist, dass die Anzahl der positiven Versuche zur Bestätigung der Hypothese begrenzt ist, während bereits ein einziger gegenteiliger Versuchsausgang die Hypothese widerlegt. Die Aufstellung von Hypothesen bzw. Zielen hat trotzdem weiterhin Berechtigung, um zumindest wahrscheinliche Vorhersagen treffen zu können. Die Hypothesen bzw. Ziele sollten jedoch falsifizierbar formuliert werden. In der GSN ist dies auf den oberen Ebenen ggf. nicht gegeben. Die Zerlegung in detailliertere Ziele, die einer logischen Argumentationskette folgen, bietet jedoch das Potential einer falsifizierbaren Formulierung.

Die Zerlegung der Ziele hat den Nachteil, dass diese lediglich durch logische Argumentation begründbar ist. In der GSN wird daher die Beschreibung von Strategien verwendet. Bei Bedarf werden auch der Kontext und getroffene Annahmen für ein Ziel angegeben. Es existieren in der GSN jedoch keine Notationselemente, um gegenteilig darzustellen, inwieweit bei der Zerlegung bzw. den getroffenen Strategien Ungewissheiten ggü. der Erreichung der Ziele bestehen. Daher wird in dieser Arbeit das zusätzliche Notationselement „Ungewissheiten“ (eng: „Uncertainties“) eingeführt. Entgegen der ursprünglichen Vorgehensweise der GSN möglichst positiv Ziele (G), Strategien (S) und Lösungen (Sn) zur Erreichung des übergeordneten Ziels darzustellen, macht die Darstellung bekannter Ungewissheiten die Schwachstellen der Argumentationskette sichtbar. Hieraus ergeben sich mindestens zwei neue Möglichkeiten: Erstens lässt sich anhand der verbleibenden Ungewissheiten einschätzen, welches Restrisiko z. B. durch Nutzung eines technischen Systems verbleibt. Das von Maurer[225] beschriebene nicht eliminierbare, inhärente Risiko bei Einführung hochautomatisierter Fahrzeuge, regt äquivalent eine solche Risikokommunikation an und bietet das Potential Vertrauen zwischen Herstellern und Nutzern oder Zertifizierungsbehörden zu schaffen. Zweitens hat die Angabe verbleibender Ungewissheiten das Potential selbige durch Identifikation weiterer Maßnahmen (wie bspw. die Nutzung datengetriebener Ansätze) in Form zu-

[224] Keuth, H.: Karl Poppers „Logik der Forschung“ (2019), a: S. 49–50, b: S. 53-54.

[225] Maurer, M.: Das inhärente Risiko autonomer Straßenfahrzeuge (2018).

sätzlicher Ziele und Strategien zu reduzieren. Wird die Ungewissheit mit den daraus abgeleiteten Zielen und Lösungen in ausreichender Form reduziert, besteht die Möglichkeit auf die Angabe der Ungewissheit im Graphen zu verzichten. Die aus der Ungewissheit abgeleiteten Ziele lassen sich dann stattdessen mit der Strategie bzw. dem Ziel verknüpfen, die bzw. das von der Ungewissheit ursprünglich betroffen war.

Die Notation des neuen Elements *Ungewissheit (U)* in der GSN folgt der Darstellung der ebenfalls beschreibenden Zusatzelemente *Annahmen (A)* oder *Legitimationen (eng.: Justification, Abk.: J)*. Die Beschreibung einer Ungewissheit befindet sich daher ebenfalls in einem Oval und steht jeweils im Kontext einer Strategie oder einem Ziel, das entsprechend der GSN Notation auf diese Ungewissheit mit einem nicht ausgefüllten Pfeil Richtung Ungewissheit verweist.[226] Abbildung 4-2 zeigt hierzu ein Beispiel.

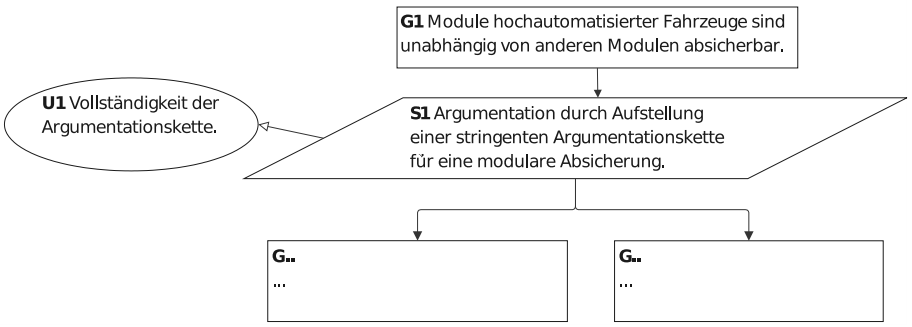


Abbildung 4-2: Beispiel für die Darstellung einer verbleibenden Ungewissheit U1 unter der gewählten Strategie S1 zur Erreichung des Ziels G1 auf Basis der GSN Notation.

Zur Verbesserung der Übersichtlichkeit werden die im folgenden Kapitel entwickelten GSN-Graphen geteilt. Zu deren Verknüpfung wird daher das Symbol nach Abbildung 4-3 eingeführt. Die verknüpften IDs befinden sich unterhalb des Symbols, wenn ein Element in einem weiteren Graphen weiterentwickelt wird. Oberhalb des Symbols befinden sich die IDs, wenn von einem vorherigen Graph auf den entsprechenden Graphen bzw. die Elemente verwiesen wird. Ein Großteil der entwickelten Ziele der folgenden Argumentationskette werden im darauffolgenden Kapitel durch eine gemeinsame Lösung adressiert. Daher werden in den folgenden GSN Graphen die in Kapitel 5 entwickelten Lösungen teilweise durch einen Kreis direkt unterhalb eines Ziels verkürzt dargestellt (vgl. Abbildung 4-3). Zur weiteren Erhöhung der Übersichtlichkeit werden die Elemente teilweise farbig dargestellt. Graue Symbole weisen darauf hin, dass die Inhalte bereits Stand der Technik sind. Dunkelblaue Symbole stehen für Inhalte nach dem Stand der Wissenschaft. Hellblaue Symbole verweisen auf Lösungen, die in der vorliegenden Arbeit neu entwickelt werden. Innerhalb der jeweiligen Unterkapitel 4.2, 4.3, 4.4 wird die Argumentationskette Schritt für Schritt aufgebaut. Zur besseren Übersicht werden die Zwischenergebnisse in unvollständigen GSN Graphen dargestellt. Am Ende der jeweiligen Unterkapitel wird der darin entwickelte Pfad des GSN Graphs noch mal vollständig dargestellt.

[226] SCSC: Goal Structuring Notation Community Standard (2021), S. 12–14.

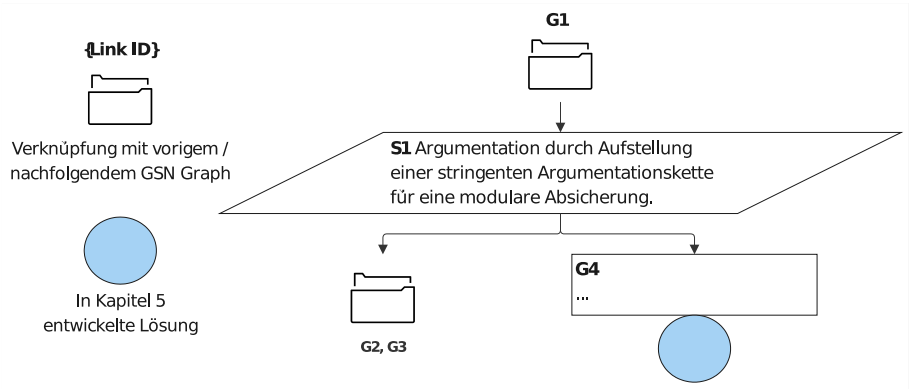


Abbildung 4-3: Erweiterung der GSN Elemente mit Darstellung von Verknüpfungen und einer Lösung, die auf die Lösung in Kapitel 5 verweist. Links: Legende zu Verknüpfung und verkürzter Lösung. Rechts: Beispiel zur Darstellung der Elemente im GSN Graph.

Ausgangspunkt der Argumentationskette

Kapitel 3 stellt den Stand der Wissenschaft zur Absicherung von System-of-Systems vor und zeigt, dass deren Systeme zwar unabhängiger voneinander sind, alle anderen Eigenschaften aber ebenfalls von Modulen erfüllt werden. Die größere Abhängigkeit zwischen Modulen und der Wegfall einer Betrachtungsebene zur Überprüfung des Zusammenspiels dieser Module in Integrations- und Systemtests bei modularer Absicherung erhöhen initial jedoch das Risiko durch zusätzliche Ungewissheiten. Das nach Kapitel 3.2 vorgestellte inhärente Risiko zeigt, dass die Nutzung technischer Systeme nicht risikofrei möglich ist. Daher ist die Elimination jeglicher Ungewissheiten im Rahmen einer modularen Absicherung nicht realistisch. Es ist aber das Ziel G1 diese Ungewissheiten und das damit einhergehende Restrisiko so gering wie möglich und damit auf Höhe des Restrisikos bei Einbezug von Systemtests zu halten. Dies ist nur durch zusätzliche Aufwände in der Analyse der Zusammenhänge der Module untereinander, ihrer Spezifikation und deren Überprüfung durch Tests zu erreichen. Ein quantitativer Vergleich des Restrisikos nach modularer Absicherung zu einer konventionellen Absicherung auf Systemebene z. B. in Form der Anzahl verbleibender Fehlerzustände lässt sich nur retrospektiv ermitteln. Daher werden im Folgenden lediglich qualitative Argumente vorgestellt, die das Sicherheitsniveau der modularen Absicherung auf das einer Absicherung auf Systemebene heben.

Die Argumentationskette verfolgt drei wesentliche Pfade, die in den folgenden Abschnitten erläutert werden. Zur Übersicht sind diese drei Pfade in verkürzter Form in Abbildung 4-4 dargestellt. Die Abbildung zeigt den jeweiligen Beginn eines Pfads der GSN mit den Strategien, Zielen und den zusammengefassten Ungewissheiten (U3, U4 und U11), die zum dritten Pfad führen. Im ersten Pfad wird dargelegt, dass Module äquivalent zu Systemen beschrieben werden können und damit einem System-of-Systems ähneln. Dadurch lassen sich Verfahren zur Absicherung auf Systemebene teils für die Absicherung von Modulen übertragen. Zur Erreichung einer äquivalenten Spezifikation auf Modulebene und einer möglichst hohen Abdeckung an Modultests für den Nachweis, dass diese Spezifikationen erfüllt werden, dient der zweite Pfad. Dieser verfolgt die Strategie S9, dass Modultests vollständig und valide durchgeführt und bestanden sind. Das dafür entscheidende Ziel G23, verfolgt die Ermittlung

möglichst aller Informationen, die zur möglichst vollständigen Spezifikation und Testfallgenerierung auf Modulebene notwendig sind. Damit liegen für das Ziel G2 des ersten Pfads Informationen über ein Modul äquivalent zu einem System vor und für das Ziel G23 des zweiten Pfads all jene Modultests, die zu einer Absicherung äquivalent zu Systemtests notwendig sind.

Die Argumentationskette verfolgt entsprechend die Argumentation, dass eine ausreichende Vollständigkeit und Korrektheit der Modulspezifikation sowie des Modultests sichergestellt werden. Trotz der Maßnahmen wird erwartet, dass Vollständigkeit in allen Punkten nicht erreichbar ist und in allen Entwicklungsschritten zur Ableitung der Spezifikation oder Modultests Irrtümer begangen werden. Daher ist die korrekte Dekomposition der Informationen anhand von Architektursichten ein wesentlicher Bestandteil zur Erreichung einer modularen Absicherung. Im dritten Pfad der Argumentationskette wird daraus folgend das Ziel G30 verfolgt, um die Ungewissheiten durch eine detaillierte Analyse des Dekompositionsprozesses zu reduzieren.

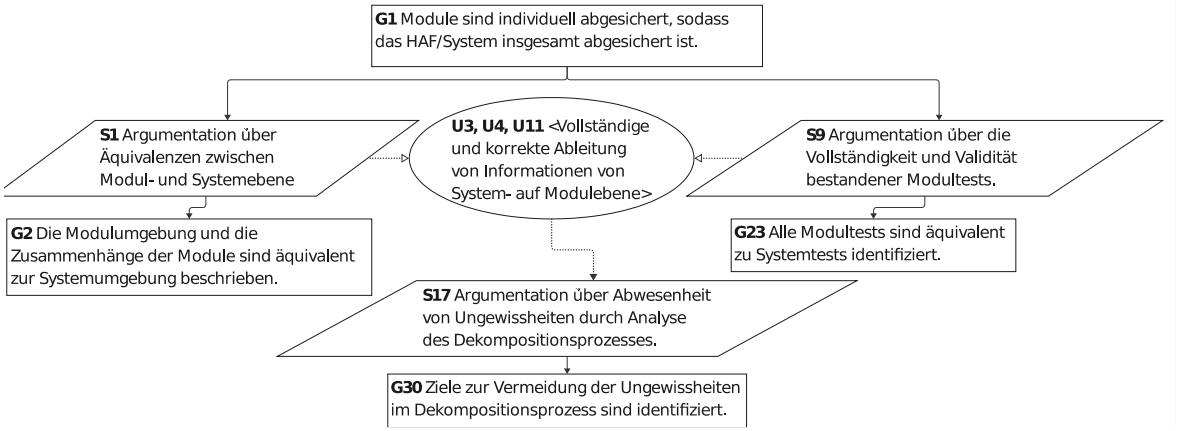


Abbildung 4-4: Übersicht der drei Hauptpfade der GSN Argumentationskette in verkürzter Darstellung. Gepunktete Pfeile weisen auf die Verkürzung des GSN Graphs an entsprechender Stelle hin.

Die Pfade der Argumentationskette für eine modulare Absicherung enden trotz unterschiedlicher Basisargumente und damit einhergehender unterschiedlicher Herangehensweise in ähnlichen Zielen. Die offenen Ziele, die sich nicht mit Methoden nach dem Stand der Wissenschaft und Technik auflösen lassen, lassen sich daher auf zwei wesentliche Punkte zusammenfassen. Erstens wird eine möglichst exakte Beschreibung der Modulumgebung benötigt. Zweitens sind die Ungewissheiten bei Ableitung der Spezifikation von Modul und Modulumgebung zu identifizieren und zu reduzieren. Der Schluss des Kapitels gibt daher einen Überblick zu potentiellen neuen Methoden und Maßnahmen, um diese noch ungelösten Ziele zu erreichen.

4.2 Argumentation über Äquivalenzen verschiedener Hierarchieebenen

Zur Bewertung der Sicherheit ist die direkte Betrachtung der übergeordneten Ebene vorteilhaft, da das induzierte Verhalten dort direkten Einfluss auf die Umgebung und das System hat. Die Auswirkungen des Verhaltens sind direkt beobachtbar. Die Auswirkung des Verhaltens einzelner Module (d.h. auf niedrigeren Hierarchieebenen) muss dagegen erst antizipiert werden. Daher zeigt der erste Pfad der Argumentationskette zur Darstellung der Möglichkeit einer modularen Absicherung, inwieweit Module äquivalent zu einem System betrachtet werden können. Einführend weisen die folgenden Erläuterungen bereits darauf hin, dass auch die Betrachtung der Systemebene Ungewissheiten besitzt. Hierbei werden Ungewissheiten einerseits aufgrund ungewisser interner Verarbeitungsschritte und Zustände erläutert, andererseits aufgrund einer ungewissen Systemumgebung und insbesondere eines ungewissen Verhaltens anderer Akteure dieser Systemumgebung:

Die reine Betrachtung einer höheren Hierarchieebene (im Folgenden wird von der Systemebene ausgegangen) kann Schwierigkeiten bei der Identifikation von Ursachen für fehlerhaftes Verhalten bereiten. Die einzelnen internen Verarbeitungsschritte, die zum Verhalten auf der betrachteten Hierarchieebene führen sind ggf. nicht vollständig bekannt, sodass potentielle Fehlerzustände und Irrtümer nicht erkannt werden. Ein fehlerhaftes Verhalten lässt sich theoretisch zwar auf Systemebene beobachten, *allerdings wird ein Testfall, der fehlerhaftes Verhalten auslöst möglicherweise nicht durchlaufen.*[227] Darüber hinaus sind Metriken und Bestehens-/Versagenskriterien zur Detektion eines bestimmten Ausfallverhaltens auf Systemebene ggf. nicht sensitiv genug bzw. das messbare Systemverhalten zeigt keine Auffälligkeiten, da interne Ausfälle kompensiert werden. Unter anderen Bedingungen besteht für diese Ausfälle aber ggf. eine Gefahr. Beispiel 4-1 erläutert einen Fall in dem spezifische interne Verarbeitungsschritte zu einem tödlichen Unfall führten.

Beispiel 4-1: Interne Verarbeitungsschritte, die in einem realen System nur unter spezifischen Bedingungen zu einem fehlerhaften Verhalten auf Systemebene führten.

Die tödliche Kollision eines Prototypenfahrzeugs mit automatisierter Fahrfunktion der Firma Uber mit einer Person verdeutlicht, wie Wissen über die internen Verarbeitungsschritte die Absicherung unterstützt: Im Unfallszenario schob die Person ein Fahrrad quer über die Fahrbahn außerhalb einer geschlossenen Ortschaft. Die Person wurde von der Perzeption abwechselnd als Fahrrad, Fahrzeug oder unbekanntes Objekt klassifiziert. Mit jedem Wechsel wurde die Person als neues Objekt aufgenommen, sodass keine Prädiktion zukünftiger Positionen möglich war. Trotz durchgängiger Erkennung der Person wurde der Kollisionskurs mit dem Egofahrzeug nicht erkannt, sodass es schließlich zur Kollision

[227] Spillner, A.; Linz, T.: Basiswissen Softwaretest (2019), S. 192.

kam.[228] Das tragische Beispiel zeigt, dass ein solcher Fehlerzustand ggf. nur zufällig durch Tests des Gesamtsystems aufgedeckt werden kann.

Eine systematische Analyse der internen Verarbeitungsschritte kann einen Zusammenhang, wie im Beispiel beschrieben, früher vermeiden oder Tests generieren, die diese Schwachstelle systematisch aufdecken. Bspw. wären Tests denkbar, die das System unter verschiedenen Szenarien mit unplausiblen Informationen testen. Sicherheit kann daher auf Systemebene ohne Wissen über potentielle Ursachen für Versagensfälle z. B. aufgrund bestimmter Verarbeitungsschritte im System nur statistisch argumentiert werden. Das szenariobasierte Testen, das als vielversprechender Ersatz für den statistischen Sicherheitsnachweis gilt, erfordert nach Amersbach und Winner[229] bisher weiterhin einen um mehrere Größenordnungen erhöhten Aufwand ggü. nicht automatisierten Fahrzeugen. Hierbei berechnen sie die Anzahl zu testender Szenarien auf Basis der notwendigen Testkilometer für den statistischen Sicherheitsnachweis, da die Auswahl relevanter Szenarien weiterhin eine offene Forschungsfrage darstellt.

Theoretisch kann ein Verhalten auf der niedrigsten vorstellbaren Ebene (z. B. ein Atom) bis zur höchst vorstellbaren Ebene (z. B. das Universum) eine Wirkung auslösen. Praktisch nimmt diese Wirkung mit jeder Ebene ab, bis sie auf der Ebene des Universums gegen Null konvergiert. Die ausgelöste Auswirkung ist dabei von der Stärke der Beziehungen zwischen den Ebenen und von der Stärke der Auswirkung selbst abhängig. Abbildung 4-5 veranschaulicht diesen Zusammenhang anhand beispielhafter Ebenen und ihren jeweilig beeinflussten Elementen. Die Darstellung ist hierbei um ein hochautomatisiertes Fahrzeug von der Bauteilebene bis zur Ebene des Universums beispielhaft skizziert.

---

[228] Harris, M.: NTSB Investigation Into Deadly Uber Self-Driving Car Crash (2019).

[229] Amersbach, C.; Winner, H.: Test Coverage for Scenario-Based Validation (2019).

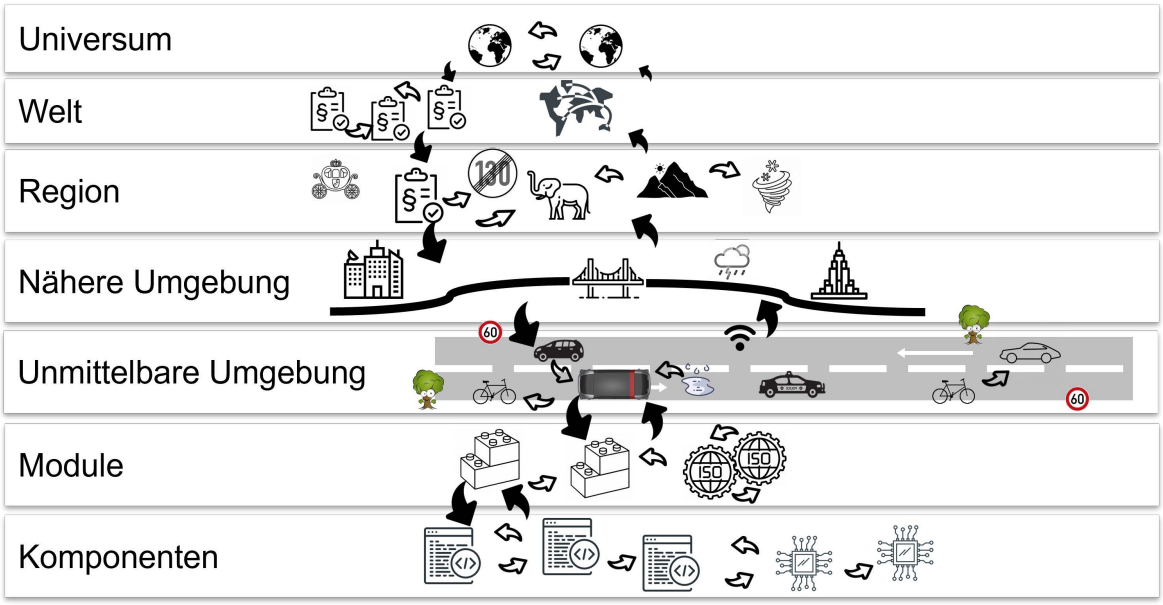


Abbildung 4-5: Bildhafte Darstellung der Zusammenhänge zwischen Hierarchieebenen, in denen sich die Einflüsse innerhalb einer Ebene über die Ebenen hinweg fortpflanzen.

Die Einflüsse und Wirkungen innerhalb eines hochautomatisierten Fahrzeugs sind aufgrund einer Vielzahl an Abhängigkeiten komplex. Ein vereinfachtes Beispiel in Bezug auf die wesentliche Verarbeitungskette für die Fahraufgabe skizziert Beispiel 4-2.

Beispiel 4-2: Einflüsse und Wirkungen über mehrere Hierarchieebenen bei einem hochautomatisierten Fahrzeug.

Das Modul zur Perzeption der Umgebung eines hochautomatisierten Fahrzeugs hat bspw. Einfluss auf dessen Fahrverhalten, da nur bei Wahrnehmung von Objekten auf diese reagiert werden kann. Das hochautomatisierte Fahrzeug hat durch sein Verhalten wiederum Einfluss auf den Verkehrsfluss. Der Verkehrsfluss kann wiederum einen Stau verursachen, wodurch Rettungskräfte z.B. eine Unfallstelle nicht erreichen können.

Zur Beschreibung der Umgebung eines hochautomatisierten Fahrzeugs bzw. der Identifikation relevanter Einflussfaktoren existiert bereits eine Vielzahl potentieller Ansätze. Hierzu werden extra Beschreibungsformen für die Umgebung in einer Operational Design Domain (ODD) und zur Zusammensetzung der Elemente einer ODD in Szenarien entwickelt (vgl. Kapitel 3.2). Nur damit wird die Absicherung eines hochautomatisierten Fahrzeugs als System in seiner Umgebung als möglich angesehen. Die Einflüsse und Wirkungen zwischen anderen Ebenen wird in diesen Forschungsarbeiten jedoch vernachlässigt, da von Tests auf Systemebene ausgegangen wird.

Resultierende Strategien und Ziele zur Argumentation von Äquivalenzen zwischen System und Modul

Anhand der Erläuterungen zu Ungewissheiten über die internen Abläufe in einem System sowie über die äußeren Abläufe und daraus resultierenden Einflussfaktoren, die auf das System einwirken, wird deutlich, dass trotz inkludierter emergenter Eigenschaften auf System-

ebene das System in seiner Einsatzumgebung Ungewissheiten vergleichbar denen eines Moduls unterliegt. Hieraus lässt sich der Ausgangspunkt des ersten Pfads der GSN Argumentationskette, wie in Abbildung 4-6 dargestellt ableiten. Darin argumentiert die erste Strategie S1, dass System- und Modulebene Äquivalenzen besitzen, die eine äquivalente Absicherung ermöglichen. Hierbei wird die Legitimation (J1) gegeben, dass die Ungewissheiten der Absicherung eines Systems als System-of-Systems auch akzeptiert werden. In der Absicherung eines Systems muss äquivalent zu Modulen die sicherheitsrelevante Hierarchieebene identifiziert, analysiert und beschrieben werden, die ggf. auch oberhalb der Ebene des betrachteten Systems liegt.

In der GSN werden Ziele abgeleitet, die eine äquivalente Beschreibung der Module und seiner Umgebung im Vergleich zu System und Systemumgebung ermöglichen. Gleichzeitig geht die Strategie S1 allerdings mit folgenden Ungewissheiten einher:

1. Die Beschreibung von Modulen und ihrer Umgebung besitzt einen höheren Abstraktionsgrad ggü. Systemen sowie deren Umgebung und damit eine höhere Ungewissheit U1. Das Systemverhalten und dessen Einflussfaktoren sind auf Systemebene meist bereits mit geringen Kenntnissen über die inneren Abläufe des Systems beschreibbar. Zur Beschreibung des Modulverhaltens und zur Ableitung des daraus erwartbaren Systemverhaltens sowie potentieller Einflussfaktoren ist dagegen ein tiefes Verständnis der Systemarchitektur notwendig.
2. Grube Doiz[230] zeigt, dass Module ggü. Systemen eine höhere Abhängigkeit zu ihrer Umgebung besitzen. Die Ungewissheit U2 beinhaltet daher potentiell unbekannte Abhängigkeiten von Modulen zur Modulumgebung.

Auf Basis der Argumentation, dass Module äquivalent zu Systemen beschreibbar sind, ergeben sich zwei Ziele: Einerseits muss das Modul selbst äquivalent zum System vollständig für die Absicherung spezifiziert sein. Dieses Ziel G3 ist auch Bestandteil des zweiten Pfades und wird daher in Kapitel 4.3 im Detail beschrieben. Andererseits ist das bereits eingeführte Ziel G2, die Modulumgebung inklusive der Zusammenhänge zwischen den Modulen äquivalent zu einem System zu beschreiben.

Die modellbasierte Entwicklung bietet eine hohe Bandbreite an Möglichkeiten zur Beschreibung der Abläufe und Zusammenhänge in einem System.[231] Darüber hinaus lassen sich Modelle z. B. in Form von Simulationsumgebungen nutzen, um die Modulumgebung und Zusammenhänge zu beschreiben oder in der Simulation darzustellen und zu beobachten. Die Argumentation zur Erfüllung des Ziels G2 erfolgt daher über die detaillierte Beschreibung der Modulumgebung und der Zusammenhänge mit Hilfe solcher Modelle oder Architekturen (S2). Ähnlich zur vorherigen Ungewissheit über die Abhängigkeiten zwischen Modulen geht

[230] Grube Doiz, N.: Bachelor Thesis, Modularisierung in der Automobilindustrie (2021), S. 53–56.

[231] Shevchenko, N.: Introduction to Model-Based Systems Engineering (2020).

diese Argumentation mit der Ungewissheit U3 einher, dass die Abstraktion der Informationen in einer Modellierungssprache ungewisse Informationsverluste erzeugt und fehlerbehaftet sein kann. Die Abstraktion ist zur Komplexitätsbeherrschung zwar erwünscht, kann aber wie erläutert Ungewissheit erzeugen. Zusätzlich weist Vogel[232] darauf hin, dass selbst einer detaillierten korrekten Beschreibung gewisse emergente Eigenschaften fehlen, die das System nur durch Zusammenspiel der einzelnen Funktionen der Module akquiriert und daher nicht vorhersagbar sind. Für die Modellierung ergibt sich daher die Ungewissheit U4, dass die emergenten Eigenschaften des Systems initial ungewiss sind und Annahmen darüber erst durch das tatsächliche Zusammenspiel der realen Module validierbar sind. Diese Ungewissheit ist nicht vollständig zu eliminieren. Allerdings reduziert eine Analyse der Wirkzusammenhänge unter Einsatz von Modellen der Modulumgebung diese Ungewissheit. Kapitel 4.4 argumentiert im Detail wie dies möglich ist. Zusätzlich lässt sich die Strategie S3 einsetzen, die argumentiert, dass eine evolutionäre Entwicklung die Ungewissheit über emergente Eigenschaften reduziert. Das Ziel G4 verfolgt hierfür die Herstellung einer ausreichenden Gleichheit zwischen alten und neuen Modulen sowie alter und neuer Modularchitektur.

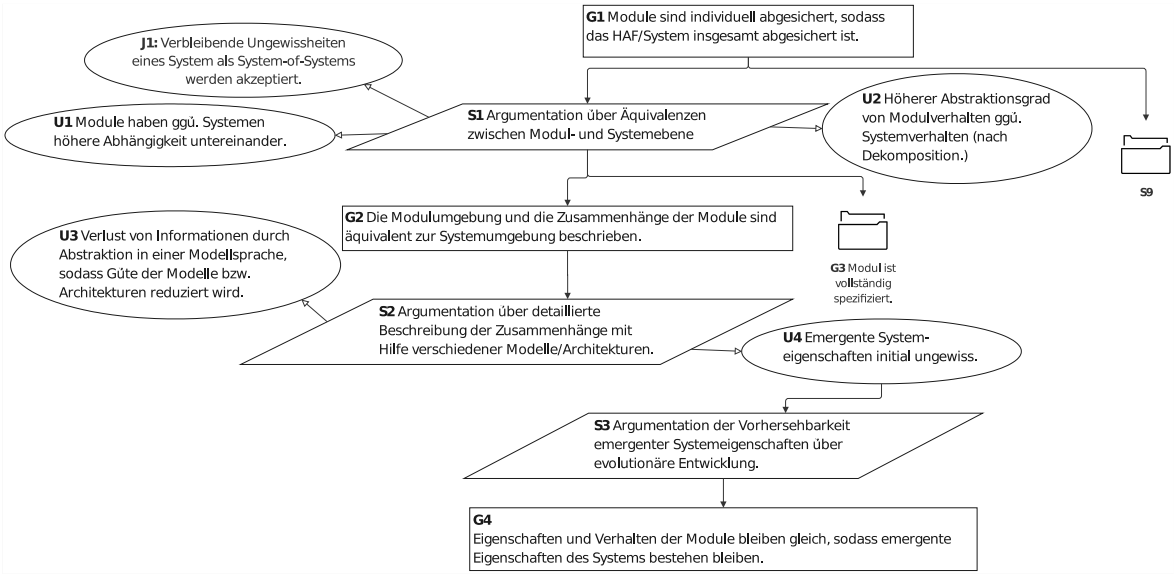


Abbildung 4-6: Ausgangspunkt des ersten Pfades der Argumentationskette.

4.2.1 Einfluss und Wirkung zwischen System und Modul

Strategie S2 argumentiert die Erreichung des Ziels G2 über die detaillierte Beschreibung der Modulumgebung und der Abhängigkeiten mit Modellen oder Architekturen. Im folgenden Absatz wird das Ziel G5 und alle weiteren sich hieraus ableitbaren Strategien, Ziele und Lösungen nach Abbildung 4-7 vorgestellt. Die Vorstellung und weitere Ableitungen des zweiten Ziels G6 erfolgt im darauffolgenden Absatz. Das erste Ziel G5 fordert, dass die Wirkung des Modulverhaltens beschrieben ist. Die Wirkung des Module under Test (MuT)

[232] Vogel, O.: Software-Architektur (2009), S. 59.

darf sich für eine modulare Absicherung jedoch nicht rein auf die unmittelbare Umgebung beschränken. Vielmehr sind zur Prädiktion des Systemverhaltens Kenntnisse über die Wirkung auf die gesamte Umgebung notwendig. Die Strategie S6 zur Erreichung dieses Ziels wird im Folgenden auch über den parallelen Pfad ausgehend von den vorgestellten Ungewissheiten U1 und U3 hergeleitet. Deren weitere Beschreibung folgt in Abschnitt 4.2.2.

Das zweite Ziel (G6) ist die Beschreibung möglicher Einflussfaktoren des Systems und der Systemumgebung auf das Verhalten der einzelnen Module. Zur Sicherheitsargumentation eines hochautomatisierten Fahrzeugs sieht der Stand der Wissenschaft Szenarien der Systemebene vor. Diese sind entsprechend auf einzelne Module herunterzubrechen, das heißt deren Einfluss ist zu identifizieren und das daraus hervorgehende Modulverhalten zu beschreiben. Für die Argumentation des ersten Teils des Ziels G6, kann an dieser Stelle auf bestehende Methoden zur Spezifikation von System und Systemumgebung verwiesen werden (S4). In Bezug auf hochautomatisierte Fahrzeuge sei hierzu insbesondere auf die Beschreibung bestehender Methoden in Abschnitt 3.2 verwiesen. Darüber hinaus nennen etablierte Normen wie die ISO 26262[233] oder die IEEE 1012[234] Methoden zur Identifikation von Anforderungen zur Spezifikation von System und Systemumgebung sowie deren Architektur. Strategie S4 wird in der GSN daher nicht weiterentwickelt. Allerdings verbleibt mit der Notwendigkeit der Systemspezifikation zur Ermittlung der Einflussfaktoren auf Module die Ungewissheit U5, ob die Ableitung der Modulspezifikation durch eine potentiell zu spezifische Systemspezifikation noch gültig ist, wenn sich Teile des Systems ändern. Dazu wird das Ziel G7 ergänzt, um eine ausreichend spezifische bzw. möglichst generische Beschreibung der Systemspezifikation sicherzustellen, die die Austauschbarkeit und Änderbarkeit von Modulen sicherstellt.

Darüber hinaus verbleibt nach Strategie S2, auch bei detaillierter Beschreibung der Abhängigkeiten eine Ungewissheit U6 über mögliche Änderungen des Systems bzw. der Modulumgebung. Bestehende Methoden zur Sensitivitätsanalyse ergeben spezifische Charakteristika für die Sensitivität eines Moduls bzw. einer Modulumgebung. Diese schränken die Weiterentwicklung und den Austausch einzelner Module ein. Ähnliche Problematik beschreibt Viehof[235] bei der Erhaltung der Gültigkeit validierter Simulationsmodelle. Werden Änderungen an Komponenten vorgenommen, die Teil des simulierten Übertragungsverhaltens sind, ist normalerweise eine erneute Validierung notwendig. Viehof stellt daher eine Validierungsmethode vor, die einen erweiterten Gültigkeitsbereich gewährleistet, indem das Modell ggü. verschiedener Systemvarianten validiert wird. Ebenso kann ein theoretisches Modell der Modulumgebung konstruiert werden, das bspw. verschiedene Varianten an Modulen repräsentiert, die das MuT beeinflussen. Das Ziel G8 ist daher auch für die spezifizierte Modulumgebung einen erweiterten Gültigkeitsbereich zu definieren.

---

[233] ISO: ISO 26262 - Road vehicles – Functional safety (2018), Teil 4, S. 6-17.

[234] IEEE: IEEE Std 1012 - Verification and Validation (2017), S. 62–86.

[235] Viehof, M.: Dissertation, Objektive Qualitätsbewertung durch statistische Validierung (2018), S. 36.

In Ergänzung zu Abbildung 4-6 zeigt Abbildung 4-7 den erweiterten Teil des ersten Pfades der Argumentationskette. Hierin werden ausgehend von Strategie S2 und den zuvor identifizierten Ungewissheiten die zuvor erläuterten Elemente zur Vervollständigung der Argumentationskette dargestellt.

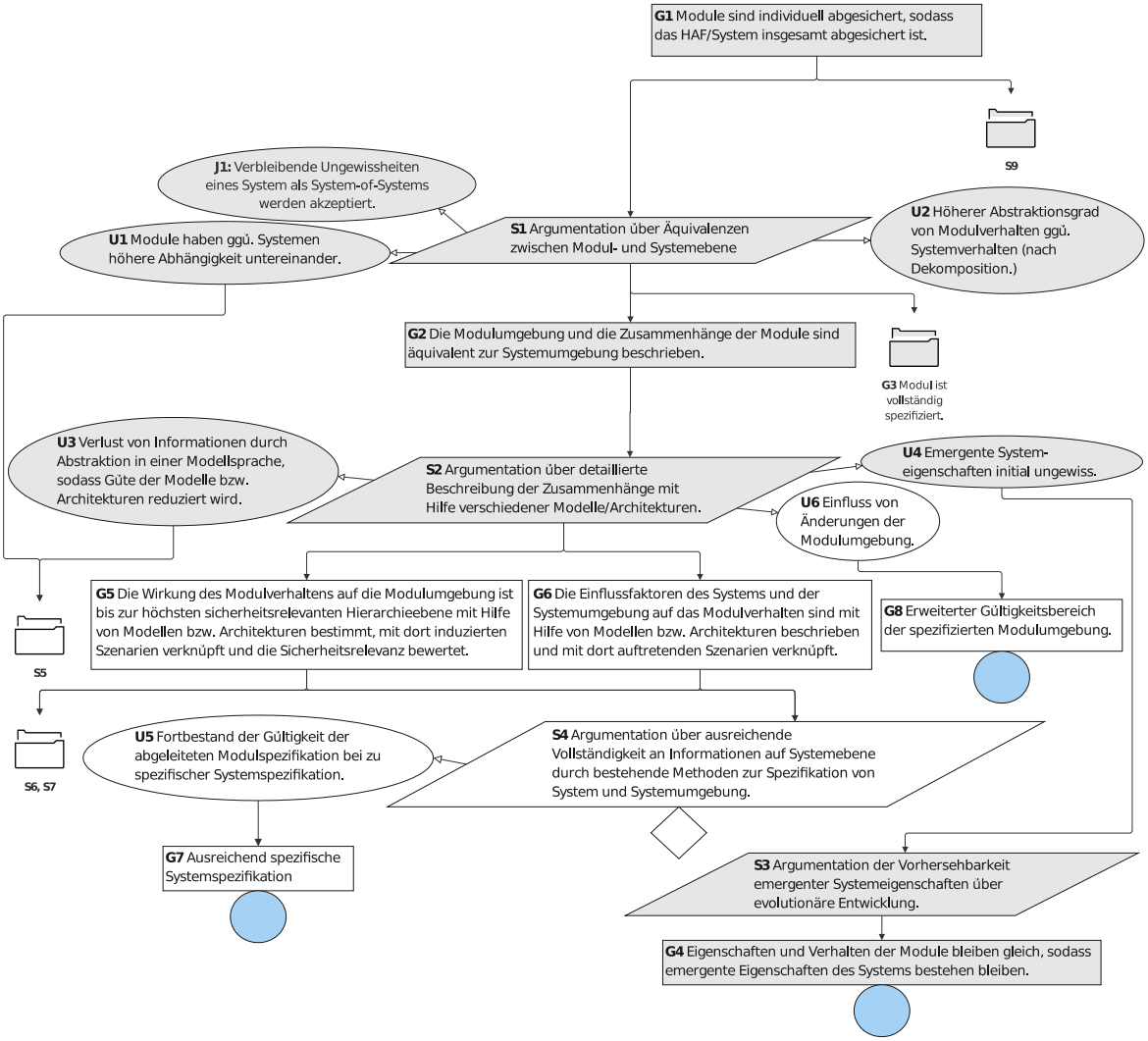


Abbildung 4-7: Einfluss und Wirkung zwischen System- und Modulebene im ersten Pfad der Argumentationskette ergänzend zu grau hervorgehobenen Elementen nach Abbildung 4-6.

4.2.2 Beherrschung der Komplexität

Den offenen Ungewissheiten U1 und U3 des vorigen Unterkapitels kann mit Strategie S5 begegnet werden. Die Strategie argumentiert, dass sich die Güte der Beschreibung der Abhängigkeiten zwischen Modulen, respektive die Modellgüte, durch Beherrschung der Komplexität erhöhen lässt. Das entsprechende Ziel G9 nach Abbildung 4-8 ist die Komplexität

des modularen Systems möglichst gering zu halten. Schoeneberg[236] unterscheidet daher zwischen Komplexitätsvermeidung, -beherrschung, und -reduktion. Maßnahmen hierfür werden, wie in Kapitel 2.2 beschrieben, teilweise bereits durch Prinzipien zur Gestaltung modularer Architekturen adressiert, die jedoch lediglich eine Sammlung an Erfahrungen bei der Architekturgestaltung sind. Im Folgenden werden stattdessen Komplexitätseigenschaften von Modulen bzw. einer modularen Architektur systematisch abgeleitet, woraus sich zusätzliche Ziele und potentielle Lösungen zu deren Erreichung ableiten. Strategie S6 sieht dazu vor, die Komplexität eines Systems und seiner Module zu beherrschen, indem Abhängigkeiten zwischen Modulen identifiziert und beschrieben werden. Darüber hinaus wird die Strategie S7 verfolgt, das jeweilige Modulverhalten zu identifizieren und zu beschreiben, um mit bekannten Abhängigkeiten die Einflussfaktoren auf ein Modul und das daraus folgende Systemverhalten zu ermitteln. Beide Strategien unterstützen gleichzeitig die Ziele G5 und G6 zur Modellierung der Abhängigkeiten in einer modularen Architektur und dem damit ableitbaren Modul- und Systemverhalten.

Im folgenden Abschnitt sind die ermittelten Komplexitätseigenschaften mit deren Nummerierung im GSN-Graph angegeben. Die Eigenschaft selbst ist fett hervorgehoben. Auf eine weitere Beschreibung der restlichen GSN-Argumentationskette wird dagegen verzichtet, um die Herleitung der Komplexitätseigenschaften zu fokussieren und die dabei verwendete Sprache verständlich zu halten. Der restliche GSN-Graph ist in Abbildung 4-8 nachzuvollziehen.

Auf Basis der Charakteristika eines komplexen und eines komplizierten Systems von Kröger und Nan[237] werden in dieser Arbeit zwei Haupteigenschaften definiert, die den Grad der Systemkomplexität bestimmen. Die erste Eigenschaft ist die Abhängigkeit zwischen den Komponenten eines Systems. Die zweite Eigenschaft ist das Verhalten dieser Komponenten. Im Folgenden werden die beiden Eigenschaften systematisch heruntergebrochen.

Abhängigkeiten sind über deren Menge und Ausprägung beschreibbar. Die **Anzahl der Abhängigkeiten** (G10) gibt einen ersten Hinweis auf die Verflechtung der Module innerhalb des Systems. Eine Analyse der Dichteverteilung an Abhängigkeiten ist bspw. hilfreich, um Komponenten, die mehr Abhängigkeiten untereinander haben, zu Modulen zu gruppieren (vgl. bspw. die Interaktionsmatrix von Pimmler und Eppinger[238]). Weitere existierende Methoden sind in den Kapiteln 2.2 sowie 3.4 beschrieben und können allgemein als Prinzipien zu Gestaltung modularer Architekturen zusammengefasst werden (Sn1). Diese beeinflussen auch alle folgenden Eigenschaften positiv, sodass die Komplexität insgesamt reduziert wird.

---

[236] Schoeneberg, K.-P.: Komplexitätsmanagement in Unternehmen (2014), S. 19–22.

[237] Kröger, W.; Nan, C.: Dealing with complexity (2019).

[238] Pimmler, T. U.; Eppinger, S. D.: Integration Analysis of Product Decompositions (1994).

Die Bestimmung der **Stärke der Abhängigkeiten** (G11) ist entscheidend, um das System so zu zerschneiden, dass Module nur soweit abhängig voneinander sind, dass sie austauschbar, änderbar und unabhängig voneinander testbar sind. Herausfordernd und daher ungewiss ist die Beschreibung, was die Stärke einer Schnittstelle kennzeichnet (U7). Göpfert[239] definiert die Stärke der Kopplung zwischen Modulen anhand des erforderlichen Aufwands, um sie zu trennen. In der Mechanik bedeutet dies z. B., dass eine Schraubverbindung zwischen Modulen weniger stark als eine Schweißverbindung ist. In der Softwaretechnik, die bei der Absicherung von automatisierten Fahrzeugen im Vordergrund steht, existieren solche physikalischen Verbindungen jedoch nicht. In der vorliegenden Arbeit wird daher eine generischere Definition eingeführt, die in der GSN die Verfolgung der Strategie S8 und das Ziel G12 für die Bestimmung der Stärke ergibt. Danach ist die *Stärke einer Abhängigkeit abhängig von der Auswirkung auf das Modul bzw. dessen Reaktion, wenn sich die Art oder Qualität der über eine zugehörige Schnittstelle übertragenen Informationen ändert*. Die Stärke hängt u. a. vom **Typ der Abhängigkeit** (G13) ab. Die Beschreibung des Typs kann daher die Bewertung der Stärke der Abhängigkeit unterstützen. Pimmler und Eppinger[240] unterscheiden bspw. die vier Typen von Abhängigkeiten: räumliche, energetische, informelle und materielle Abhängigkeiten. Auf Basis des identifizierten Typs stellen die Autoren außerdem eine wissensbasierte Methode zur Bewertung der Stärke der Abhängigkeiten vor. Kohls[241a] beschreibt für Softwareschnittstellen insgesamt acht verschiedene Typen von Abhängigkeiten. Hierbei werden bspw. dynamisch veränderliche Abhängigkeiten aufgrund von Bedingungen im Softwarecode berücksichtigt sowie Interdependenzen an Schnittstellen z. B. in Form von inter-datenbasierten Abhängigkeiten beschrieben. Inter-datenbasierte Abhängigkeiten beschreiben wechselwirkende Daten z. B. aufgrund von Konsistenzkriterien.[241]

Rinaldi et al.[242] zeigen, dass **Interdependenzen** (G14, G15) als spezieller Typ einer Abhängigkeit die Komplexität durch ihre bidirektionale Abhängigkeit erhöhen. Neben direkten Antworten auf einen Stimulus gehören hierzu auch Rückkopplungsschleifen über andere Module oder die Systemumgebung. Beispiel 4-3 verdeutlicht dies anhand einer Interdependenz zwischen Planung und Regelung. Kohls[241b] stellt hierzu eine Methode zur Identifikation solcher Interdependenzen vor. Diese Methode wird in der GSN-Argumentationskette als eine mögliche Lösung Sn2 aufgenommen.

Beispiel 4-3: Interdependenzen im Bereich hochautomatisiertes Fahren.

Der Trajektorienregler stellt im Projekt UNICAR*agil* Beschleunigungsgrenzen bereit, die vom Trajektorienplaner abonniert werden.[243] Eine Interdependenz ergibt sich, wenn der

[239] Göpfert, J.: Modulare Produktentwicklung (2009), S. 114.

[240] Pimmler, T. U.; Eppinger, S. D.: Integration Analysis of Product Decompositions (1994).

[241] Kohls, S. K.: Master Thesis, Abhängigkeiten zwischen Modulen (2022), a: S. 42–43, b: -.

[242] Rinaldi, S. M. et al.: Identifying critical infrastructure interdependencies (2001).

[243] Homolla, T.: Dissertation, Gekapselte Trajektorienfolgeregelung für autonomes Fahren (2023), S. 84–87.

Trajektorienplaner aufgrund der Beschleunigungsgrenze eine geringere Kurvenkrümmung plant, die wiederum eine Verringerung der Beschleunigungsgrenze erfordert. Dies kann z. B. durch die Fahrt über einen Teil der Fahrbahn erfolgen, auf dem der Reibwert aufgrund von Nässe oder Schnee geringer ist. Die daraus folgende Verringerung der Beschleunigungsgrenze ergibt dann eine weitere Verringerung der geplanten Kurvenkrümmung. Hierdurch wird ggf. ein Fahrbahnabschnitt mit noch geringerem Reibwert erreicht, sodass sich die Kette immer weiter fortpflanzt. Im Beispiel könnte die Beschleunigungsgrenze bspw. für eine bestimmte Zeit konstant gehalten werden, um die Kausalkette zu unterbrechen.

Das Beispiel 4-3 zeigt, dass durch gegenseitige Abhängigkeiten Kausalketten entstehen, die aufgrund einer Kette von nicht vollständig vorhersehbarem Verhalten ein erhöhtes Fehlerpotential bieten.[244] Diese Ketten können durch Sicherheitsmechanismen unterbrochen werden, die fehlerhaftes Verhalten innerhalb des Systems erkennen und darauf reagieren.

Das spezifische **Verhalten** (G16) (siehe Abbildung 4-8) einzelner Module und wie dieses beschrieben werden kann, hängt individuell vom Zweck und der Implementierung ab. Die Systemkomplexität wird jedoch hauptsächlich durch die **Veränderbarkeit des Verhaltens** (G17) beeinflusst. Eine höhere Veränderbarkeit steigert die Komplexität, da sie eine größere Bandbreite an möglichem Verhalten impliziert und eine höhere Anpassungsfähigkeit der beeinflussten Module erfordert. Veränderungen des Verhaltens können ggf. auch nicht bekannt sein, sodass andere Module sich nicht entsprechend adaptieren können. Das superpositionierte Verhalten der Module wird ungewiss und somit die Systemkomplexität erhöht. Eine größere Veränderbarkeit des Verhaltens kann durch das Prinzip und die Vielfalt entstehen, mit der ein Modul arbeitet. Darüber hinaus können Veränderungen des Verhaltens auch durch Updates oder kompletten Austausch der Module entstehen. Ein typisches Beispiel zu möglichen Änderungen von Modulen hochautomatisierter Fahrzeuge illustriert Beispiel 4-4.

Beispiel 4-4: Veränderbarkeit des Verhaltens im Bereich hochautomatisiertes Fahren.

Die Verarbeitungsschritte der Perzeption eines hochautomatisierten Fahrzeugs nutzen üblicherweise Prinzipien des maschinellen Lernens, insbesondere tiefe neuronale Netze. Deren Verhalten ist aufgrund der Größe und Komplexität der Netze für Menschen nicht nachvollziehbar.[245] Das Verhalten kann außerdem bei nur leicht veränderten Szenarien z. B. sehr unterschiedlich ausfallen und folglich nicht eindeutig vorhersehbar sein. Diese hohe Veränderbarkeit des Verhaltens erfordert, dass z. B. die Planung, für dieses spezifische Verhalten abgestimmt werden muss.

Anhand der vorgestellten Komplexitätseigenschaften können zur Veranschaulichung eines hochkomplexen Systems folgende Worst-Case-Bedingungen angenommen werden: Das Verhalten von Modulen in einem System ist ungewiss, nicht beschreibbar und weist eine

[244] Kröger, W.; Nan, C.: Interdependencies of Complex Technical Networks (2014), S. 305.

[245] Faria, J. M.: Non-determinism and Failure Modes in Machine Learning (2017).

hohe Veränderbarkeit auf. Zusätzlich existiert eine hohe Anzahl starker Abhängigkeiten zwischen den Modulen, die zu einem großen Teil nicht bekannt oder nicht beschreibbar sind. Die Module dieses Systems ändern sich außerdem häufig, was zu einer Änderung ihres Verhaltens und ihrer Abhängigkeiten führt.

Aus den hergeleiteten Komplexitätseigenschaften lassen sich die entsprechenden Ziele nach Abbildung 4-8 zur Reduktion oder Beschreibung jener Eigenschaften herleiten, die folglich die Komplexität reduzieren bzw. durch die Aufdeckung der Eigenschaften beherrschbar machen.

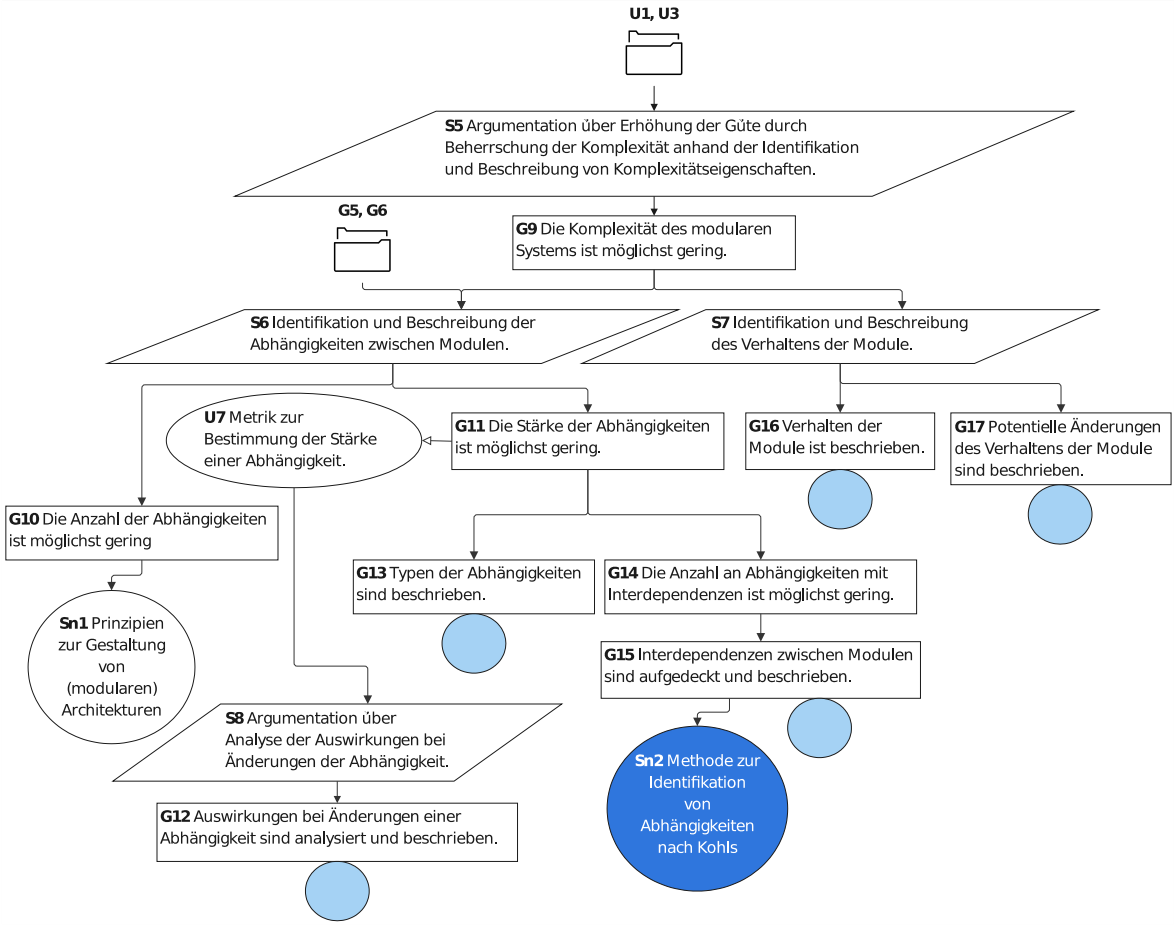


Abbildung 4-8: Beherrschung der Komplexität einer modularen Architektur als Teil des ersten Pfads der GSN Argumentationskette.

4.2.3 Fazit zur Argumentation über Äquivalenzen verschiedener Hierarchieebenen

Anhand des ersten Pfads der Argumentationskette für eine modulare Absicherung lässt sich insgesamt ableiten, dass die Modulumgebung gegenüber einer Systemumgebung abstrakter, komplexer und stärker abhängig voneinander sein kann. Zur Argumentation der Äquivalenz zwischen Modul und System bedarf es daher einer genaueren Analyse und Beschreibung der Eigenschaften, die dieser Abstraktion, Komplexität und Abhängigkeit zutragen. Neben der

notwendigen Beschreibung des Systems nach dem Stand der Wissenschaft und Technik sowie des Moduls selbst und seiner Umgebung, können diese Eigenschaften auf die Beschreibung bzw. die Ziele zur Reduktion bzw. Beherrschung von Komplexitätseigenschaften kondensiert werden. Eine vollständige Elimination der Komplexität zu einem komplizierten System ist für ein hochautomatisiertes Fahrzeug jedoch unwahrscheinlich. Das Ziel der modularen Absicherung ist jedoch lediglich unabhängig von spezifischen Umsetzungen der Module eines zuvor spezifizierten Systems zu sein. D.h. Änderungen an anderen Modulen sollen das MuT nicht beeinflussen. Als Erweiterung zu dem Ziel potentielle Änderungen des Modulverhaltens zu beschreiben, leitet sich aus der Ungewissheit über emergente Eigenschaften eines Systems gegenüber den einzelnen Modulen das Ziel ab, dass die Eigenschaften und das Verhalten der Module für verschiedene Konfigurationen oder Generationen soweit konstant bleiben, dass sich die emergenten Eigenschaften des Systems nicht ändern.

4.3 Argumentation über Vollständigkeit und Validität der Modultests

Zur Absicherung von Modulen verfolgt der zweite Pfad der Argumentationskette die Strategie S9, dass bei der Absicherung bestandene Modultests vollständig und valide durchgeführt sind. Das folgende Kapitel gliedert sich daher in die Analyse dreier Ziele: Modultests müssen bestanden werden, sollen valide durchführbar sein und einen Grad an Vollständigkeit erreichen, der den Systemtests äquivalent ist. Abbildung 4-9 zeigt hierzu den Beginn des zweiten Pfades.

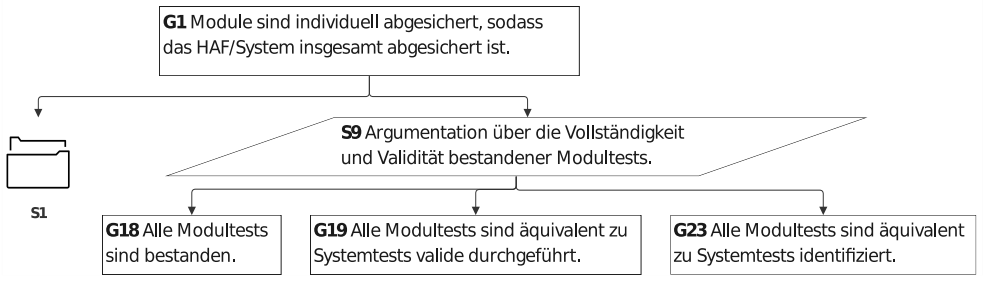


Abbildung 4-9: Übersicht des zweiten Pfades der Argumentationskette.

4.3.1 Bestehen der Modultests

Das erste naheliegende Ziel G18 ist, dass alle Modultests auch bestanden werden. Auf die weitere Ableitung detaillierter Strategien und Ziele von diesem Ziel ausgehend wird verzichtet, da dieses von den Entwicklern zu leisten ist. Die in Kapitel 4.2 vorgestellte Robustheit von Modulen zur Erreichung einer möglichst hohen Änderungsbandbreite der Modulumgebung bzw. eines maximalen Gültigkeitsbereichs z. B. von Simulationsmodellen der Modultests, kann dazu führen, dass andere Anforderungen an das MuT nicht erfüllt werden.

Daher ist eine enge Abstimmung der funktionalen Anforderungen bzw. der Testbestehenskriterien mit den Anforderungen an Robustheit sowie Gültigkeit notwendig.

4.3.2 Valide Durchführung der Modultests

Das Ziel G19 der Strategie S9 fordert, dass alle Modultests auch äquivalent zu den Systemtests valide durchgeführt sind. Das Ziel wird mit der Äquivalenz zu Systemtests eingeschränkt, da das Ziel der modularen Absicherung nicht die Entwicklung besserer oder risikoärmerer Systeme beabsichtigt, sondern den Absicherungsaufwand zumindest über mehrere Systemgenerationen zu verringern. Eine Aufwertung des Qualitäts- oder Sicherheitsniveaus sind lediglich potentielle Nebeneffekte, durch effizienteren Einsatz der vorhandenen personellen und materiellen Ressourcen. Das Ziel kann wie in Abbildung 4-11 dargestellt durch die Validierung aller notwendigen Testumgebungen erreicht werden (S10). Hier ist direkt das Ziel G20 ableitbar, dass nur solche Modultests definiert werden, die eine validierbare Testumgebung erfordern. Das Ziel steht in direktem Konflikt mit dem im weiteren Verlauf des Kapitels folgenden Ziel G23, dass alles Modultests identifiziert sind, wie sie äquivalent auf Systemebene identifizierbar sind. Die Definition der Testfälle ist daher ein iterativer Prozess, in dem die identifizierten Modultests der Realisibarkeit valider Testumgebungen zur Durchführungen der Testfälle gegenübergestellt werden. Der Zusammenhang zwischen den verschiedenen Einheiten der Testumgebung, dem Testfall und den jeweiligen Validitätsanforderungen und -eigenschaften ist in Abbildung 4-10 dargestellt. Die Testumgebung besteht dabei aus dem Module under Test (MuT) und den Modellen der Umgebung sowie von anderen Modulen des gemeinsamen Systems. Diese Modelle besitzen bestimmte Validitätseigenschaften. Die Abbildung zeigt dazu das Testziel als Bestandteil eines Testfalls, aus dem sich die notwendige Validität zur Validierung dieser Eigenschaften ableitet. Ein gleich verlaufender Testfall kann bspw. unter dem Testziel ein Modul auf seine grundlegende Lauffähigkeit zu prüfen, gänzlich andere Validitätskriterien erfordern, als unter dem Testziel eine bestimmte Leistungsfähigkeit des Moduls nachzuweisen. Ebenso hängt die notwendige Validität davon ab, wie sensitiv das MuT auf das Verhalten seiner Umgebung reagiert, d.h. welche Sensitivitätseigenschaften das MuT besitzt.

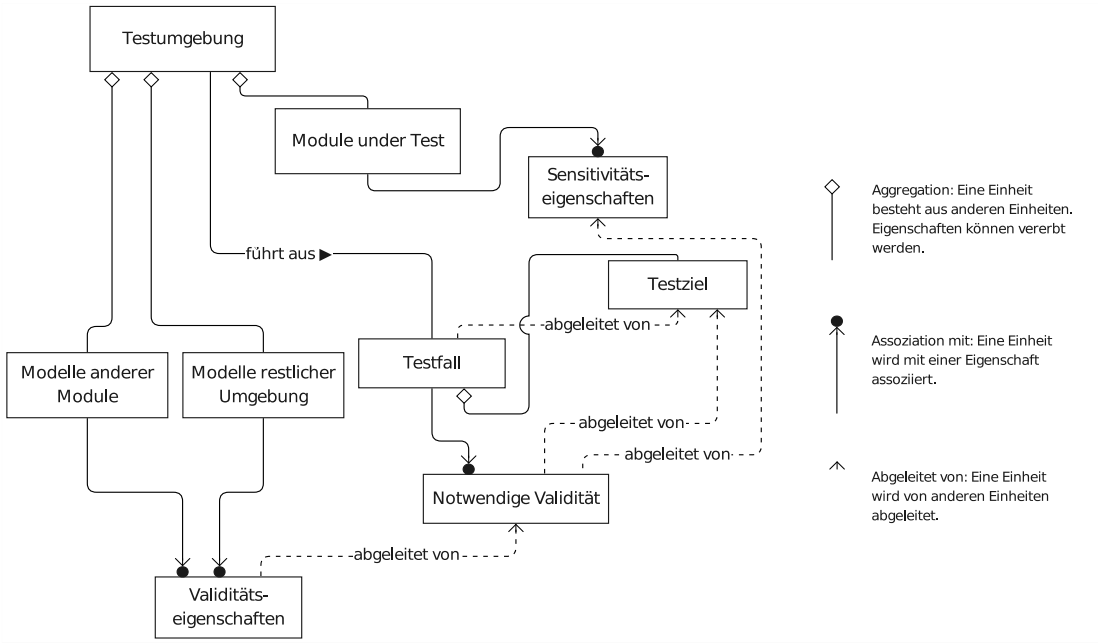


Abbildung 4-10: Zusammenhänge zwischen Testumgebung und Testfall sowie ihrer Validität dargestellt als UML-Klassendiagramm.

Das Ziel G21 fordert schließlich, dass die Testumgebungen für die notwendigen Testfälle validiert sind. Zur Validierung bietet der Stand der Technik und Wissenschaft eine breite Auswahl an Methoden (vgl. die Übersicht von Viehof und Winner[246] und ergänzende Lösungen u. a. von Viehof[247] sowie von Danquah et al.[248]), sodass diese als Lösung Sn3 aufgenommen werden. Für hochautomatisierte Fahrzeuge besteht jedoch weiterhin die Problematik, die Umgebung und die Sensorik zur Wahrnehmung dieser Umgebung ausreichend valide zu modellieren. Dies wird daher als bestehende Ungewissheit U8 in die GSN aufgenommen. Aufgrund des hohen Stellenwertes im Bereich der Entwicklung hochautomatisierter Fahrzeuge ist das Ziel G22 erforderlich, um neue Methoden zur Modellierung der Umgebung und von Sensoren für die Simulationen zu erforschen. Umfangreiche Forschungsarbeiten in den Projekten SET Level[249] oder VIVID[250] verfolgen Lösungen hierzu, weshalb das Ziel in dieser Arbeit nicht weiter betrachtet wird. Parallel dazu besteht unter Strategie S10 die Ungewissheit U9, dass Änderungen an Modulen die vorherige Validierung nicht mehr gültig sind. Kapitel 4.2.1 beschreibt bereits mit Ziel G8, dass bei solchen Änderungen die Erhöhung des Gültigkeitsbereichs der Validierung nach Viehof[251] verfolgt werden sollte.

[246] Viehof, M.; Winner, H.: Modellvalidierung im Anwendungsbereich der Fahrdynamiksimulation (2017).

[247] Viehof, M.: Dissertation, Objektive Qualitätsbewertung durch statistische Validierung (2018).

[248] Danquah, B. et al.: Statistical Validation Framework for Automotive Vehicle Simulations (2021).

[249] SET Level Projekt: Das Projekt hinter SET Level (2022).

[250] Heinze, L.: Steckbrief Forschungsprojekt VIVID (2020).

[251] Viehof, M.: Dissertation, Objektive Qualitätsbewertung durch statistische Validierung (2018), S. 36.

Nach Abbildung 4-10 besteht zusätzlich die Möglichkeit die Sensitivität des MuTs zu reduzieren oder das Testziel zu ändern, um die Anforderungen an die Validität zu reduzieren. Eine Verringerung der Sensitivität bzw. die Erhöhung der Robustheit eines Moduls kann allerdings mit Einbußen der Leistungsfähigkeit einhergehen. Dies ist insbesondere für Verfahren des maschinellen Lernens zu beobachten und wird bspw. durch Mori et al.[252] demonstriert. Darüberhinaus kann eine Änderung des Testziels die Anforderung zur Erreichung der notwendigen Validität verringern, wenn Anforderungen an die Qualität des Test verringert werden, indem bspw. das geforderte Signifikanzniveau für die Validierung erhöht wird. Hierbei verbleibt also die Ungewissheit U10 darüber, ob die Validität der Testumgebungen auf Modulebene ausreichend ist und wie dies argumentiert werden kann, ohne dass Integrationstests notwendig sind. Dabei bleibt ebenso das Ziel G20 offen, dass Modultests auch so definiert sind, dass die notwendigen Testumgebungen validierbar sind. Für beide Ziele wird in Kapitel 4.6.2 eine neue Methode präsentiert, die hierfür eine mögliche Lösung Sn4 darstellt. Dies ist ebenfalls in Abbildung 4-11 ersichtlich, die eine Zusammenfassung des zweiten Pfads der Argumentationskette über die Validität von Modultests darstellt.

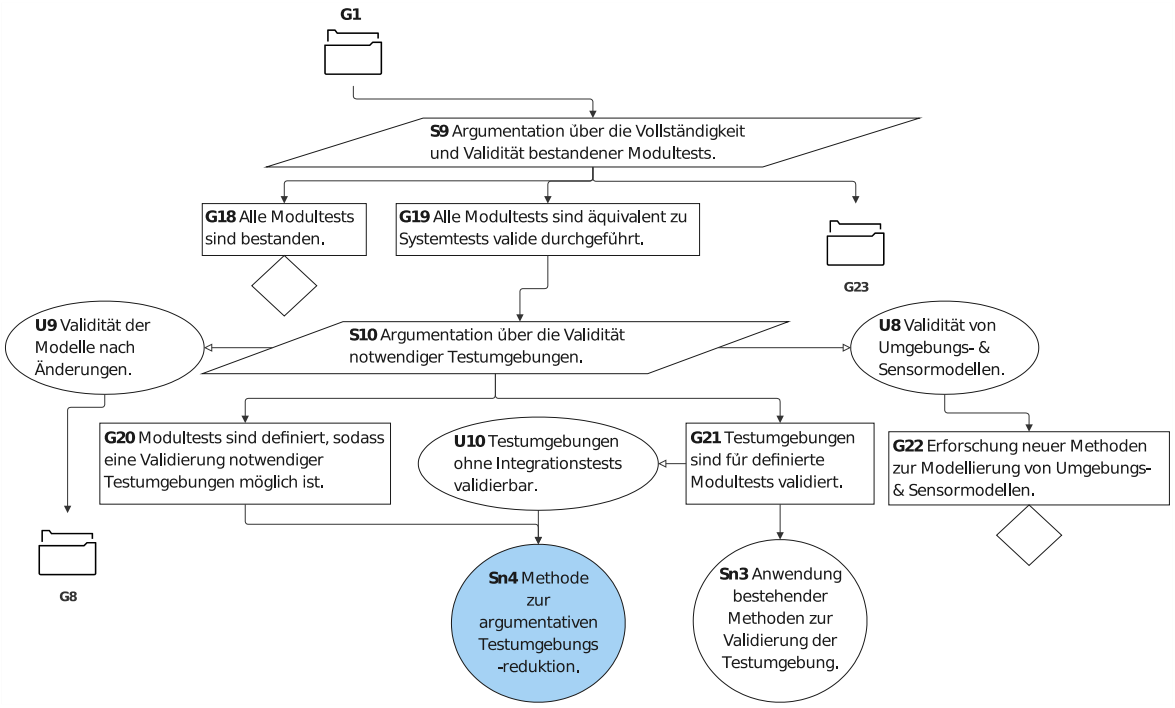


Abbildung 4-11: Zweiter Pfad der GSN Argumentationskette zur Validität von Modultests.

4.3.3 Äquivalenz identifizierter Modultests ggü. Systemtests

Das Ziel G23 nach Abbildung 4-12 trägt zur Argumentation der Vollständigkeit von Modultests als weiterer Teil der Strategie S9 bei. Das Ziel strebt die Identifikation aller Modultests

[252] Mori, K. T. et al.: The Inadequacy of Discrete Scenarios (2022).

an, die zur Absicherung des Moduls notwendig sind. Eingeschränkt wird das Ziel erneut darin nur Fehlerzustände oder Irrtümer mit Modultests aufzudecken, die auch auf Systemebene aufdeckbar wären und somit. äquivalent zu Systemtests sind. Dies unterstützt die Argumentationskette dahingehend, dass auch die modulare Absicherung nicht das ambitionierte Ziel erreichen muss, alle Testfälle zur Aufdeckung aller Fehlerzustände oder Irrtümer zu identifizieren. Insbesondere für hochautomatisierte Fahrzeuge wird nach derzeitigem Forschungsstand auch auf Systemebene erwartet, dass nicht jedes Szenario zuvor identifizierbar ist, das ein ungewünschtes Systemverhalten hervorruft.[253] Modul- und Systemtests sind ohne die jeweiligen Testergebnisse kaum vergleichbar. Stattdessen folgt die weitere Argumentationskette der Strategie S11, dass die Modulspezifikation äquivalent zur Systemspezifikation ist, sodass daraus die notwendigen Modultests abgeleitet werden können.

Aus Strategie S11 ergeben sich zwei Ziele. Als erstes muss das Ziel erreicht werden, dass Modultests anhand einer potentiell vollständigen Modulspezifikation vollständig bzw. verlustfrei abgeleitet sind (G24). Es wird mit Strategie S12 die Annahme getroffen, dass die bestehenden Methoden des Stands der Technik, wie u. a. in Kapitel 2.4 beschrieben, hierfür ausreichend sind. Das zweite Ziel G3 (vgl. auch Kapitel 4.2) verfolgt die entsprechende Vollständigkeit der Modulspezifikation äquivalent zur Systemspezifikation.

Das Ziel einer vollständigen Modulspezifikation ist wiederum durch mehrere Strategien erreichbar. Zunächst wird, wie bereits in Kapitel 4.2 vorgestellt, eine möglichst vollständige Systemspezifikation als Grundlage zur Ableitung der Modultests benötigt (S4). Darüber hinaus muss sichergestellt werden, dass die Systemspezifikation auch auf die Modulebene abgeleitet werden kann, um die Vollständigkeit der Modulspezifikation zu unterstützen. Für eine entsprechende Strategie S13 hat Kapitel 4.2 ebenfalls bereits Ziele hergeleitet, die eine dazu ausreichende Beschreibung der Modulumgebung und der Zusammenhänge zwischen Modulen sicherstellen (G2). Hier offenbart sich jedoch eine zusätzliche Ungewissheit U11, da bei der Ableitung von Informationen Irrtümer geschehen können, die sich in Fehlerzuständen einer Spezifikation oder einer Implementierung manifestieren.

Der zugehörige GSN Graph der Argumentationskette zur Argumentation, dass alle Modultests äquivalent zu Systemtests identifiziert sind, ist in Abbildung 4-12 dargestellt. Darin werden zusätzlich die zwei Strategien S14 und S15 dargestellt, die in den folgenden zwei Unterkapiteln 4.3.3.1 und 4.3.3.2 erläutert werden. S14 argumentiert dabei zunächst, dass die Elemente zur Beschreibung eines hochautomatisierten Fahrzeugs in einer Item Definition nach dem Stand der Wissenschaft auf Module übertragbar sind. Mit der Strategie S15 wird darüber hinaus argumentiert, dass Modultests vollständig herleitbar sind, wenn alle notwendigen Informationen zur Testfallableitung mit etablierten Methoden verfügbar sind.

[253] PEGASUS Projekt: PEGASUS Abschlussbericht (2020), S. 30.

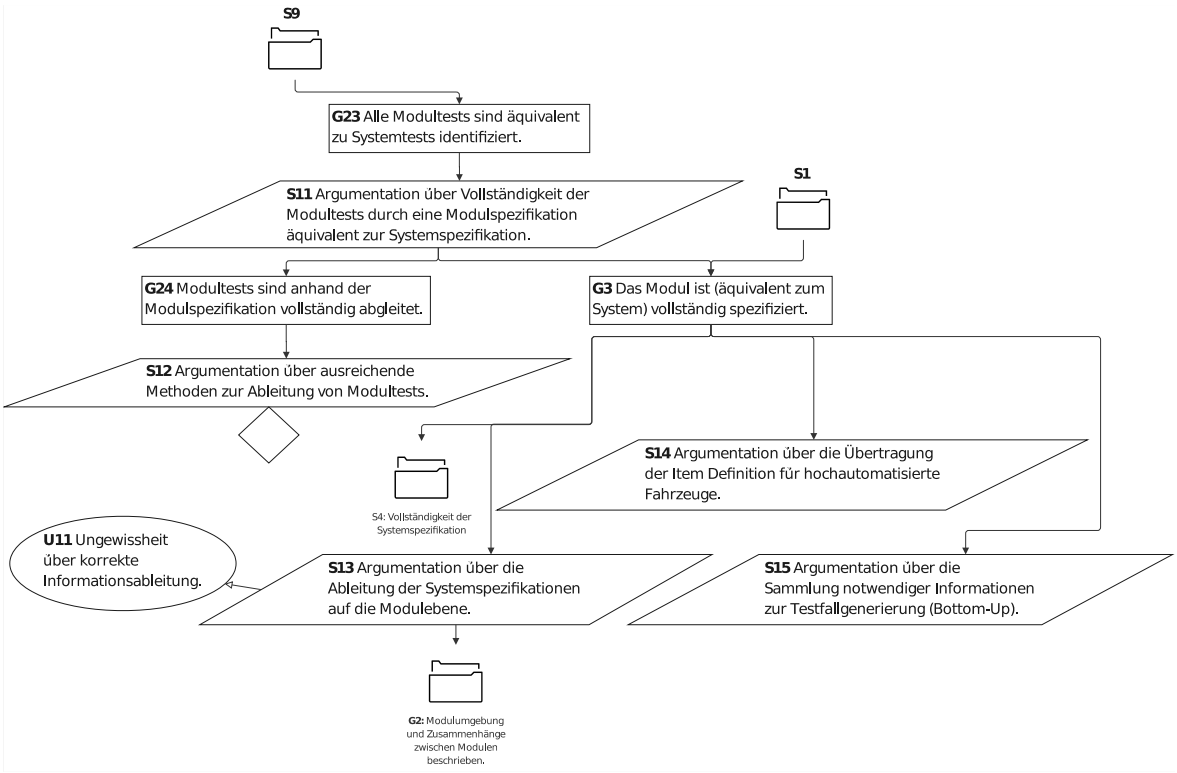


Abbildung 4-12: Ausgangspunkt zur Argumentation der Vollständigkeit von Modultests im zweiten Pfad der Argumentationskette.

4.3.3.1 Übertragung der Item-Definition für ein hochautomatisiertes Fahrzeug

Die folgende Analyse zielt auf die Erreichung des Ziels G23 zur Beschreibung eines Moduls äquivalent zu bestehenden Beschreibungen eines hochautomatisierten Fahrzeugs. Während in Kapitel 4.2 bereits die Ähnlichkeiten der Problematik der Spezifikation hochautomatisierter Fahrzeuge herausgestellt wird, argumentiert Strategie S14 nach Abbildung 4-12 zusätzlich, dass auch neue Elemente der Systemspezifikation eines hochautomatisierten Fahrzeugs in Form einer Item-Definition auf Module übertragbar sind.

Die Norm ISO 26262[254] schlägt die Item-Definition als Grundlage der Systemspezifikation zur Sicherstellung funktionaler Sicherheit vor. Reschka[255] erweitert diese für hochautomatisierte Fahrzeuge. Im Folgenden wird erläutert, inwieweit die Elemente der Item-Definition nach Reschka auf Module übertragbar sind. Hierbei wird nicht erneut analysiert, inwieweit die Informationen auf Modulebene ableitbar sind, sondern die Möglichkeit betrachtet, die konkreten Elemente zur Spezifikation des Systems automatisiertes Fahrzeug auf ein Modul

[254] ISO: ISO 26262 - Road vehicles – Functional safety (2018), Teil 3, S. 4-5.

[255] Reschka, A.: Dissertation, Fertigkeiten- und Fähigkeitengraphen von automatisierten Fahrzeugen (2017), S. 101–105.

zu übertragen. Die Analyse steht für sich und wird nicht vollständig in die Argumentationskette eingebunden, da die spezifischen Elemente der Item-Definition nicht vollständig übertragbar sind. Die ermittelten Möglichkeiten und Hindernisse unterstützen jedoch den Aufbau einer Sicherheitsargumentation für die modulare Absicherung eines konkreten Systems. Zu jedem Bestandteil werden die Informationen hergeleitet, die zur Beschreibung aus Modulsicht notwendig sind. Darauffolgend werden in Stichpunkten die sich daraus ergebenden Ziele aufgelistet, die in Kapitel 5 der Identifikation notwendiger Informationen über Modulschnittstellen dienen.

Anwendungsfall: Die Übertragung des Anwendungsfalls auf Modulebene entspricht einer üblichen Funktionsbeschreibung. Zur Schaffung eines echten Äquivalenten sollte die Beschreibung des Anwendungsfalls von Modulen die Verwendung des beschriebenen Moduls innerhalb des Systems (z. B. durch andere Module) erläutern. Hierzu zählt bspw., wie Informationen des betrachteten Moduls genutzt werden oder welche anderen Dienstleistungen (z. B. Speichern von Informationen oder zur Verfügung stellen von Rechenkapazität) genutzt werden. Daher sind für Module folgende Ziele anzustreben, um Informationen über Module zu aggregieren, die zu der Beschreibung des Anwendungsfalls vergleichbar sind:

- Verwendung des Moduls innerhalb seines Systems bzw. aus Sicht anderer Module spezifiziert
- Modulanwendungsfälle mit Systemanwendungsfällen verknüpft

Automatisierungsgrad: Der Automatisierungsgrad bestimmt die möglichen Anwendungsfälle auf System- wie auf Modulebene. Eine Bestimmung des Automatisierungsgrads auf Modulebene ist dagegen nicht zielführend, da der Automatisierungsgrad vom Zusammenspiel mehrerer Module abhängt. Zur Identifikation geltender Normen und Gesetze sowie zur Ableitung von Anwendungsfällen und -szenarien sollten einem Modul alle potentiellen Automatisierungsgrade zugeordnet werden, für die es im Systemverbund eingesetzt werden soll:

- Module mit Automatisierungsgraden, d.h. jene, für die es eingesetzt werden soll, verknüpft und daraus folgende Anforderungen definiert

Anwendungsszenarien: Die Anwendungsfälle der Module können äquivalent zur Systemebene als einzelne Szenarien (mit der Definition nach Ulbrich et al.[256]) detailliert werden. Den Anwendungsfällen entsprechend fallen Szenarien auf Modulebene ggf. abstrakter als auf Systemebene aus. Die Gesamtheit solcher abstrakten Testszenarien wird im Folgenden als Lastkollektiv bezeichnet. Dies entspricht dem Sprachgebrauch mechanischer Systeme, für die ebenfalls abstrahierte Testszenarien entwickelt werden, um diese modular zu testen. Ein Beispiel hierfür sind Getriebe, die auf Basis zuvor ermittelter Kollektive ausgelegt und getestet werden. Diese Lastkollektive sind u. a. abhängig von den zugeordneten Motoren und dem vorgesehenen Einsatzzweck des Fahrzeugs. Daher sind für Module folgende Ziele

[256] Ulbrich, S. et al.: Definition Szene, Situation und Szenario (2015).

anzustreben, um Informationen über Module zu aggregieren, die der Beschreibung der Anwendungsszenarien vergleichbar sind:

- Anwendungsszenarien bzw. Lastkollektive aus Anwendungsfällen abgeleitet
- Lastkollektive mit Systemszenarien zur Reduktion der Ungewissheit verknüpft
    - Systemszenarien nach Einflussfaktoren auf das Modul modulspezifisch geclustert (Anmerkung: Verschiedene Systemszenarien unterscheiden sich aus Modulsicht ggf. nicht voneinander)

Sollverhalten: Das Verhalten eines Moduls auf bestimmte Stimuli ist anhand von Tests bestimm- und auch verifizierbar. Die Definition eines Modulsollverhaltens muss jedoch die weiteren Auswirkungen berücksichtigen, sodass bspw. das gewünschte Systemverhalten erzeugt wird. Es besteht jedoch bereits mit den Anforderungen an ein Modul und der Systemarchitektur eine Erwartungshaltung gegenüber des ausgelösten Systemverhaltens durch bestimmte Modulausgaben. Hieraus lässt sich ein Modulsollverhalten definieren. Die Ungewissheit betrifft dabei erneut die weitere Verarbeitungskette bis zur höchsten sicherheitsrelevanten Hierarchieebene. Für das Modulsollverhalten werden zusätzlich zu den vorherigen Inhalten der Item Definition folgende Ziele angestrebt:

- Informationen zur Bewertbarkeit des Modulverhaltens beschrieben z. B. durch
    - Systemarchitektur mit Beschreibung des Verhaltens aller Module in Abhängigkeit ihrer Stimuli
    - Generische Metrik zur Bewertung des Modulverhaltens im Zielsystem
    - Generische (Simulations-)Modelle aller Module

Sichere Zustände: Für Module ist die Definition des Verhaltens unter Fehlerzuständen (im Folgenden als Fehlerverhalten bezeichnet) ebenfalls von essentieller Bedeutung zur Sicherstellung eines sicheren Verhaltens bzw. zur Erreichung sicherer Zustände auf Systemebene. Äquivalent zum Sollverhalten ist für das Fehlerverhalten eines Moduls ein resultierendes Systemverhalten zu bestimmen. Zuvor müssen hierfür mögliche Fehlerzustände identifiziert werden. Im Rahmen einer Absicherung hochautomatisierter Fahrzeuge zählen hierzu neben kompletten Funktionsausfällen oder einer reduzierten Performanz, auch Abweichungen vom Sollverhalten. Hierzu wird für hochautomatisierte Fahrzeuge nach ISO/TR 4804[257] der Einsatz eines Betriebsmodusmanagements (eng.: ADS mode manager) vorgeschlagen, das Informationen aus der Überwachung der einzelnen Elemente des Systems nutzt. Module sollten daher ebenfalls die Überwachung der Zustände des Moduls und seiner Ausgaben vornehmen. Schlatow et al.[258] schlagen für eine solche Überwachung eines hochautomatisierten Fahrzeugs den Begriff Selbstwahrnehmung (eng.: self-awareness) vor. Daher sind für Module folgende Ziele anzustreben, um Informationen über Module zu aggregieren, die der Beschreibung sicherer Zustände eines hochautomatisierten Fahrzeugs vergleichbar sind:

[257] ISO: ISO/TR 4804 - Safety and cybersecurity for automated driving systems (2020), S. 30–32.

[258] Schlatow, J. et al.: Self-awareness in autonomous automotive systems (2017).

- Informationen zur Bewertbarkeit des Fehlerverhaltens des Moduls beschrieben z. B. durch
  - Systemarchitektur mit Beschreibung des potentiellen Fehlerverhaltens aller Module in Abhängigkeit ihrer Stimuli
  - Generische Metrik zur Bewertung des Fehlerverhaltens im Zielsystem
  - Generische Modelle, die ein Fehlerverhalten ausreichend valide simulieren
- Potentielle Fehlerzustände und Irrtümer beschrieben
- Informationen für eine Selbstwahrnehmung zur frühzeitigen Erkennung eines Fehlerverhaltens unter Berücksichtigung der Auswirkungen auf der Systemebene beschrieben
  - Charakteristika für Abweichungen vom Sollverhalten beschrieben und mit Ursachen verknüpft

Fahrmanöver: Das Sollverhalten von Modulen kann z. B. anhand der Differenzierung der Ausgaben am Modul oder mit Hilfe derselben Fahrmanöver auf Systemebene gruppiert werden. Eine Gruppierung nach Fahrmanövern hat den Vorteil, dass diese bereits für das System „hochautomatisiertes Fahrzeug“ definiert sind. Die Unterschiede bei der Gruppierung sind allerdings stark vom betrachteten Modul abhängig. Wie bei Szenarien ergeben sich in unterschiedlichen Manövern (wie sie auf Systemebene beobachtbar sind) für ein Modul ggf. übereinstimmende Stimuli und Ausgaben. Ein Antriebsmotor erfährt z. B. bei den von Reschka definierten Manövern „Fahrstreifenwechsel“ und „Abbiegen“ sehr ähnliche Stimuli bzw. erzeugt sehr ähnliche Ausgaben. Das Ziel für ein einzelnes Modul sollte daher gegenüber den Anwendungsszenarien und den Fahrmanövern sein, auch eine eigene Unterteilung der Lastkollektive zu erzeugen, um Stimuli und Ausgaben zu gruppieren. Die Gruppierung ist dabei stark von der Charakteristik der Informationen und der Sensitivität des Moduls abhängig. Die Gruppierung entspricht dem Verfahren zur Bildung von Äquivalenzklassen schnittstellenbasierter Testverfahren, bezieht jedoch direkt Kenntnisse über die Sensitivität des Moduls mit ein. Daher sind für Module folgende Ziele anzustreben, um Informationen über Module zu aggregieren, die der Beschreibung von Fahrmanövern vergleichbar sind:

- Fahrmanöver auf Systemebene mit Informationen auf Modulebene verknüpft
- Stimuli und Ausgaben auf Modulebene beschrieben und geclustert
  - Sensitivität der Module ggü. Stimuliänderungen identifiziert
  - Auswirkungen des Modulverhaltens auf höchste sicherheitsrelevante Hierarchieebene beschrieben

Fertigkeiten: Module können einzelnen Fertigkeiten des Systems zugeordnet werden. Mit einer weiteren Detaillierung der Fertigkeiten lässt sich auch für ein Modul ein eigener Fer-

tigkeitengraph erstellen. Für die Fertigkeiten eines Moduls besteht allerdings die Schwierigkeit, dass teils kein äquivalentes menschliches Verhalten existiert. Reschka[259] lässt das Vorgehen zur Ableitung des menschlichen Verhaltens offen, sodass eine alternative Grundlage für Module unklar bleibt. Der Fertigkeitengraph kann daher abhängig vom Modul vergleichsweise klein und damit wenig nützlich zur weiteren Ableitung von Anforderungen sein. Fertigkeiten können als Spezialfall für lösungsunabhängige Anforderungen auf Systemebene betrachtet werden, die ein bereits bestehendes System (den menschlichen Fahrer) abbilden. Die Beschreibung von Fertigkeiten auf Modulebene lässt sich daher mit der Definition lösungsunabhängiger Anforderungen auf Basis des Modulsollverhaltens unter verschiedenen Stimuli übertragen. Daher sind für Module folgende Ziele anzustreben, um Informationen über Module zu aggregieren, die der Beschreibung der Fertigkeiten vergleichbar sind:

- Fertigkeitengraph auf Modulebene in Verbindung mit Fertigkeiten des Systems modelliert (d.h. ein gemeinsamer Fertigkeitengraph mit dem System, der äquivalent zu anderen Architektursichten aufgebaut ist)
- Fertigkeiten falls möglich mit menschlichem Verhalten verknüpft
- Fertigkeiten auf Modulebene als lösungsunabhängige Anforderungen auf Basis des Modulsollverhaltens abgeleitet

Funktionale Systemarchitektur: Die funktionale Systemarchitektur wird bereits nach dem Stand der Technik auch für Module bzw. für Funktionen der Module definiert. Einzelne funktionale Komponenten können Modulen zugeordnet werden. Aus Modulsicht lassen sich diese weiter herunterbrechen. Zur Definition der Funktionen dienen das Modulsollverhalten und die Modulfertigkeiten bereits als Informationsquelle. Dieses Vorgehen vermeidet eine verfrühte Definition lösungsabhängiger Funktionsbeschreibungen, die bei Betrachtung der Module als fertige Teillösungen zu erwarten ist.[260]

Fazit zur Übertragbarkeit der Item Definition auf Module

Die Analyse der Übertragbarkeit der Elemente der Item Definition nach Reschka[259] zeigt, dass Module sich ähnlich dem System „hochautomatisiertes Fahrzeug“ definieren lassen. Über alle Aspekte der Item-Definition lässt sich jedoch feststellen, dass es Unterschiede ggü. der Beschreibung der Systemebene und damit Ungewissheiten bei einer potentiellen Modul Definition gibt. Daher wird zur Aufstellung einer äquivalenten Modul-Definition folgendes empfohlen:

- Die Informationen über Module sind bis zur höchsten sicherheitsrelevanten Hierarchieebene rückverfolgbar. In Verbindung mit der Item-Definition nach Reschka[259] sollten

[259] Reschka, A.: Dissertation, Fertigkeiten- und Fähigkeitengraphen von automatisierten Fahrzeugen (2017), S. 146–148.

[260] Ebert, C.: Systematisches Requirements Engineering (2014), S. 23.

nach der vorigen Analyse ausgehend von der Systemebene insbesondere die Anwendungsfälle, Automatisierungsgrade inkl. der zugehörigen Regeln bzw. Anforderungen, Systemszenarien, Fahrmanöver und der Fertigkeiten des Systems und eines menschlichen Fahrers verbunden werden.

- Zur äquivalenten Übertragung der Item-Definition Elemente in eine Modul-Definition sollte die Abstraktion möglichst gering sein. Die Elemente der Modul-Definition sollten aus der Perspektive eines möglichst unabhängigen, eigenständigen Moduls beschrieben werden, als sei es ein mit der Umgebung interagierendes Item.

Das Ziel G25 zur Spezifikation eines Moduls äquivalent zu der eines hochautomatisierten Fahrzeugs ist mit den in der Analyse ermittelten Teilzielen zu Teilen möglich. Die detaillierten Ziele G25a bis G25h sind in Anhang A zusammengefasst. Neben den beschriebenen Ungewissheiten über die Ableitung und Abstraktion der Informationen von System- auf Modulebene verbleibt eine Ungewissheit auch auf Systemebene durch noch offene Forschungsfragen zur Absicherung hochautomatisierter Fahrzeuge. Die vorliegende Äquivalenzanalyse muss daher auf den Stand neuer Forschungsergebnisse erweitert werden.

4.3.3.2 Notwendige Informationen zur Testfallgenerierung

Die bisherigen Strategien zur Modulspezifikation haben sich auf eine Top-Down Betrachtung ausgehend von der Systemspezifikation konzentriert. Die folgende Strategie S15 argumentiert die Vollständigkeit der Modulspezifikation zusätzlich anhand der Sammlung notwendiger Informationen zur Testfallgenerierung. Dabei werden ebenfalls Informationen von der Systemebene abgeleitet, zusätzlich wird aber auf Basis bestehender Methoden zur Testfallgenerierung auch ein Bottom-Up Ansatz verfolgt.

Kapitel 2.4.1 zeigt, dass nach der Norm ISO 26262 konkrete Testmethoden und Analyseverfahren für Items oder Module je nach ASIL Einstufung explizit vorgeschrieben sind.[261] Die Literatur beinhaltet weitere Testmethoden, die in der Norm ISO 26262 nicht genannt werden. Anstatt jede dieser Testmethoden einzeln zu betrachten, lassen sich Testmethoden nach ihren notwendigen Informationsquellen gruppieren. Zur Identifikation der notwendigen Informationen (G26) wird mit Strategie S16 entsprechend angenommen, dass sich bereits existierende Methoden den im folgenden vorgestellten Gruppen zuordnen lassen. Zusätzlich zu den Testmethoden aus den Normen wird die systemtheoretische Prozessanalyse (STPA) als Risikoanalysemethode betrachtet, da sie, verglichen mit anderen Methoden, eine eigene Modellierung des Systems erfordert und gleichzeitig als neue, vielversprechende Methode zur Sicherheitsanalyse komplexer Systeme gilt.[262, 263]

---

[261] ISO: ISO 26262 - Road vehicles – Functional safety (2018), Teil 4, S. 15-24.

[262] Patriarca, R. et al.: Past and present of System-Theoretic Accident Model And Processes (2022).

[263] Jimeno Altelarrea, S. et al.: STPA enabled safety assessment of complex systems (2022).

Die Unterteilung der Testmethoden orientiert sich an den Vorschlägen von Horstmann[264] und von Weitzel et al.[265]. Diese unterscheiden zwischen spezifikationsbasierten, risikobasierten und schnittstellenbasierten Testmethoden. Die Testmethoden sind weitgehend auf verschiedenen Hierarchieebenen anwendbar. Allerdings prüfen die generierten Tests auf Systemebene teils andere Bereiche als auf Modulebene ab. Die Unterschiede werden in folgenden Absätzen herausgestellt. Die dabei aufgezeigten Informationen über Module und erweiterten Methoden zur schnittstellenbasierten Testfallgenerierung dienen der teilweisen Kompensation der damit einhergehenden Ungewissheiten über die Vollständigkeit und Äquivalenz der Modultests ggü. Systemtests. In der GSN wird dies mit den Zielen G27, G28 und G29 nach Abbildung 4-13 verfolgt. Die notwendigen Informationen sind in den folgenden Herleitungen im Text fett hervorgehoben. Die notwendigen Informationen in Form formulierter Ziele der GSN Argumentationskette sind in Anhang A tabellarisch zusammengefasst.

Spezifikationsbasierte Testfallgenerierung

Die spezifikationsbasierten Methoden zur Testfallgenerierung definieren Testfälle anhand der Inhalte der Spezifikation eines Objekts. In einer Spezifikation werden bspw. die vorgesehene Umgebung, die Benutzer und ihr Verhalten durch Anwendungsfälle oder die Operational Design Domain (ODD) beschrieben.[266] Anhand der Spezifikation werden wiederum Anforderungen abgeleitet. Die folgende Analyse berücksichtigt über die bereits beschriebene Item-Definition hinaus Verfahren zur Identifikation von Anforderungen. Die spezifikationsbasierte Testfallgenerierung dient entsprechend zur Verifikation der Anforderungen. Die Testfälle müssen dazu 100 % der spezifizierten Anforderungen abdecken.[267]

Die Unterteilung von Anforderungen in verschiedene Kategorien unterstützt die Übersichtlichkeit und die Identifikation passender Informationsquellen. Nach Ebert[268] sind die Hauptkategorien funktionale Anforderungen, Qualitätsanforderungen und Randbedingungen. Die drei Kategorien können jeweils weiter unterteilt werden. Zu den Qualitätsanforderungen zählen Sicherheitsanforderungen, die die wichtigste Kategorie für die Absicherung darstellen. Bei hochautomatisierten Fahrzeugen können jedoch alle Anforderungen an die automatisierte Fahrfunktion ein unsicheres Fahrverhalten erzeugen und müssen daher bei der Absicherung mitberücksichtigt werden. Techniken zur Ermittlung von Anforderungen

[264] Horstmann, M.: Dissertation, Verflechtung von Test und Entwurf (2005), S. 108.

[265] Weitzel, A. et al.: Absicherungsstrategien für Fahrerassistenzsysteme (2014), S. 43.

[266] Reschka, A.: Dissertation, Fertigkeiten- und Fähigkeitengraphen von automatisierten Fahrzeugen (2017), S. 101–105.

[267] Horstmann, M.: Dissertation, Verflechtung von Test und Entwurf (2005), S. 108.

[268] Ebert, C.: Systematisches Requirements Engineering (2014).

listen Bennaceur et al.[269] auf und unterteilen diese in Datenerfassungs-, kollaborative, kognitive, kontextuelle und kreative Techniken.

Zur *Datenerfassung* von Modulen ist ein bereits existierendes System mit den gleichen Schnittstellen dieser Module notwendig. Daher ist dies nur mit einem **evolutionär entwickelten System** möglich. Die Analyse der Daten eines existierenden **Systems mit ausreichend äquivalenten Schnittstellen bzw. Modulen** können bereits Inspiration und Fakten zur Definition von Anforderungen fördern. Was ausreichend äquivalent ist, muss von den Modulentwicklern jedoch individuell entschieden werden. Neben der Analyse technischer Datensätze schlagen Bennaceur et al.[269] vor, bestehende Spezifikationen, Pläne, Änderungshistorien oder bestehendes Wissen der Entwickler zu erfassen und zu analysieren. Daher ist für eine evolutionäre Entwicklung von Modulen die **Dokumentation aller Entwicklungs- sowie Entscheidungsprozesse, aber auch jeglicher Fehlerzustände und Irrtümer** während der Entwicklung und des Betriebs hilfreich.

Kollaborative und kognitive Techniken zur Anforderungsgenerierung erfordern vorwiegend Expertenwissen und genaue Kenntnisse über das konkrete System bzw. das Modul.[270] Auch hier sind ein vorher bestehendes äquivalentes System und das Wissen der daran beteiligten Entwickler hilfreich. Darüber hinaus sollte bei den Verantwortlichen zur Anforderungsgenerierung von Modulen ein möglichst valides mentales Modell über das zu entwickelnde Modul bestehen. Beschreibungen über dessen **Funktion und seine Einbindung in die Systemumgebung** sind daher z. B. in Form der zuvor dargestellten Item- bzw. Modul-Definition nützlich.

Kontextuelle Techniken ordnen das untersuchte System in seine Umgebung ein und beobachten, wie es darin arbeiten würde.[270] Für Module ist es hierfür notwendig die **Modulumgebung näher zu beschreiben** und mögliche **Umgebungseinflüsse zu identifizieren**. Die Schwierigkeit bei Modulen besteht darin, dass deren Umgebung ggü. einem System, das für den Menschen beobachtbares Verhalten aufweist, schwieriger vorstellbar sein kann. Die Modulumgebung besteht in der Informationstechnik üblicherweise aus ausgetauschten Daten. Zwei Techniken sind denkbar, die eine Abstraktion der Modulumgebung potentiell verständlicher machen. Erstens können die **Umgebung und die ausgetauschten Informationen eines Moduls durch Modelle oder die Beschreibung von Parallelen** dargestellt werden. In Modellen werden Datenflüsse zwischen Komponenten bspw. in Form von Pfeilen und die Funktion in Form von Kästen dargestellt. Als Parallele wird für hochautomatisierte Fahrzeuge meist der menschliche Fahrer herangezogen, sodass z. B. die Ausgaben eines Verhaltensplaners mit Entscheidungen eines menschlichen Fahrers verglichen werden. Zweitens sind die **ausgetauschten Informationen auch mit Szenarien auf Systemebene verknüpfbar**. Diese sind besser vorstellbar, sodass die Beschreibung der Informationen leichter fällt. Die Beschreibung der Verknüpfung zur Systemebene kann durch natürliche

[269] Bennaceur, A. et al.: Requirements Engineering (2019), S. 61–63.

[270] Bennaceur, A. et al.: Requirements Engineering (2019), S. 61–63.

Sprache, formelle Sprache sowie Strukturen oder Modelle erfolgen. Entsprechende Beschreibungen abstrahieren die Modulumgebung so, dass daraus das erzeugte Systemverhalten direkt ersichtlich und das Modulverhalten eindeutig spezifizier- und bewertbar ist.

Kreativitätstechniken benötigen einen möglichst weiten Spielraum, um neue Ideen hervorzurufen.[270] Eine bereits bestehende enge Definition eines Moduls wäre hierbei hinderlich. Daher benötigt diese Technik eine möglichst **allgemeine Beschreibung der Funktion oder des Anwendungsfalls** des Moduls. Ebenso sollte die **Umgebung des Moduls möglichst allgemein beschrieben** werden, ohne die Nennung konkreter Implementierungen und Designentscheidungen.

Die Analyse der Techniken von Bennaceur et al.[270] zeigen, dass diese unter den genannten Bedingungen auch auf Modulebene anwendbar sind. Die Ergebnisse der beschriebenen Analysen (einschließlich der folgenden Schnittstellen- oder Risikoanalyse in den nächsten Abschnitten) werden als Ziele aufgenommen. Diese sind zur Übersicht in Anhang A als Ziele G27a bis G27f aufgelistet. Irrtümer im Anforderungsspezifikationsprozess können allerdings dazu führen, dass Aspekte, die z. B. als Teil eines Risikoszenarios identifiziert sind, verloren gehen. Diese Ungewissheiten sind vermeidbar, indem Testfälle direkt z. B. aus identifizierten Risiken abgeleitet werden. Daher werden im weiteren Verlauf dieses Kapitels zusätzlich notwendige Informationen für Methoden der risikobasierten und schnittstellenbasierten Testfallgenerierung ermittelt.

Risikobasierte Testfallgenerierung

Die risikobasierten Testfallgenerierungsverfahren verwenden Risiken als Priorisierungskriterien für die Definition möglicher Szenarien, Bedingungen oder anderer Parameterkombinationen. Die Priorisierung und Herleitung der Tests erfolgt danach, wie wahrscheinlich ein Ereignis ist und wie hoch der Schweregrad bei Eintritt des ungewünschten Ereignisses erwartet wird. Testfälle, die einem hohen Schweregrad zugeordnet sind und eine hohe Eintrittswahrscheinlichkeit haben, werden zuerst getestet. Bei sicherheitsrelevanten Systemen findet die Identifikation von Gefährdungen bereits zu Beginn des Entwicklungsprozesses statt.[271] Zu jeder Gefährdung werden Sicherheitsziele definiert, die die Gefährdung vermeiden oder deren Wahrscheinlichkeit verringern sollen. Beispiele aus Forschungsprojekten zur Absicherung hochautomatisierter Fahrzeuge zeigen, dass die Definition von Sicherheitszielen teils sehr unterschiedliche Granularitäten in Bezug auf die betreffenden Komponenten eines Systems besitzen, als auch unterschiedliche Detailgrade in der Formulierung des Ziels aufweisen.[272] Klamann et al.[273] schlagen daher mit dem Ziel einer modularen Absicherung eine klare Zuordnung von Sicherheitszielen zu Modulen vor.

[271] ISO: ISO 26262 - Road vehicles – Functional safety (2018), Teil 2, S. 5.

[272] Stolte, T. et al.: Hazard analysis and risk assessment for automated vehicles (2017).

[273] Klamann, B. et al.: Pass-/Fail-Criteria for Particular Tests (2019).

Das Ziel die Sicherheitsziele eindeutig Modulen zuordnen zu können, legt nahe, die risikobasierten Testmethoden der Hierarchieebene entsprechend zu kategorisieren, auf der die Methoden angewendet werden. Ein Großteil der Methoden zur Testfallgenerierung ist jedoch auf verschiedenen Hierarchieebenen anwendbar (vgl. Klamann und Winner[274]).

Popovic und Vasic[275] geben einen umfassenden Überblick über 22 verschiedene Methoden der Gefährdungs- und Risikoanalyse. Sie teilen diese nach ihrem zeitlichen Einsatz im Entwicklungsprozess in sieben verschiedene Analysetypen ein. Der Zeitpunkt im Entwicklungsprozess hat jedoch nur indirekten Einfluss auf die Informationen, die zur Durchführung der Methoden notwendig sind. Daher wird für die folgende Analyse stattdessen auf die von Popovic und Vasic genutzte Einteilung in induktive und deduktive Methoden zurückgegriffen. Die Methoden zur Gefährdungs- und Risikoanalyse sind damit basierend auf ihrer unterschiedlichen Ausgangslage für die Einteilung notwendiger Informationen gruppiert. Die daraus abgeleiteten Informationen werden als Ziele für die Argumentationskette formuliert und sind zur Übersicht in Anhang A als Ziele G28a bis G28g aufgelistet.

Induktiv risikobasierte Methoden

Die induktiven Methoden identifizieren Gefährdungen und Risiken ausgehend von niedrigen Hierarchieebenen. Dabei wird z. B. davon ausgegangen, dass eine Funktion keine, ungewünschte oder fehlerhafte Informationen liefert, d.h. außerhalb der Spezifikation arbeitet. Anhand des spezifischen Fehlers wird analysiert, welche Ereignisse dadurch ausgelöst werden und ob diese eine Gefahr auf sicherheitsrelevanten Ebenen verursachen. Die Fehlermöglichkeits- und Einflussanalyse (FMEA) ist eine der gebräuchlichsten Methoden der induktiven Risikoanalyse, die auch von der ISO 26262[276a] empfohlen wird. Bei der FMEA werden Fehlerarten von Experten mit detaillierten Kenntnissen über das System oder Modul ermittelt. Diese Fehlerarten werden verwendet, um mögliche Ursachen (Fehlerzustände oder Irrtümer) und ihre Auswirkungen auf die Sicherheit (Versagen) zu finden und gleichzeitig ihre mögliche Schwere, Eintrittswahrscheinlichkeit und Entdeckungswahrscheinlichkeit zu bewerten. Für die induktiven Methoden ist daher eine Kenntnis über die **Implementierung, d.h. die verschiedenen Abläufe und Informationsflüsse innerhalb eines Moduls** notwendig, um potentielle Abweichungen von der Sollfunktion zu identifizieren. Mit Hilfe von **Erfahrungen und Daten aus der Entwicklung vorheriger Module, Vereinfachungen, die bei Berechnungen erfolgen, oder anderer bekannter Schwachstellen eines Moduls** lassen sich Testfälle ableiten, die diese Schwachstellen ansprechen und überprüfen. Die Norm ISO 26262 bezeichnet dies als Error-Guessing.[276b] Zusätzlich listet die Norm die Fault-Injection Tests auf, die Fehlerzustände simulieren und das Modul damit auf Robustheit ggü. **bekannter Fehlerzustände** oder die korrekte Implementierung von **Sicherheitsmechanismen** testen. Beim Stress-Test werden Systeme an ihre maximale Leistungsfähigkeit gebracht

[274] Klamann, B.; Winner, H.: Analyse von Dekompositionsprozessen zur modularen Absicherung (2022).

[275] Popovic, V. M.; Vasic, B. M.: Review of hazard analysis methods (2008).

[276] ISO: ISO 26262 - Road vehicles – Functional safety (2018), a: Teil 3, S. 6, b: Teil 12, S. 19-20.

und ggf. überlastet, um ihre Robustheit zu prüfen.[276b] Daher muss die **Leistungsfähigkeit des Moduls,** ein **zugehöriges Leistungsmaß** sowie die **Kriterien für die notwendige Leistung** (ggf. in Abhängigkeit der Szenarien) bekannt sein, um solche Tests abzuleiten. Zur Identifikation von Versagensfällen auf Systemebene durch multiple Fehlerzustände unterschiedlicher Module müssen die bekannten **Fehlerzustände aller Module und die zugehörigen Bedingungen** allen anderen Modulen zugänglich sein.

Deduktiv risikobasierte Methoden

Bei deduktiven Methoden wird eine mögliche Fehlerwirkung in mögliche Ursachen zerlegt. Eine weit verbreitete Methode hierfür ist die Fehlerbaumanalyse (FTA), die ausgehend von einem ungewünschten Ereignis die Ursachen über die Ableitung von Zwischenereignissen anhand einer Baumstruktur ermittelt. Die letzte Ursache ist der ermittelte Fehlerzustand, der sich in der Regel auf Komponenten- oder Bauteilebene befindet. Die potentiellen Fehlerwirkungen gehen in deduktiven Verfahren von höheren Hierarchieebenen aus. Daher muss auch für die Absicherung von Modulen der **Anwendungsfall des übergeordneten Systems** und **davon ausgehende Abweichungen vom Sollverhalten** bekannt sein. Für hochautomatisierte Fahrzeuge kann z. B. der Bruch eines Sicherheitsziels in einem bestimmten Szenario als Ausgangslage für eine Fehlerwirkung herangezogen und auf die Modulebene heruntergebrochen werden (vgl. Klamann et al.[277]). Abweichungen vom Sollverhalten sind teils nur im Kontext, d.h. im spezifischen Szenario als solche Abweichungen bewertbar. Daher ist es auch notwendig, die **relevanten Szenarien auf Systemebene** zu erfassen. Zur Ableitung der Fehlerwirkung wird, wie für die induktive Vorgehensweise Wissen über die **Abhängigkeiten im System** benötigt, mit denen sich die verschiedenen Ursachen nachverfolgen lassen, wie ebenfalls Klamann et al.[277] vorstellen. Hierzu lassen sich Architekturbeschreibungen z. B. in Form von Blockdiagrammen nutzen. Dajsuren[278] nennt insgesamt sechs Sichtweisen für Architekturen, die in der Automobilindustrie wichtige Bestandteile der Entwicklung und Absicherung sind. Diese betreffen jeweils verschiedene Stakeholder und Abhängigkeitstypen (vgl. Kapitel 2.2). Für das Forschungsprojekt UNICAR*agil* analysiert Kohls[279] für eine Risikoanalyse die Funktions- und Softwarearchitektur auf existierende Abhängigkeiten und unterscheidet dabei neun verschiedene Typen. Die Abhängigkeitstypen werden im Gegensatz zu verschiedenen Architektursichten innerhalb des gleichen Graphen dargestellt. Dies verringert bei komplexen Systemen ggf. die Übersichtlichkeit, macht Interdependenzen zwischen Architektursichten jedoch sichtbarer. Die **Einteilung der Abhängigkeiten in verschiedene Typen** (siehe bereits G13) hilft außerdem bei der Identifikation von Abhängigkeiten sowie der Nachverfolgung der Wirkketten z. B. ausgehend von einer Fehlerwirkung.

[277] Klamann, B. et al.: Pass-/Fail-Criteria for Particular Tests (2019).

[278] Dajsuren, Y.: Defining Architecture Framework for Automotive Systems (2019), S. 161.

[279] Kohls, S. K.: Master Thesis, Abhängigkeiten zwischen Modulen (2022), S. 42–45.

STPA

Die Systemtheoretische Prozessanalyse (STPA) lässt sich nicht eindeutig den deduktiven oder induktiven Verfahren zuordnen. Es werden zuerst Verluste und Gefährdungen auf Systemebene definiert, sodass diese für Analysten gut vorstellbar sind. Die darauffolgende Analyse geschieht jedoch auf Komponentenebene, indem auf Basis einer Kontrollstruktur mit den einzelnen Funktionen oder Komponenten mögliche Kausalfaktoren ermittelt werden, die schlussendlich Verluste und Gefährdungen auslösen.[280a] Die STPA versucht damit die Schwäche konventioneller Gefährdungs- und Risikoanalysen, die eine verlustfreie Zerlegung eines übergeordneten Problems annehmen, zu umgehen. Verlustfrei bedeutet, dass die Analyse der kleineren Probleme die gleichen Ergebnisse liefert, wie die Analyse des größeren Problems. Der Fokus der STPA liegt damit nicht auf der Fehlfunktion einzelner Komponenten, sondern der Interaktion dieser Komponenten miteinander.[280b]

Die in STPA aufgestellte Kontrollstruktur erfordert erneut Wissen über die **Abhängigkeiten zwischen Komponenten und Funktionen** im System. Bei der Analyse der Kontrollaktionen innerhalb der Kontrollstruktur (d.h. der Schnittstellen dieser Architektur) sollen die Auswirkungen auf das Gesamtsystem unter verschiedenen Bedingungen (z. B. fehlerhafte Informationsübertragung) bewertet werden. Dies bedeutet die Notwendigkeit des Verständnisses der Abläufe im System und der dadurch ausgelösten Auswirkungen auf das Systemverhalten.

STPA macht grobe Vorgaben in Form von vier Kategorien an unsicheren Kontrollaktionen. Zur Ableitung unsicherer Kontrollaktionen eines Moduls sind detaillierte Beschreibungen des **Modulverhaltens** und Kenntnisse über deren **Übertragung auf Systemebene** notwendig. Zur Analyse der ersten Kategorie, ob eine ausbleibende Kontrollaktion eine Gefahr auslöst, muss bekannt sein, wie die weitere Wirkkette an Modulen auf **ausbleibende Informationen aus dem betrachteten Modul reagiert**. Die zweite Kategorie zielt auf Kontrollaktionen, die eine Gefahr hervorrufen. Hierzu zählen neben fehlerhaften Kontrollaktionen auch solche, die trotz Einhaltung der Spezifikation ein unsicheres Verhalten auf Systemebene auslösen. Daher müssen **potentiell fehlerhafte Kontrollaktionen** eines Moduls bekannt bzw. identifizierbar sein. Darüber hinaus müssen auch hier **Reaktionen des Systems auf Kontrollaktionen** eines Moduls vorausgesagt werden können. Für die dritte Kategorie ist es notwendig, die Reaktionen des Systems auf **zu frühe oder zu späte Kontrollaktionen** zu kennen. Hierfür ist es hilfreich, wenn die Zeitangaben darüber bekannt sind, **wie zu früh oder zu spät eine Kontrollaktion** auftritt und **unter welchen Randbedingungen** dies auftreten kann. Mit Kenntnis der Randbedingungen lassen sich potentielle Auswirkungen genauer abschätzen. Die vierte Kategorie in STPA erfordert Kenntnisse darüber, welche **Auswirkungen zu kurz oder zu lang applizierte Kontrollaktionen** auf ein

[280] Leveson, N. G.; Thomas, J. P.: STPA Handbook (2018), a: -, b: S. 5-6.

System haben. Auch hier ist es hilfreich **Zeitangaben und Randbedingungen bzw. Ursachen** über zu kurz oder zu lang applizierte Kontrollaktionen zu kennen, um potentielle Auswirkungen zu prädizieren.

Schnittstellenbasierte Testfallgenerierung

Die Generierung von Testfällen auf der Grundlage von Schnittstellen nutzt Schnittstellenbeschreibungen, um Testfälle strukturell abzuleiten. Schnittstellen können je nach Architektur unterschiedlich beschrieben sein. Gemeinsam haben sie, dass sie in der jeweiligen Architektursicht die Verbindungseigenschaften oder die Interaktionen zwischen den Entitäten jener Architektursicht beschreiben.[281] Die **Schnittstellenbeschreibung** muss hierzu die **Struktur der Informationen (die Syntax)** und **die Bedeutung dieser Informationen (die Semantik)** beinhalten. Darauf aufbauend werden die Kombinationen der Schnittstellenparameter, also die möglichen Stimuli, zur Festlegung konkreter Testfälle definiert.

Das schnittstellenbasierte Testen stützt sich auf die formal beschriebene Schnittstellenspezifikation und verwendet strukturierte oder statistische Ansätze zur Generierung konkreter Testfälle. Die Hauptvorteile sind daher, dass die Tests im Vergleich zu anderen Methoden der Testfallgenerierung ab diesem Punkt objektiv sind. Darüber hinaus ist die Testabdeckung leicht messbar und mit den spezifizierten Wertebereichen einstellbar. Wie das Beispiel des szenariobasierten Testens (vgl. Kapitel 3.2) zeigt, werden auf Systemebene die möglichen Eingänge zum System als Schnittstellen betrachtet und in Testfällen variiert. Zusätzlich wird das Fahrverhalten des Fahrzeugs variiert, so dass verschiedene Verhaltenselemente, z. B. unterschiedliche Manöver oder Reaktionen, beobachtet werden können. Auf modularer Ebene ist die Variation von Parametern bei schnittstellenbasierten Tests in der Regel auf die Eingänge beschränkt. Dies kann jedoch gleichermaßen auf Ausgaben angewendet werden. Bestimmte Ausgaben (ob absichtlich oder im Falle von Fehlerzuständen oder Irrtümern) treten ansonsten ggf. nur bei ungewöhnlichen, spezifischen Eingaben auf.

Eine hohe Testabdeckung der Ein- und Ausgänge durch Wahl einer feinen, gleichverteilten Diskretisierung der Parameterkombination ist als eher ineffizient einzuschätzen, da nicht erwartet wird, dass Fehlerzustände gleichmäßig über den Parameterraum verteilt sind. Daher ist das risikobasierte Testen vorzuziehen, um Bereiche mit einer höheren erwarteten Fehlerwahrscheinlichkeit abzudecken. Nalic et al.[282] stellen hierzu eine beispielhafte Methode zur Absicherung hochautomatisierter Fahrzeuge vor. Allgemein sind solche Methoden auch als „Important Sampling“ bekannt.[283] Die Methode des Importance Sampling bietet eine mathematische Beschreibung, um den risikobasiert generierten Testdatensatz z. B. mit realen Testdaten zu vergleichen. Die mathematische Beschreibung kann die statistische Äquivalenz zwischen beiden Testsätzen zeigen.[283]

---

[281] Sanchez, R.: Integrating technology strategy and marketing strategy.

[282] Nalic, D. et al.: Stress Testing Method for Scenario-Based Testing of Automated Driving Systems (2020).

[283] Zhao, D.: Dissertation, Accelerated Evaluation of Automated Vehicles (2016), S. 45–48.

Der bedeutendste Nachteil der schnittstellenbasierten Tests für eine modulare Absicherung ist die Abhängigkeit von der korrekten Spezifikation der Schnittstellen. Bei einer modularen Absicherung werden fehlerhaft spezifizierte Schnittstellen nicht mehr durch Integrationstests aufgedeckt. Gleichermaßen können Parameter, die von einer spezifizierten Schnittstelle des Systems nicht berücksichtigt werden, zu unsicherem Systemverhalten führen. Beispiel 4-5 beschreibt hierzu ein Szenario für eine fehlerhaft spezifizierte Schnittstelle für ein hochautomatisiertes Fahrzeug.

Beispiel 4-5: Fehlerhaft spezifizierte Schnittstelle im Bereich hochautomatisierter Fahrzeuge.

Falls Schnee in einer ODD spezifiziert ist, wird ein Fahrdynamikregler voraussichtlich durch Variation des Reibkoeffizienten getestet. Existiert in dem System jedoch keine Schnittstelle zur Bereitstellung der Information, dass es schneit, ist das Systemverhalten unter Schnee zumindest ungewiss. Erkennt das Planungsmodul aufgrund dessen nicht oder zu spät, dass es schneit, führt dies zu einer späteren (ggf. zu späten) Anpassung der Trajektorie auf die verringerten Beschleunigungsgrenzen.

Zusammengefasst sind schnittstellenbasierte Testmethoden ohne weitere Anpassungen zur Verifizierung, nicht aber zur Validierung geeignet. Die Informationen zur Ableitung schnittstellenbasierter Tests sind daher notwendig, aber nicht hinreichend für eine modulare Absicherung. Gegenüber spezifikations- und risikobasierter Testfallgenerierung können für die schnittstellenbasierte Testfallgenerierung objektive und konkrete Informationen abgeleitet werden. Die Generierung und Durchführung von Tests ist außerdem aufgrund der Objektivität gut automatisierbar. Anhand der vorigen Erläuterungen lassen sich folgende Informationen auflisten, die zur schnittstellenbasierten Testfallgenerierung erforderlich sind. Diese sind für die Argumentationskette als Ziele G29a bis G29g formuliert auch in Anhang A aufgelistet:

- Parameterwerte und -kombinationen an den Modulschnittstellen
- Einflussfaktoren auf die Parameterwerte
- Sensitivität des Moduls ggü. Parameterwertänderungen
- Äquivalenter Wertebereiche bzw. Leistungsklassen anhand der Einflussfaktoren und Sensitivität
- Grenzwerte bzw. maximale und minimale Leistungsfähigkeit eines Moduls
- Verbindung der Äquivalenzklassen und Grenzwerte auf Modulebene mit Szenarien auf Systemebene
- Verbindung der Äquivalenzklassen und Grenzwerte auf Systemebene (bspw. in Form systematisch variierter Szenarien wie von Schuldt et al.[284] demonstriert) mit Parameterkombinationen bzw. Werteverläufen auf Modulebene
- Bestimmung äquivalenter Parameterwerte auch für die Ausgaben eines Moduls

[284] Schuldt, F. et al.: Systematic Test Case Generation for ADAS (2018), S. 169–170.

Ergänzend zu Abbildung 4-12 zeigt Abbildung 4-13 die vervollständigte GSN Argumentationskette für den zweiten Pfad, der die Vollständigkeit von Modultests argumentiert. Blaue Kreise am Ende der Ziele G25, G27, G28 und G29 verdeutlichen, dass Beiträge zu deren Lösung in Kapitel 5 vorgestellt werden. Hierzu sind die dafür in den vorigen Kapiteln 4.3.3.1 und 4.3.3.2 entwickelten Ziele in Anhang A aufgelistet.

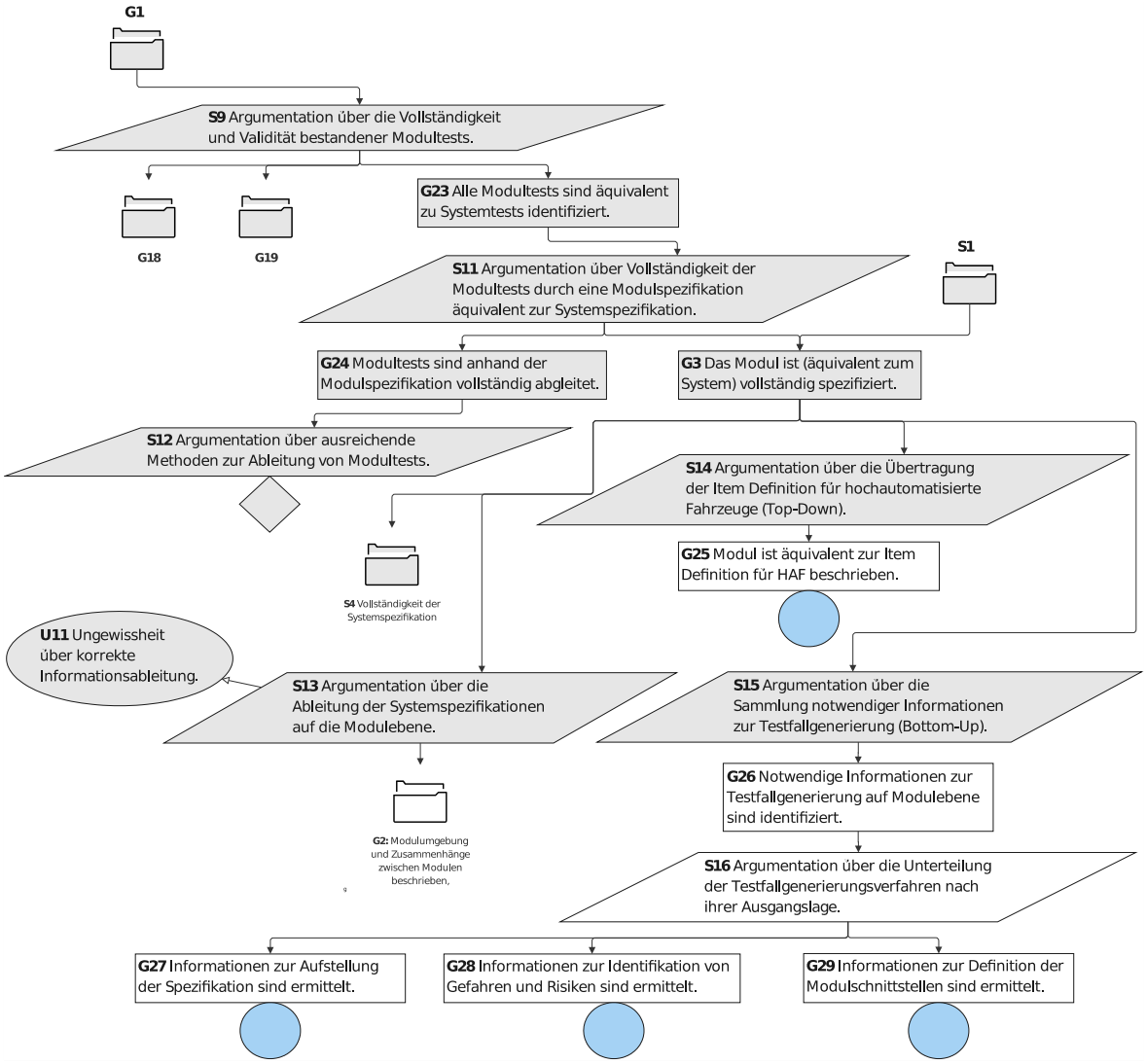


Abbildung 4-13: Argumentation der Vollständigkeit der Modultests im zweiten Pfad der GSN Argumentationskette die ergänzend zu grau hervorgehebenen Elementen nach Abbildung 4-12.

4.4 Argumentation über Abwesenheit von Ungewissheiten im Dekompositionsprozess[285]

Die vorigen Kapitel 4.2 und 4.3 decken in der bisherigen Argumentationskette an mehreren Stellen Ungewissheiten bei der Zerlegung des Systems in Module auf. Ungewissheit U3 weist bereits auf den höheren Abstraktionsgrad durch Dekomposition hin, U4 greift die Ungewissheit über emergente Eigenschaften des Systems auf und U11 stellt die korrekte Informationsableitung durch Nutzung der spezifizierten Architekturen des Systems in Frage, die bisher ungelöst bleibt. Das folgende Kapitel analysiert unter der Strategie S17 daher die Entstehung von Ungewissheiten im Dekompositionsprozess. Anhand potentieller Irrtümer werden auf Basis des Ziels G30, das Ungewissheiten im Dekompositionsprozess identifiziert, allgemeine Ziele zur Vermeidung der Irrtümer definiert. Mit diesen wird die Argumentationskette, wie in Abbildung 4-21 dargestellt, vervollständigt.

Ein System wird allgemein durch die Dekomposition in kleinere Bestandteile, z. B. Module zerlegt. Darauf aufbauend werden alle Informationen auf Systemebene auch auf die Modulebene heruntergebrochen, um jedes Modul in der Modulspezifikation zu beschreiben. Dies ist beispielhaft in Abbildung 4-14 dargestellt. Darin sind beispielhafte Ergebnisse der Prozessschritte innerhalb des V-Modells in grauen, eckigen Kästen aufgeführt. Gestrichelte Pfeile zeigen den Fluss dieser Ergebnisse zur Generierung von Modultests. Die "Architektur"-Kästen mit gestrichelter Begrenzungslinie verdeutlichen, dass die Dekomposition zur Ermittlung aller dargestellten Ergebnisse, die zuvor spezifizierte Architektur direkt oder indirekt verwendet. Das Gleiche gilt für den potentiellen Ansatz aus bereits definierten konkreten Integrations- oder Systemtests, Modultests abzuleiten. Das Systems-Engineering nimmt bei solchen Dekompositionsprozessen an, dass die dekomponierten kleineren Einheiten zusammen die Eigenschaften der größeren Einheit erzeugen.[286] Die aggregierte Funktionalität wird in der Regel durch Integrations- und Systemtests verifiziert. Bei einer modularen Absicherung ist es erforderlich, dass diese Tests nicht mehr notwendig sind. Daher ist eine Reduktion der Ungewissheit des Dekompositionsprozesses nur durch modulare Tests oder durch die Vermeidung von Irrtümern im Dekompositionsprozess möglich.

---

[285] Teile diese Kapitels wurden bereits in Klamann, B.; Winner, H.: Comparing Different Levels of Technical Systems (2021) und Klamann, B.; Winner, H.: Analyse von Dekompositionsprozessen zur modularen Absicherung (2022) veröffentlicht.

[286] Schäuffele, J.; Zurawka, T.: Automotive Software Engineering (2016), S. 127–128.

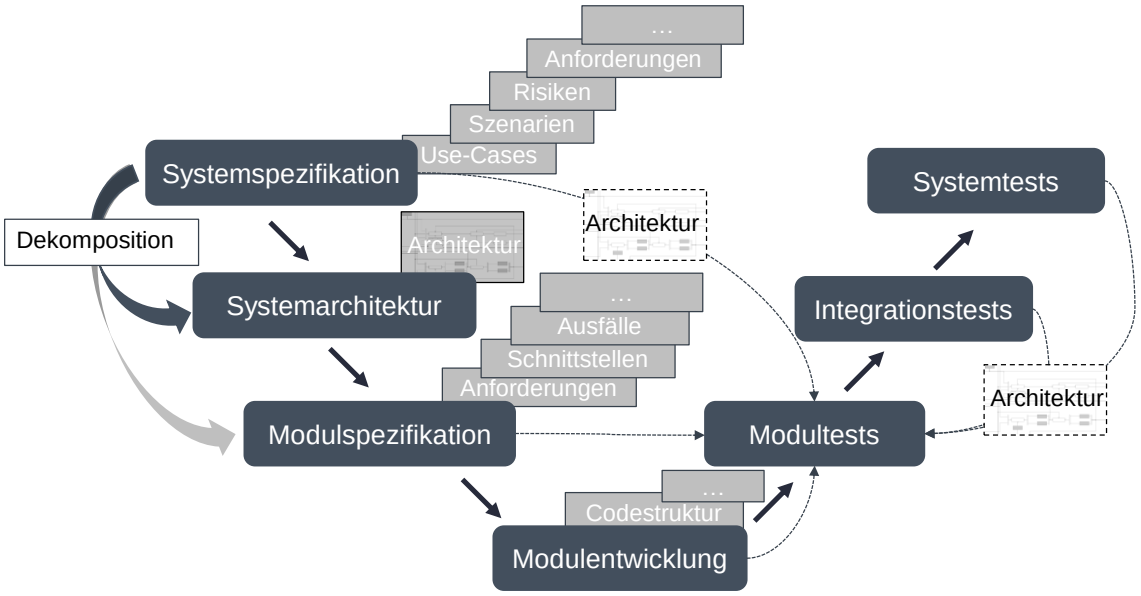


Abbildung 4-14: Verwendung der Dekomposition und der resultierenden Architektur innerhalb des V-Prozesses.[287]

Vorwiegend werden im Folgenden die Dekomposition der Systemfunktionen und die Dekomposition in Hardware- und Softwareteile betrachtet. Diese leiten die Architektursichten ab, die z. B. nach der Norm ISO 26262 hauptsächlich für die Absicherung verwendet werden. Die in der Dekomposition gewählten Funktionsgrenzen hängen von den möglichen logischen Zusammenhängen ab. Die zerlegten Funktionen sollen sich dabei logisch zu einer übergeordneten Funktion zusammensetzen lassen.

4.4.1 Methode zur Identifikation von Irrtümern im Dekompositionsprozess

Normen wie die ISO 26262 beschreiben nicht, wie das Gesamtsystem oder die Systemeigenschaften zerlegt werden sollen. Obwohl die Norm ISO 26262 vorgibt, die Item-Definition auf Fahrzeugebene zu beschreiben, erfordert die im Prozess folgende Gefährdungsanalyse Kenntnisse über die konkrete Systemarchitektur. Dies wird zumindest anhand der gegebenen Beispiele in der Norm deutlich.[288] Schäuffele et al.[289] weisen darauf hin, dass der

[287] Abgerundete Kästchen sind Aktivitäten, nicht abgerundete Kästchen sind die Ergebnisse der Aktivitäten. "Architektur"-Kästen mit gestrichelter Begrenzungslinie verdeutlichen, dass eine Beschreibung der Architektur zur Ableitung der Informationen erforderlich ist. Durchgehende Pfeile sind Prozessabläufe, gestrichelte Pfeile stellen die Verwendung von Ergebnissen einer Aktivität in einer anderen dar.

[288] ISO: ISO 26262 - Road vehicles – Functional safety (2018), part 3.

[289] Schäuffele, J.; Zurawka, T.: Automotive Software Engineering (2016), S. 129.

durchgeführte Dekompositionsprozess (explizit in Bezug auf Hierarchiebildung und Modularisierung) meist intuitiv ist. So ergibt sich nach bestehenden Prozessen bei Modulentwicklung und Modultest kein durchgängig strukturiertes Vorgehen.

Im folgenden Unterkapitel wird die Methode zur Ableitung von Regeln zur Vermeidung von Irrtümern durch die Verwendung von Modulen anstelle des Systems erläutert. Anhand einer Identifikation von Ungewissheiten während des Dekompositionsprozesses lassen sich entsprechende Irrtümer ableiten. In dieser Arbeit wird, wie bereits von Leveson[290] vorgeschlagen, die Verfeinerung oder Zerlegung von Informationen (z. B. Anforderungen oder Parameterräume) in Zusammenhang mit Funktion, Hardware und Software analysiert. Andere Entwicklungsschritte, die keine Informationen von der System- zur Modulebene ableiten werden an dieser Stelle nicht analysiert.

Zur Erreichung einer modularen Absicherung wird angenommen, dass die Durchführung von Modultests nicht ausreicht und daher insbesondere bei komplexen Funktionen eines automatisierten Fahrzeugs auch die Beherrschung des Entwicklungsprozesses zwingend erforderlich ist. Der folgende Ansatz ist daher eine ganzheitliche Betrachtung der Entwicklung sowie des Testvorgehens. Es werden jedoch nur mögliche Irrtümer der Dekomposition, nicht aller Schritte des Entwicklungsprozesses betrachtet. Die konkreten Irrtümer hängen außerdem von der konkreten Umsetzung des Entwicklungsprozesses ab. Diese Prozesse sind je nach Bereich und Unternehmen sehr unterschiedlich. Daher erfolgt die Analyse lediglich auf Basis eines generischen Entwicklungsprozesses nach V-Modell, sodass sich daraus generische Irrtümer ableiten.

Die Identifikation von Irrtümern erfolgt mit Hilfe einer Fehlerbaumanalyse[291], die von einem unerwünschten Top-Ereignis ausgeht, das in Ursachen zerlegt wird, die zu diesem Ereignis führen können. Diese Ursachen werden als weitere unerwünschte Ereignisse betrachtet und in weitere Ursachen zerlegt. Dies wird so lange wiederholt, bis eine Ebene erreicht ist, auf der die Ursachen noch auf eine generalisierte Systemarchitektur eines HAFs anwendbar sind. Ursachen in einer Fehlerbaumanalyse sind Fehlerzustände des Top-Level-Ereignisses. Die Definition der Begriffe hängt jedoch von der eingenommenen Perspektive ab. Bei der Perspektive und dem Umfang dieser Analyse sind die ermittelten Ursachen Irrtümer, die auf den Dekompositionsprozess zurückzuführen sind (vgl. hierzu auch die Terminologien in Kapitel 2.1). Zur Ergänzung des Fehlerbaums werden wie bei der systemtheoretischen Prozessanalyse (STPA) von Leveson et al.[292] vereinfachte Kontrollstrukturen modelliert, um Irrtümer zu identifizieren. Für Irrtümer bei der Modulspezifikation entspricht die Kontrollstruktur einer Funktionsarchitektur und für Irrtümer bei Modultests einer Testumgebung. Die durch die Fehlerbaumanalyse identifizierten Irrtümer werden den modellierten Entitäten

[290] Leveson, N.: Engineering a safer world (2012), S. 191.

[291] Vesely, W. E.: Fault tree handbook (1981), S. IV-1.

[292] Leveson, N. G.; Thomas, J. P.: STPA Handbook (2018).

oder Verbindungen der jeweiligen Kontrollstruktur zugeordnet. Entitäten oder Verbindungen, denen kein Irrtum zugeordnet ist, werden genauer auf mögliche Irrtümer durch den Dekompositionsprozess untersucht. Neu aufgedeckte Irrtümer werden daraufhin dem Fehlerbaum hinzugefügt, um dessen Vollständigkeit zu erhöhen. Die folgende Analyse erscheint aufgrund der Betrachtung ähnlicher Informationen über Module und das System insgesamt als eine rückwärtige Betrachtung der identifizierten Ziele der Kapitel 4.2 und 4.3. Es gilt jedoch zu beachten, dass im Folgenden nur auf Irrtümer durch Dekomposition Bezug genommen wird. Die Strategie S18 argumentiert, dass Irrtümer anhand von nicht erreichten Prozessergebnissen der Dekomposition identifizierbar sind. Die Strategie schließt damit die Lücke zwischen Informationen auf Systemebene und den daraus abgeleiteten Informationen auf Modulebene.

4.4.2 Irrtümer durch Dekomposition

Auf Systemebene wird gemäß der Definition der Norm ISO 26262 eine funktionale Beschreibung, z. B. auf der Grundlage möglicher Anwendungsfälle des Systems, gegeben. Diese Funktionsbeschreibung kann bereits in eine Reihe von Funktionen zerlegt werden, woraus eine funktionale Architektur entsteht. Zusätzlich lassen sich mit der funktionalen Beschreibung Anforderungen ableiten. In der Norm ISO 26262[293] werden diese Anforderungen auf Fahrzeugebene als Sicherheitsziele definiert. Zur dritten wichtigen Informationskategorie zählen Gefährdungen, die z. B. durch Szenarien beschreibbar sind, in denen sich eine Gefährdung zu einem Schaden manifestieren kann. Im Bereich der automatisierten Fahrzeuge werden diese Szenarien als kritisch bezeichnet. Westhofen et al.[294] beschreiben Kritikalitätsmetriken zur Einordung, wie kritisch ein Szenario ist. Klamann et al.[295] und Stolte et al.[296] zeigen wie Gefährdungen, die sich z. B. ausgehend von verletzten Sicherheitszielen ableiten, auf die Modulebene herunterbrechen lassen. Dennoch können solche Dekompositionsprozesse ebenfalls fehlerhaft sein und werden daher im Folgenden analysiert.

Für die Argumentationskette unter Strategie S18 werden zwei Ziele identifiziert. Das erste Ziel G31 ist, dass Irrtümer durch Dekomposition bei der Modulspezifikation identifiziert werden. Dieses Ziel leitet sich von der Anforderung ab, dass die Funktion des Moduls zusammen mit den Funktionen der anderen Module die Anforderungen auf Systemebene erfüllt. Zusätzlich ist zur Erreichung des zweiten Ziels G32 die Identifikation von Irrtümern durch Dekomposition bei Generierung, Durchführung und Auswertung von Modultests notwendig. Dieses Ziel leitet sich von der Anforderung ab, dass die korrekte Darstellung

[293] ISO: ISO 26262 - Road vehicles – Functional safety (2018), Teil 1, S. 22.

[294] Westhofen, L. et al.: Criticality Metrics for Automated Driving (2022).

[295] Klamann, B. et al.: Pass-/Fail-Criteria for Particular Tests (2019).

[296] Graubohm, R. et al.: Functional Safety Concept of Automated Driving (2019).

spezifizierter Funktionen des Moduls durch dessen Implementierung erforderlich ist. Ausgehend von den Irrtümern U3, U4 und U11 zeigt Abbildung 4-15 die GSN Argumentationskette bis zu diesen zwei Zielen zur Identifikation von Irrtürmen im Dekompositionsprozess.

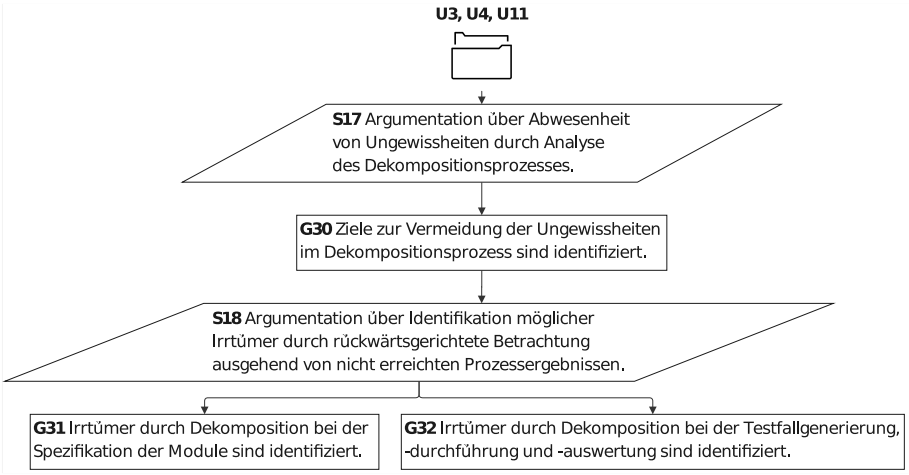


Abbildung 4-15: Abwesenheit von Ungewissheiten im Dekompositionsprozess als Ausgangspunkt im dritten Pfad der Argumentationskette.

4.4.2.1 Irrtümer aufgrund unzureichender Modulspezifikation

Zur Ableitung möglicher Gefährdungen wird beim Ansatz der Fehlerbaumanalyse das Top-Ereignis genutzt, dass ein Modul nicht ausreichend abgesichert ist. Auf der zweiten Ebene wird davon ausgegangen, dass mindestens eines der beiden Ziele nicht erreicht wird. Die erste Ursache besteht also darin, dass die *spezifizierte Funktion des Moduls zusammen mit den Funktionen der anderen Module die Anforderungen auf Systemebene nicht erfüllt*. Der Fehlerbaum für diesen Irrtum A und der abgeleiteten Ursachen ist in Abbildung 4-16 vollständig dargestellt. Alle Mehrfachverknüpfungen sind ODER-Gatter, weshalb auf das zugehörige Symbol in den Fehlerbäumen verzichtet wird. Die jeweiligen Ursachen sind als weitere mögliche Arten von Irrtümern entsprechend der Reihenfolge im Fehlerbaum durchnummeriert und werden in den folgen Absätzen systematisch hergeleitet.

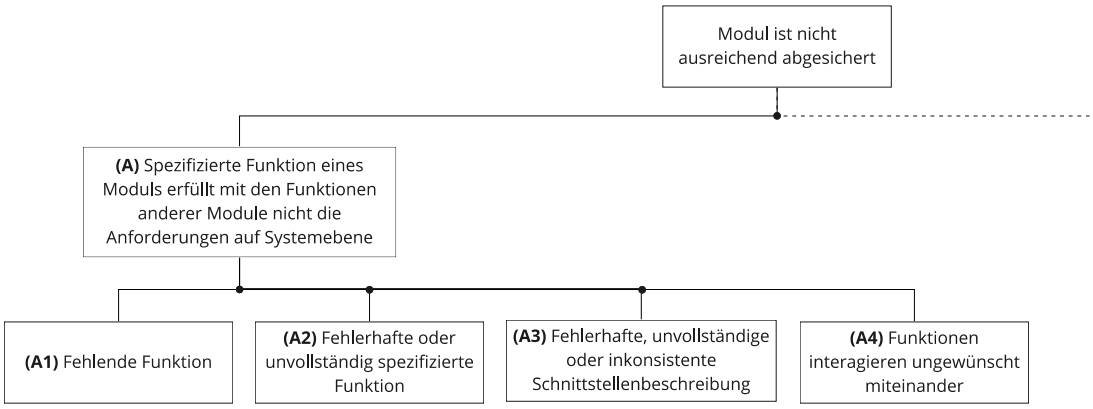


Abbildung 4-16: Baumstruktur des Irrtums A.

Die Herleitung der Irrtümer wird durch eine vereinfachte funktionale Architektur eines Systems mit einem Eingang und einem Ausgang in Abbildung 4-17 veranschaulicht und unterstützt somit die weitere Ableitung der Irrtümer. Ein Irrtum der spezifizierten Funktion bedeutet, dass die implementierte Funktion des Moduls wie spezifiziert arbeitet, aber die Spezifikation der Funktion fehlerhaft ist. In der dargestellten funktionalen Architektur sind die Irrtümer ihrem Entstehungsort zugeordnet, durch Kästchen visualisiert und mit dem Buchstaben des übergeordneten Irrtums und einer durchnummerierten Zahl entsprechend dem Fehlerbaum versehen. Das System besteht aus Modul 1 und Modul 2, die die Funktionen F1, F2 und F3 darstellen, die Informationen empfangen und senden.

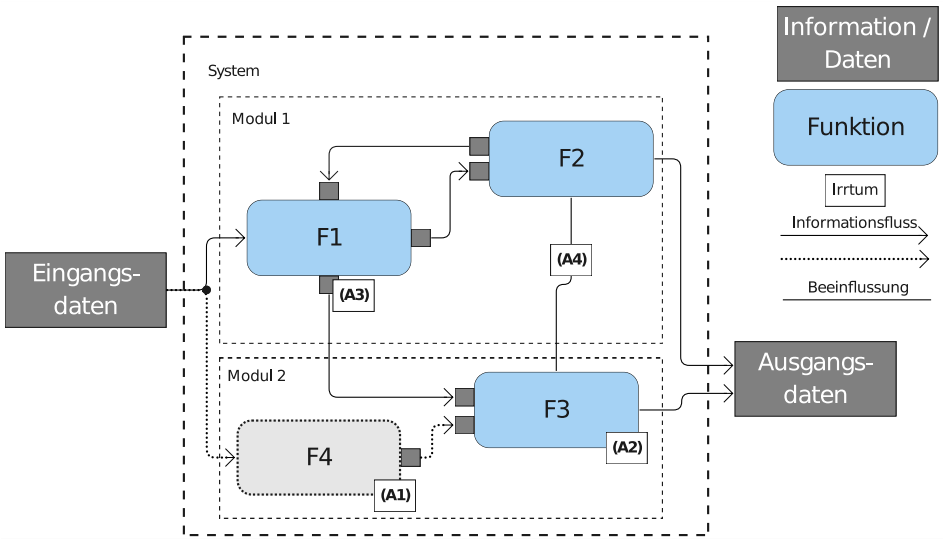


Abbildung 4-17: Beispielhaft abstrahierte funktionale Architektur mit den Funktionen F1 bis F4 innerhalb der System- und Modulgrenzen, dargestellt als UML-Aktivitätsdiagramm.

Die Funktion F4 in Abbildung 4-17 wird durch ein graues Rechteck visualisiert, um eine fehlende Funktion als Irrtum A1 darzustellen. Diese Funktion ist bspw. in einem Großteil möglicher Szenarien nicht zwingend erforderlich, um die Gesamtanforderungen zu erfüllen. In bestimmten Szenarien führt deren Fehlen jedoch zu einer Abweichung vom Sollverhalten. Dadurch ist dieser Irrtum auch in Modultests nur in diesen bestimmten Szenarien aufdeckbar.

Beispiel 4-6: Fehlende Funktion unter einem Irrtum A1.

Die von Homolla et al.[297] beschriebene initial fehlende Funktion einer Pose[298]-Offset-Korrektur in der UNICAR*agil*-Architektur hätte bspw. zu einer zunehmenden Abweichung der Steuerung bei inkonsistenten Lokalisierungsdaten geführt. Die vereinfachte Funktionsarchitektur von UNICAR*agil*, die diesen beispielhaften Irrtum zeigt, ist in Abbildung 4-18 dargestellt. In der Architektur verwendet die Trajektorienplanung Lokalisierungsdaten x(t) aus einer kamerabasierten Quelle, wie von Buchholz et al.[299] beschrieben. Die

[297] Homolla, T. et al.: Verfahren zur Korrektur von inkonsistenten Lokalisierungsdaten (2020).

[298] Die Pose beschreibt Position und Ausrichtung eines Objekts im Raum.

[299] Buchholz, M. et al.: Automation of the UNICARagil vehicles (2020), S. 1541–1542.

Trajektorienregelung verwendet die Lokalisierungsdaten x(t) aus dem Modul Fahrdynamikzustandsschätzung (FZS), das auf den Sensordaten des globalen Navigationssatellitensystems (GNSS), der Odometrie und einer Inertialmesseinheit (IMU) basiert. Die hinzugefügte Funktion Pose-Offset-Korrektur eliminiert den Offset zwischen beiden Lokalisierungsdaten zu x(t) (korrigiert). Bei Abweichungen eines der Lokalisierungseingänge bleibt der Ausgang der Offsetkorrektur stabil. Somit wird sichergestellt, dass die Position und Ausrichtung eingeregelt werden, die von der Trajektorienplanung auf Basis der kamerabasierten Lokalisierung ermittelt werden. Die FZS wird für den Trajektorienregler genutzt, um eine ausreichend hohe Aktualisierungsrate der Zustandsgrößen zu erlauben. Die kamerabasierte Lokalisierung ist dazu nicht fähig. Weitere Erläuterungen und Vorteile dieser Architektur werden von Homolla et al.[297] beschrieben.

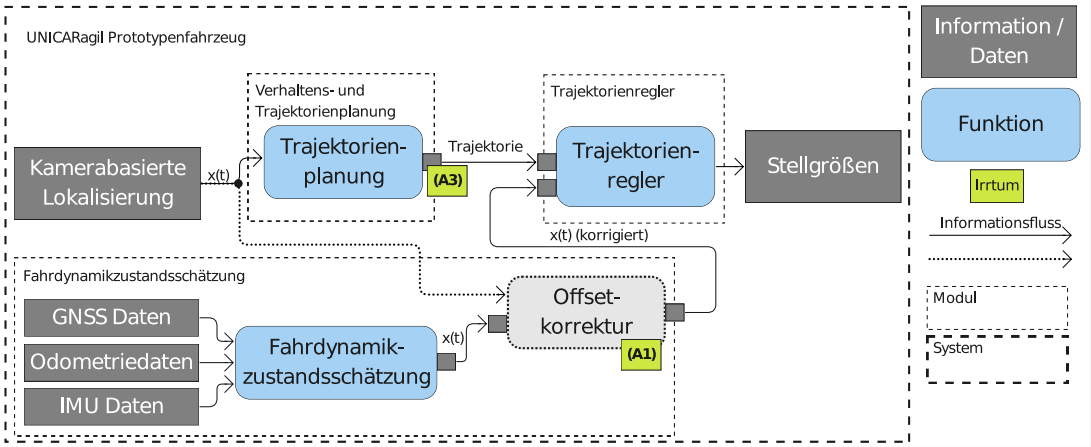


Abbildung 4-18: Vereinfachte Funktionsarchitektur ohne Umfeldwahrnehmung mit kartenbasierter Trajektorienplanung der UNICAR*agil*-Prototypen mit der ergänzten Funktion Offsetkorrektur.[297]

Der Irrtum A2 beschreibt eine fehlerhafte oder unvollständig spezifizierte Funktion, die dadurch die korrekte Funktion des Moduls bzw. des Systems verhindert. Ein einfaches Beispiel hierfür ist die Anforderung auf Systemebene, die maximale Geschwindigkeit einzuhalten, die durch eine Drehzahlbegrenzung als Funktion umgesetzt wird. Ist diese Begrenzung jedoch mit einer falschen Getriebeübersetzung berechnet, kann die Geschwindigkeitsbeschränkung auf Systemebene überschritten werden, obwohl die Anforderung auf Modulebene erfüllt ist.

Ebenso kann mit Irrtum A3 nur die Schnittstellenbeschreibung fehlerhaft, unvollständig oder inkonsistent in sich oder zu anderen Schnittstellen sein. Ein beispielhafter konkreter Irrtum ist die Verwendung unterschiedlicher Koordinatensysteme, sodass Informationen in der weiteren Verarbeitung eines anderen Moduls mit einem anderen Koordinatensystem fehlerhaft referenziert werden. Schnittstellen können darüber hinaus ungewünschte Interaktionen auslösen, die unter Irrtum A4 im Dekompositionsprozess nicht erkannt werden. Abbildung 4-18 zeigt eine solche Interferenz zwischen den Funktionen F2 und F3. Insbesondere in der Mechanik ist dies oft nicht vermeidbar, da z. B. die Verbindung zweier Bauteile nicht nur Kraft, sondern auch Wärmeenergie überträgt, deren Übertragung aber nicht immer gewünscht ist. Zwischen zwei Softwarediensten kann z. B. eine Interferenz durch die Nutzung desselben

Bussystems durch verschiedene Funktionen auftreten. Wenn F2 fehlerhaftes Verhalten aufweist (z. B. durch Senden einer großen Menge fehlerhafter Daten), ist dies möglicherweise nicht sicherheitsrelevant. Die gesendeten Daten können jedoch den Bus verstopfen, so dass die sicherheitsrelevante Funktionalität von F3 nicht mehr in der Lage ist, ihre Daten zu senden.

4.4.2.2 Irrtümer aufgrund unzureichender Modultests

Für den Fall, dass das zweite Ziel (Modulimplementierung stellt die spezifizierte Funktion dar) nicht erreicht wird, ist die Implementierung des Moduls fehlerhaft. Hierfür können Irrtümer durch Dekomposition bis zur konkreten Implementierung verantwortlich sein. Dies ist aber nicht Teil dieser Analyse, da aufgedeckte Irrtümer nicht zu einer Freigabe führen, vorausgesetzt die weiteren Organisationsprozesse für eine Freigabe sind nicht fehlerhaft.

Sicherheitskritisch ist jedoch, wenn eine fehlerhafte Implementierung nicht aufgedeckt wird, sodass ein Modul freigegeben wird, obwohl es die Spezifikation nicht erfüllt. Irrtümer in Modultests aufgrund der Dekomposition werden daher im Weiteren näher analysiert. Der daraus folgende zweite übergeordnete Irrtum B beschreibt, dass die spezifizierte Funktion des Moduls nicht ausreichend getestet ist. Dies ist entweder darauf zurückzuführen, dass die richtigen Tests falsch durchgeführt werden (B1) oder dass die falschen Tests durchgeführt werden (B2).

Abbildung 4-19 zeigt eine verallgemeinerte Testumgebung mit dem Module under Test (MuT), die spezifizierte Testumgebung sowie den Testfallgenerierungs- und Evaluationsprozess. Sie dient zur Unterstützung und Ergänzung der folgenden Ableitung von Irrtümern. Zusätzlich ist der Prozess zur Generierung von Testdaten und der Speicherung der Testfälle in einer Datenbank dargestellt. Ursachen für Irrtümer bei der Testfalldefinition (d.h. u. a. aufgrund fehlerhafter Informationen für die Testfallgenerierung) werden unter Irrtum B2 weiterentwickelt. Der Prüfstand liest die Testfälle ein und erzeugt anhand von Simulationsmodellen des Umfelds eines Moduls sowie anderer Module des gemeinsamen Systems Stimuli. Die Stimuli z.B. in Form generierter Daten werden an das MuT weitergegeben. Das MuT erzeugt daraus Ausgaben, die entweder direkt evaluierbar sind, falls eine Metrik hierfür existiert, oder die von Simulationsmodellen weiterverarbeitet werden, sodass die Ausgaben bewertbar werden. Falls erforderlich werden diese weiterverarbeiteten Ausgaben zurückgespeist, sodass von einer Closed Loop Simulation gesprochen wird. Wie zuvor bereits erläutert, zeigen die eckigen Kästchen die Lage der zugehörigen Irrtümer, die durch den Fehlerbaum entwickelt werden. Nach Entwicklung des Fehlerbaums, werden verbliebene Komponenten und Verbindungen ohne zugeordnete Irrtümer analysiert. In diesem Schritt sind neu aufgedeckte Irrtümer im Fehlerbaum und der Skizze des Prüfstands als blaue Kästchen dargestellt.

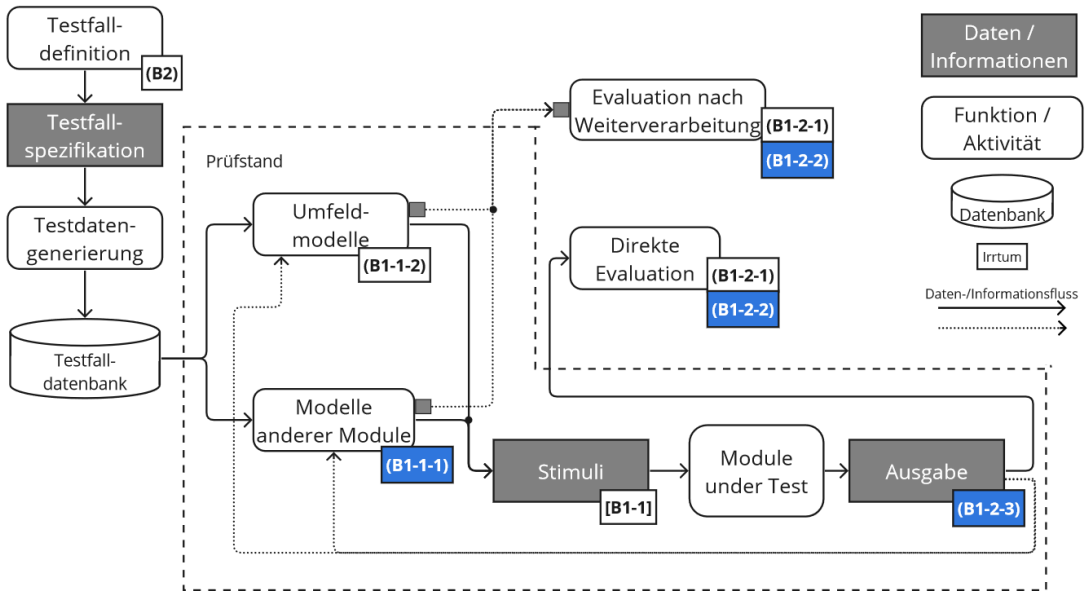


Abbildung 4-19: Verallgemeinerte Testumgebung eines Moduls, visualisiert als UML-Aktivitätsdiagramm. Zusätzlich sind die Schritte zur Testfallidentifikation und -generierung sowie der Auswertung dargestellt. Die gestrichelte Linie stellt die Grenze der Testumgebung dar.

Richtige Tests falsch durchgeführt

Falsch durchgeführte Tests lassen sich auf invalide Stimulation (B1-1) oder invalide Evaluation (B1-2) zurückführen. Dabei wird für die modulare Absicherung die Bedingung gesetzt, dass Module unabhängig von anderen Modulen testbar sind. Die Einschränkung folgt aus der Anforderung der modularen Absicherung, dass Module veränderbar sind, ohne die Absicherung anderer Module erneut durchführen zu müssen. Die Unabhängigkeit bezieht sich also auf zukünftige Änderungen anderer Module sowie deren aktuellen Entwicklungsstand bzw. die exakte Implementierung der Funktionen nach dem Konzept der Kapselung.[300] Dies ist nur möglich, wenn der Einfluss geplanter Änderungen auf das untersuchte Modul bekannt ist, was ebenfalls durch die Ziele G12 und G17 adressiert wird. Vielversprechend ist dabei der Einsatz vereinfachter Repräsentationen anderer Module, die lediglich das für das untersuchte Modul relevante Verhalten abbilden und damit, wie auch unter Ziel G8 angestrebt, einen erweiterten Gültigkeitsbereich besitzen.

Abbildung 4-19 zeigt, dass die Stimulation eines Testobjekts in der Regel in die Stimulation durch andere Module des betrachteten Systems und die Stimulation durch die übrige Umgebung unterteilt wird. Als Ursache für invalide Stimulation (B1-1) kann der Irrtum B1-1-1 auftreten, wenn die angegebene und gewünschte Stimulation durch andere Module des Gesamtsystems invalide ist. Zusätzlich kann der Irrtum B1-1-2 zu invalider Stimulation führen, wenn die übrige Umgebung invalide modelliert ist. Nicht dekomponierte Systeme benötigen in einer Gesamtsimulation nur ein Modell ihrer Umgebung. Bei Tests von Systemen in ihrer Zielumgebung werden keinerlei Modelle benötigt. Im Vergleich dazu besitzen dekomponierte Module Schnittstellen zu anderen Modulen und direkte oder indirekte Schnittstellen

[300] Broy, M.; Kuhrmann, M.: Einführung in die Softwaretechnik (2021), S. 332–335.

zur übrigen Umgebung. Die Wahl der Modulgrenzen bei der Dekomposition der Systeme beeinflusst daher wesentlich den notwendigen Aufwand zur Erreichung einer ausreichend validen Testumgebung. Die beiden Irrtümer B1-1-1 und B1-1-2 können daher bereits bei der Wahl der Modulgrenzen verursacht werden.

Der Irrtum B1-2 tritt auf, wenn nicht nur die Stimulation, sondern auch die Auswertung invalide ist. Die Auswertung von Modultests kann direkt am Ausgang der getesteten Module erfolgen. Eine weitere Möglichkeit ist die indirekte Bewertung mit Hilfe von weiterverarbeiten Modellausgaben. Dieser zweite Weg erfordert Modelle anderer Module oder der übrigen Umgebung, was wiederum zu den Irrtümern B1-1-1 und B1-1-2 führen kann. Auch hier sind die Irrtümer eine Folge der Dekomposition bzw. der Wahl der Modulgrenzen.

Beide Wege erfordern spezifische Metriken und entsprechende Bestehens-/Versagenskriterien (von Steimle et al.[301] auch als Evaluationskriterien oder von Hood et al.[302] als Verifikationskriterien bezeichnet). Dabei ist zu beachten, dass eine geeignete Schnittstelle zur Bewertung von den verfügbaren Metriken abhängt. Die Metrik prädiziert dabei prinzipiell das Verhalten auf Systemebene. Modulgrenzen können allerdings so ungeschickt definiert sein, dass eine gültige Bewertung nicht möglich ist. Zum Beispiel ist die Bewertung der Erreichung der beabsichtigten Funktionalität bzw. der Korrektheit der Ausgaben eines dekomponierten Umfeldsensors, wie von Rosenberger et al.[303] dargestellt, und anhand von Sensor-Rohdaten derzeit nicht möglich und auch zukünftig voraussichtlich sehr anspruchsvoll.

In Abbildung 4-19 wird die Ausgabe des MuTs in der Fehlerbaumanalyse initial keinem Irrtum zugeordnet. Eine daraus initiierte weitere Iterationsschleife mit Hinblick auf den Dekompositionsprozess deckt an dieser Stelle den Irrtum B1-2-3 auf. Dieser besagt, dass die Ausgabe für eine Bewertung ggf. unzureichend ist, z. B., wenn Informationen fehlen. Im Gegensatz zu einer unzureichenden Metrik (B1-2-1) gibt es in diesem Fall zwar eine Metrik, allerdings ist es technisch nicht möglich, die notwendigen Informationen für diese Metrik zu liefern. Ein beispielhaftes Testziel für diesen Fall ist die Prüfung, ob ein Planer in der Lage ist, Verkehrsregeln richtig zu erkennen. Dies lässt sich erreichen, indem eine Metrik die vom Planer erkannten Regeln mit den existierenden, gültigen Regeln abgleicht (vgl. bspw. Lippert et al.[304]). Zum Beispiel kann überprüft werden, ob der Planer ein Stoppschild korrekt erkennt und weiß, dass es das Fahrzeug auffordert, vor der Haltelinie anzuhalten. Der Verhaltens- und Trajektorienplaner des Projekts UNICAR*agil* liefert hingegen eine Trajektorie als wesentliche Ausgabe. Die Informationen über die erkannten Verkehrsregeln können daher nur indirekt über die vorgegebene Trajektorie abgeschätzt werden. Dies führt zu einer komplexeren und weniger eindeutigen Metrik.

[301] Steimle, M. et al.: Terminology for Scenario-Based Development and Test.

[302] Hood, C. et al.: Requirements Management (2008), S. 165.

[303] Rosenberger, P. et al.: Functional Decomposition of Lidar Sensor Systems (2020).

[304] Lippert, M. et al.: Behavior-Semantic Scenery Description (2024).

Falsche Tests durchgeführt

Der rechte Ast des Fehlerbaums listet lediglich falsche Tests aufgrund unzureichender Ableitung von Testfällen (B2-1) auf, da hier nur der Dekompositionsprozess, nicht aber die Testfallgenerierung auf Systemebene betrachtet wird. Nach Amersbach und Winner[305] kann die individuelle Relevanz eines Parameters für eine funktionale Ebene bspw. durch Experten oder durch Sensitivitätsanalysen ermittelt werden. Um die Relevanz durch Experten zu spezifizieren, müssen die ausführenden Experten Wissen über die Schnittstellen, Funktionalitäten und die Implementierung des Moduls haben. Kenntnisse über die Implementierung sind wichtig für Fälle, in denen ein Parameter auf Systemebene p_{S1} nur indirekt an das betrachtete Modul mit einem Parametersatz p_M übertragen wird, so dass $p_{S1} \neq p_M$ gilt. Wenn p_{S1} andere Parameter von p_M beeinflusst, gilt p_M (p_{S1}). Folglich muss p_{S1} bei Modultests berücksichtigt werden. Fehlinformationen an den Schnittstellen sind bereits durch den Irrtum A3 abgedeckt. Dennoch kann der beschriebene Zusammenhang zwischen System- und Modulebene zu Irrtum B2-1-1 führen, so dass relevante Parameter bei Dekomposition der Systemparameter nicht berücksichtigt werden. Dies kann durch einen unzureichenden Ableitungsprozess (B2-1-1-1) oder durch die bereits vorgestellte invalide Stimulation (B1-1) oder Auswertung (B1-2) verursacht werden. Darüber hinaus sind auch andere Informationen, die für die Generierung von Modultestfällen in Abschnitt 3.3.3 vorgeschrieben sind, nach Dekomposition ungewiss. Dies wird durch die Irrtümer (B2-1-2) und (B2-1-3) adressiert. Die abgeleiteten Irrtümer werden durch den Fehlerbaum in Abbildung 4-20 veranschaulicht.

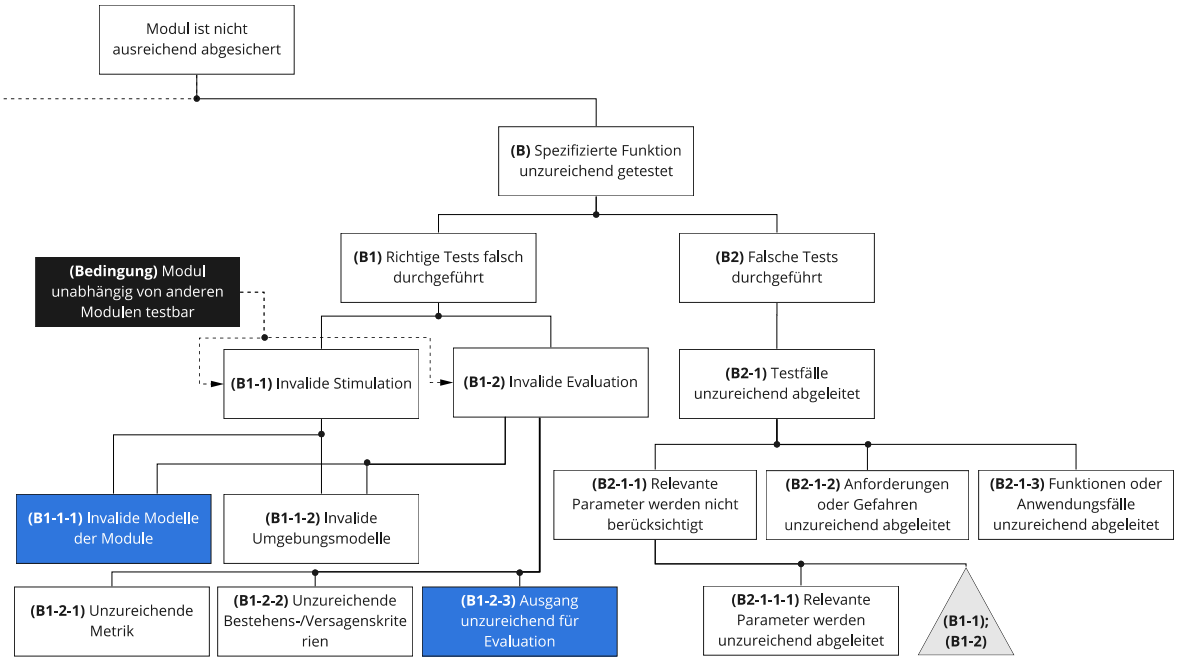


Abbildung 4-20: Baumstruktur des Irrtums B.

[305] Amersbach, C.; Winner, H.: Functional Decomposition to Reduce Approval Effort (2017).

4.4.3 Vermeidung von Irrtümern im Dekompositionsprozess

Im Folgenden werden Ziele zur Vermeidung der zuvor identifizierten Irrtümer definiert und in der GSN-Argumentationskette ergänzt. Hierzu werden die Irrtümer auf den niedrigsten Ebenen negiert und die Erkenntnisse aus deren Herleitung genutzt, um die Ziele zu konkretisieren. Die Ziele konkretisieren die Ziele G31 und G32 nach Abbildung 4-15 und unterstützen daher die Verringerung der zu Beginn von Kapitel 4.4 eingeführten Ungewissheiten bei Durchführung der Dekomposition eines Systems bzw. von Informationen dieses Systems. Mit Definition eines konkreteren Entwicklungsprozesses lassen sich die dabei identifizierten Arten von Irrtümern noch weiter konkretisieren, d.h. die beiden Baumstrukturen ergänzen. Hierdurch lassen sich äquivalent zum folgenden Vorgehen ebenso konkretere Ziele zu deren Vermeidung definieren, wie in Abbildung 4-21 dargestellt. Zur Wahrung der anfangs festgelegten Unabhängigkeit von einem konkreten Entwicklungsprozess werden im Folgenden lediglich die zuvor abgeleiteten Irrtümer betrachtet.

Das erste Ziel G33 leitet sich insbesondere aus den Irrtümern A1 und A2 ab. Die damit adressierte Notwendigkeit, dass alle Funktionen auch alle Anforderungen erfüllen, erscheint trivial. Allerdings leitet sich für den Dekompositionsprozess auch die Forderung nach Rückverfolgbarkeit von Funktionen und Anforderungen zwischen System- und Modulebene ab. Insbesondere für Systeme mit verschiedenen Entwicklungspartnern, die ggf. nur beschränkt Informationen untereinander austauschen, ist dies eine Herausforderung.[306] Eine rein statische Betrachtung der Funktionen und Anforderungen eines dynamischen Systems ist außerdem nur zulässig, wenn die Funktionen und Anforderungen für alle dynamischen Zustände gleich formulierbar sind. Komplexe Systeme erfordern meist die Formulierung verschiedener Systemzustände (z. B. für verschiedene Betriebsmodi, Informationsverfügbarkeiten oder bei Richtungsabhängigkeiten), da sich ein Modul in diesen ggf. unterschiedlich verhält. Diese Zustände und zugehörigen Zustandsübergänge sind bei der Spezifikation von Funktionen zu berücksichtigen.

Auf Basis von Irrtum A3 leitet sich das Ziel G34 ab, dass die Schnittstellen zwischen den Modulen so detailliert zu beschreiben sind, dass nicht nur die spezifizierte Modulfunktion dargestellt wird, sondern auch sichergestellt ist, dass alle Module im Verbund alle Anforderungen der Systemebene erfüllen. Obwohl die Beschreibung der Schnittstellen als wichtiger Meilenstein im Entwicklungsprozess gesehen wird, ist die Umsetzung einer technischen Schnittstellenbeschreibung in Entwicklungsprozessen oft lückenhaft.[307,308]

Das nächste Ziel G35 ist eine Erweiterung des Ziels G34. Es berücksichtigt auch indirekte Abhängigkeiten über mehrere Module hinweg. Hierdurch werden Einflüsse von Modulen identifiziert, die keine Schnittstellen zum untersuchten Modul haben. Auf Grundlage von

[306] Schäuffele, J.; Zurawka, T.: Automotive Software Engineering (2016), S. 151.

[307] Sanchez, R.: Building real modularity competence (2013), S. 221–224.

[308] Leveson, N.: Engineering a safer world (2012), S. 314.

Sensitivitätsanalysen der jeweiligen Module lässt sich die Relevanz dieser Einflüsse quantifizieren.

Das Ziel G36 präzisiert das Ziel G35 für Einflüsse, die nicht erwünscht, aber ggf. nicht vermeidbar sind, wie in Irrtum A4 beschrieben. Solche Einflüsse sind zu identifizieren und z. B. durch Sensitivitätsanalysen zu quantifizieren, um zu entscheiden, ob diese in Modultests zu berücksichtigen oder ob die Einflüsse vernachlässigbar sind.

Mit Ziel G37 wird gefordert, dass Modulschnittstellen so definiert sind, dass alle notwendigen Modultests in einer ausreichend validen Umgebung durchführbar sind. Dieses Ziel adressiert die möglichen Irrtümer B1-1-1 und B1-1-2 und erweitert die bereits in Kapitel 4.3.2 formulierten Ziele an Validität der Testumgebung, um den Einfluss der Dekomposition auf die mögliche Erreichung einer validen Testumgebung zu berücksichtigen.

Zur Vermeidung der Irrtümer B1-2-1, B1-2-2 und B1-2-3 wird ergänzend zu Ziel G37 das Ziel G38 formuliert. Dieses fordert die Wahl von Modulschnittstellen, die eine Bewertung der Testergebnisse anhand der gewählten Metriken und Bestehens-/Versagenskriterien ermöglichen. Zur Erfüllung der Ziele G37 und G38 sind die Testziele, Testfälle, Metriken und Bestehens-/Versagenskriterien daher bereits früh im Entwicklungsprozess vor der endgültigen Definition der modularen Architektur zu definieren. Dies ermöglicht die Überarbeitung der Modulschnittstellen abhängig von den zur Verfügung stehenden Test- und Bewertungsmethoden.

Unter Ziel G39 wird die Rückverfolgbarkeit aller Informationen zwischen Modul- und Systemebene gefordert, um die Irrtümer B2-1-1-1, B2-1-2 und B2-1-3 zu vermeiden. Dies erscheint als erweiterte Anwendung von Ziel G33 auf alle Informationen. G39 zielt jedoch nicht auf eine durchgehende Konsistenz zwischen den Informationen verschiedener Architektursichten, wie sie nach G33 zwischen Funktionen und Anforderungen notwendig ist. Stattdessen wird für alle Architektursichten gefordert, dass die Informationen innerhalb der jeweiligen Architektursicht zwischen den Hierarchieebenen rückverfolgbar sind. Die Konsistenz zwischen verschiedenen Architektursichten ist nur dann möglich, wenn Informationen logisch miteinander verknüpfbar sind. Betrachtet man z. B. die Architektursichten für Topologie und Software, so haben diese keine direkten Äquivalente, da die Topologie Verbindungen zwischen Hardwaremodulen beschreibt (vgl. auch Kapitel 2.2). Hierunter fallen bspw. auch rein mechanische Verbindung oder Materialflüsse. Diese besitzen keine äquivalente Softwarekomponenten.

Auf Basis der Erkenntnis, dass Informationen von der Systemebene potentiell unzureichend abgeleitet werden und eine Vielzahl an Architekturen zur Absicherung beitragen, wird das Ziel G40 ergänzt. Dieses fordert, dass alle Informationen auf der untersten abgeleiteten Ebene auch eindeutig einem Modul zugeordnet sind.

Abbildung 4-21 zeigt die Vervollständigung der GSN-Argumentationskette mit den vorgestellten Zielen auf Basis möglicher Irrtümer im Dekompositionsprozess. Alle dargestellten Ziele fließen in die neu entwickelte Lösung in Kapitel 5 ein, sodass in der Darstellung auf die hellblauen Kreise verzichtet wird.

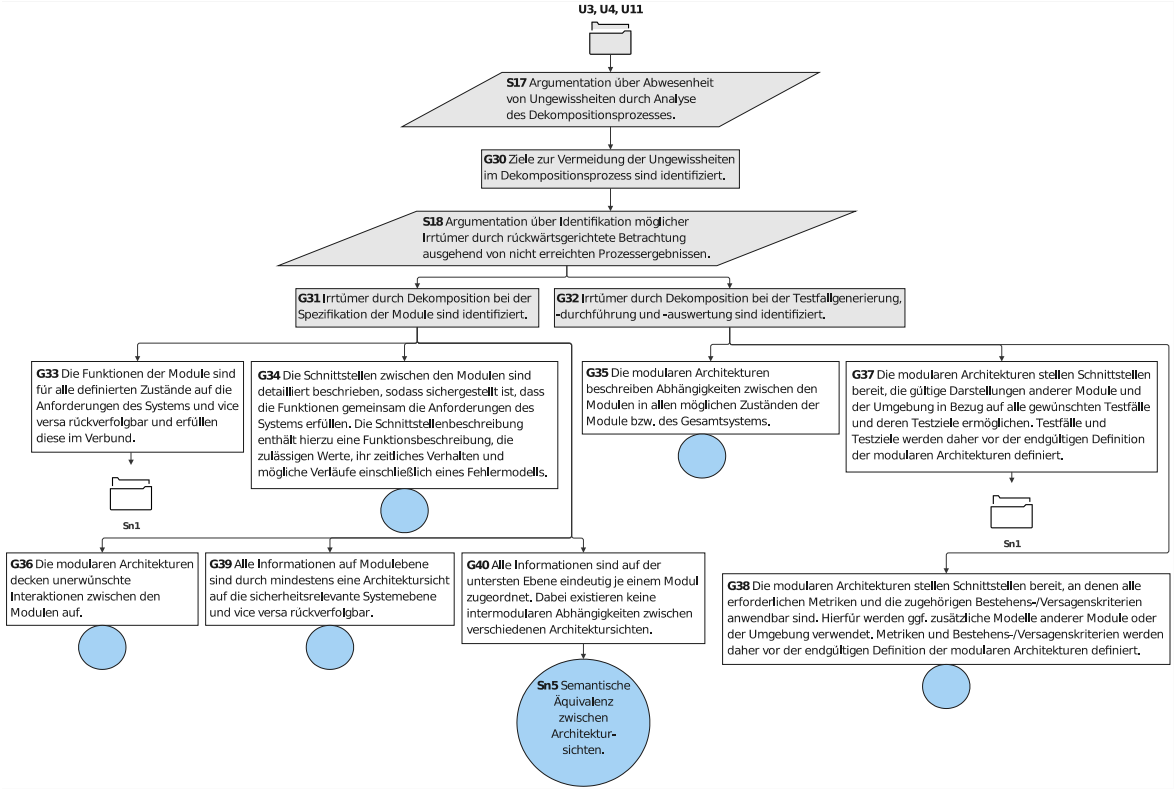


Abbildung 4-21: Dritter Pfad der GSN-Argumentationskette zur Vermeidung von Irrtümern im Dekompositionsprozess.

4.5 Zwischenfazit

Für die Systemebene sowie für die Spezifikation modularer Architekturen zeigt sich, dass für die notwendigen Ziele der Argumentationskette für eine modulare Absicherung bereits Lösungen existieren. Kapitel 4.2 beschreibt, dass Methoden aus der Systemtechnik bereits zur Argumentation einer äquivalenten Betrachtung eines Systems gegenüber einem Modul beitragen. Die Ziele zur Komplexitätsbeherrschung werden bspw. anhand von Prinzipien bei der Architekturgestaltung und einer detaillierten Modellierung der Modularchitektur adressiert. Kapitel 4.3 zeigt, dass zur Argumentation der Vollständigkeit und Validität von Modultests ebenfalls umfassende Lösungen aus dem Bereich des Testmanagements existieren, die auch in den für die Automobilindustrie geltenden Normen gefordert werden. Die Argumentationspfade der Kapitel 4.2 und 4.3 sammeln darüber hinaus Informationen und Methoden, die eine modulare Absicherung unterstützen.

Die Argumentationspfade bringen an mehreren Stellen die wesentlichen Schwachstellen einer modularen Absicherung hervor. Bisherige Prinzipien zur Architekturgestaltung sorgen demnach nicht für eine konsequent äquivalente Zuordnung von Informationen zu Modulen zu den nach dem Stand der Technik verwendeten Architektursichten. Neben der bereits durch mehrere Forschungsprojekte adressierten Notwendigkeit zur Modellierung valider Simulationsmodelle, zeigt die Argumentationskette, dass bereits die Definition und Auswahl

einer Testumgebung nicht klar strukturiert ist, wodurch die Ungewissheit über die Vollständigkeit der Modultests erhöht wird. Darüber hinaus bestätigt sich mehrfach, trotz der Argumentation anhand unterschiedlich gewählter Pfade, die zu Beginn des Kapitels getroffene Annahme, dass die Spezifikation der Module, die Modellierung der Abhängigkeiten zwischen den Modulen und die daraus folgenden Modultests die wesentlichen Ungewissheiten darstellen. Daher argumentiert Kapitel 4.4 zusätzlich die Abwesenheit von Ungewissheiten im Dekompositionsprozess sowohl zur Spezifikation als auch zur Testfallgenerierung, -durchführung und -auswertung. Hieraus werden weitere Informationen und Gestaltungsregeln ermittelt, die Irrtümer im Dekompositionsprozess vermeiden. Zur Erreichung der daraus abgeleiteten Ziele der GSN Argumentationskette werden im Folgenden neue Lösungen vorgestellt.

4.6 Neue Lösungsansätze für eine modulare Absicherung

Der folgende Abschnitt stellt auf Basis der Ziele, die nach bisherigen Methoden des Stands der Wissenschaft und Technik nicht lösbar sind, neue Ansätze vor, die diese Ziele adressieren. Die Ziele sind in Anhang A vollständig aufgelistet. Die Zuordnung der neuen Lösungsansätze zu den Zielen erfolgt innerhalb der folgenden Kapitel sowie anhand des Anhangs B.1, der die Attribute der detaillierten semantischen Schnittstellenbeschreibung nach Kapitel 5 den Zielen zuordnet. Zuerst wird ein neues Prinzip zur Architekturgestaltung vorgestellt, das unterschiedliche Modulgrenzen in Architektursichten vermeidet, die für die Absicherung relevant sind. Das Prinzip beschreibt eine semantische Äquivalenz zwischen Architektursichten. Zur Argumentation, dass Testumgebungen für Module Validitätseigenschaften erfüllen, wird ein neuer Ansatz vorgestellt, um die Testumgebungen Schritt für Schritt argumentativ zu reduzieren. Hierdurch wird die Ungewissheit über die Vollständigkeit der Modultests reduziert, da die Testumgebungen der Module systematisch ausgewählt und spezifiziert werden. Im dritten Abschnitt wird schließlich das Ziel einer Reduktion der Ungewissheit durch die Dekomposition von Informationen der Systemebene adressiert. Hierzu beschreibt der Ansatz einer evolutionären Entwicklung von Modulen hochautomatisierter Fahrzeuge und die dafür notwendigen Anforderungen eine potentielle Lösung. Insbesondere wird dabei die Notwendigkeit der Standardisierung von Schnittstellenbeschreibungen hervorgehoben.

4.6.1 Semantische Äquivalenz zwischen Architektursichten

Die Ziele G33, G36, G39 und insbesondere G40 erfordern zusammengefasst, dass Informationen eindeutig einem Modul zugeordnet sind und keine intermodularen Abhängigkeiten

zwischen Architektursichten bestehen. Maurer[309] weist ebenso darauf hin, dass eine Eigenschaft komplexer Systeme beinhaltet, dass keine Eins-zu-Eins Abbildungen zwischen Architektursichten bestehe. Module werden in der Regel innerhalb einer Architektur aufgebaut und von bestehenden oder leicht zu erstellenden Schnittstellen abgeleitet. Fairbanks[310] beschreibt jedoch Inkonsistenzen zwischen den Architekturen als ein zentrales Problem, insbesondere für die Rückverfolgbarkeit von Informationen, wenn sich Teile der Architekturen ändern. Inkonsistenz führt zu fehlenden notwendigen Anpassungen bei anderen Architektursichten, die wiederum zu einem Versagen führen können. Zur Erreichung von Konsistenz zwischen funktionaler Architektur und Softwarearchitektur beschreiben Dajsuren et al.[311] den Begriff der semantischen Konsistenz. Semantische Konsistenz bedeutet, dass alle Funktionen von der Software erfüllt werden und Änderungen in einer Architektursicht für andere nachvollziehbar sind. Driessen[312] verwendet den Begriff der semantischen Äquivalenz für Softwarekomponenten als Anforderung für geänderte Komponenten, sodass alle anderen Softwarekomponenten für diese Änderung erneut verifiziert werden müssen. Dies stellt jedoch nur sicher, dass Komponenten nach Änderungen semantisch äquivalent zum vorherigen Stand sind. Beide Ansätze vermeiden jedoch nicht, dass verschiedene Architektursichten unterschiedliche Modulgrenzen aufweisen. Den Architektursichten der Hardware- und Softwarearchitektur wird mit Maßnahmen wie bspw. dem Schutz des Speichers auf der Hardware (eng.: Memory Protection) begegnet.[313] Weniger Beachtung findet dagegen bspw. die eindeutige Zuordnung zwischen den Elementen von Funktions-, Fähigkeits- und Hardwarearchitektur.

Im Folgenden wird als zusätzliche Teillösung Sn5 die semantische Äquivalenz zwischen verschiedenen Architektursichten vorgestellt. Semantische Äquivalenz zwischen Architektursichten bedeutet, dass ein Modul einer Architektursicht das semantisch äquivalente Modul einer anderen Architektursicht darstellt. Hieraus lässt sich schlussfolgern, dass ihre Modulgrenzen bzw. Schnittstellen sowie die Inhalte eines Moduls äquivalent sind und somit eindeutig zugeordnet werden können. Zur Erreichung semantischer Äquivalenz ist es daher notwendig, dass keine Querverknüpfungen (im Folgenden auch als diagonale Beziehungen bezeichnet) zwischen unterschiedlichen Modulen verschiedener Architektursichten bestehen. Alle Architektursichten, die in Zusammenhang mit der Absicherung genutzt werden, sollten diese Eigenschaft aufweisen. Neben den essentiellen funktionalen, Software- und Hardwaresichtweisen zählt hierzu insbesondere auch die Sicherheitsargumentation z. B. in Form einer GSN, wie bereits von Despotou und Kelly[314] gefordert.

---

[309] Maurer, M.: Complexity management in engineering design (2017).

[310] Fairbanks, G.: Just enough software architecture (2010), S. 304.

[311] Dajsuren, Y. et al.: Formalizing correspondence rules for automotive architecture views (2014).

[312] Driessen, T.: Dissertation, Modularity by Design (2019), S. 10.

[313] ISO: ISO 26262 - Road vehicles – Functional safety (2018), Teil 6, S. 14.

[314] Despotou, G.; Kelly, T.: Argument modularity for safety assurance (2008).

Als Beispiel für die Adaption der Elemente einer Architektursicht zur Erreichung semantischer Äquivalenz zwischen Architektursichten wird der von Reschka et al.[315] eingeführte Fähigkeitengraph betrachtet. Im Fähigkeitengraph werden Aktivitäten ggf. mit ihren Qualitäten aufgetragen und miteinander verknüpft, um jeweils eine übergeordnete Fähigkeit zu erfüllen. Die Fähigkeiten können entsprechend im Zusammenspiel Aufgaben und Anforderungen des hochautomatisierten Fahrzeugs erfüllen (vgl. die Beschreibung verschiedener Architektursichten in Kapitel 2.2). Zusätzlich wird die Hardware- und Softwaresicht im Sinne einer Allokationssicht dargestellt, um die Zuordnung der Komponenten zu verdeutlichen.

Eine Fähigkeit, die z. B. mit zwei Modulen verbunden ist, lässt sich nicht unabhängig von dem jeweils anderen Modul absichern. Außerdem kann bei einer Änderung eines Moduls die Gültigkeit der Absicherung der Fähigkeit nur wiederhergestellt werden, indem alle Module, die direkt mit dieser Fähigkeit verbunden sind, erneut geprüft werden. Abbildung 4-22 zeigt ein einfaches Beispiel, bei dem die übergeordnete Fähigkeit "Längsdynamik kontrollieren" (wie von Bagschik et al.[316] beschrieben) zunächst mit dem Trajektorienregler, dem Antriebsmotor und der hydraulischen Bremse verbunden ist (linker Teil der Abbildung). Die Fähigkeit wird in Beschleunigung und Verzögerung zerlegt, um eine detaillierte Beschreibung der Fähigkeiten zu erhalten. Es ist in dem Fall jedoch nicht möglich, die Steuerung von den Aktoren zu trennen, sodass der Regler als unabhängiges Modul ohne die Aktoren betrachtet und abgesichert werden kann, das als Ziel m Projekt UNICAR*agil* verfolgt wird.[317] Daher werden die Fähigkeiten stattdessen weiter in die Fähigkeiten "Stellgrößen ermitteln", "positives Radmoment erzeugen" und "negatives Radmoment erzeugen" zerlegt.

In UNICAR*agil* werden der Elektromotor und die hydraulische Bremse innerhalb eines Moduls („Dynamikmodul“) betrachtet, sodass diese gemeinsam den Fähigkeiten "Radmoment in Drehrichtung erzeugen" und "Radmoment entgegen Drehrichtung erzeugen" zugeordnet werden. Dies ist für die Absicherung von Vorteil, da z. B. die sonst notwendige Beschreibung des sogenannten „Blending“ (Kombination aus Drehmoment des Elektromotors und der Hydraulikbremse) komplexer ist. Die Fähigkeit "Stellgrößen ermitteln" kann durch die Zerlegung eindeutig dem Trajektorienregler zugeordnet und dahingehend abgesichert werden.

Die Bewertung der Stellgrößen ist dagegen herausfordernd, da diese z. B. gegenüber einer Regelabweichung oder einer Geschwindigkeit abstrakte Ausgaben sind. Deren Transformation in ein Verhalten auf Systemebene ist stark abhängig von den verwendeten Aktoren und somit für einen Menschen selbst bei Kenntnis der Aktoren schwierig zu interpretieren. Die modulare Absicherung des Trajektorienreglers erfordert daher zusätzlich Modelle der Aktorik oder eine Metrik, die Stellgrößen direkt bewertet.

---

[315] Reschka, A. et al.: Ability and skill graphs for system modeling (2015).

[316] Bagschik, G. et al.: Architecture Framework for Safe Automated Vehicles (2018).

[317] Homolla, T.; Winner, H.: Encapsulated trajectory tracking control for autonomous vehicles (2022).

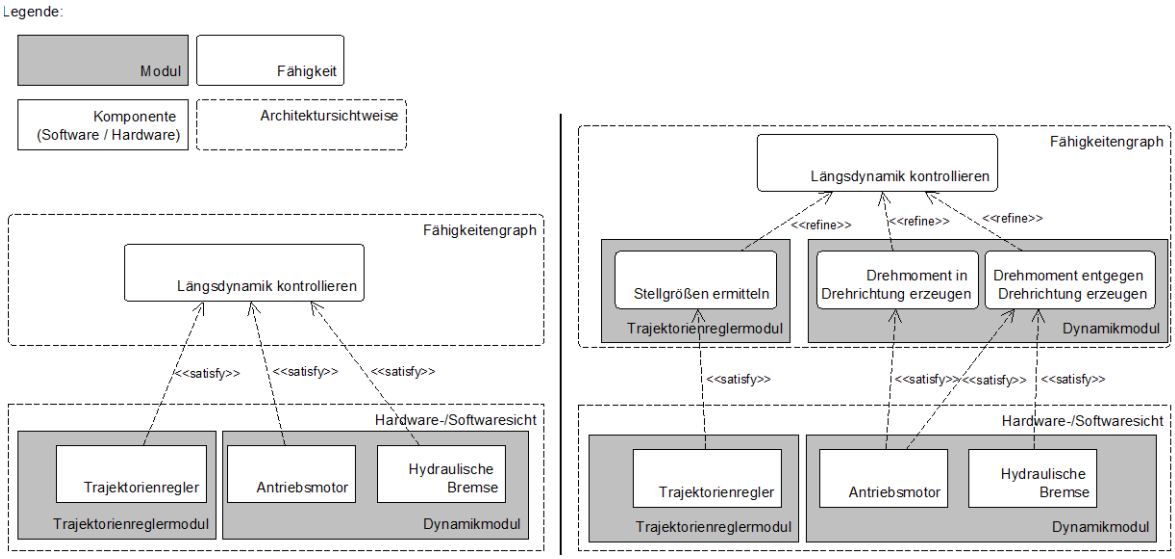


Abbildung 4-22: Erzeugung semantischer Äquivalenzen durch Zerlegung der Fähigkeit und Aufbau äquivalenter Module im Fähigkeitengraph. Vereinfachtes Beispiel aus UNICAR*agil*, dargestellt durch ein UML-Aktivitätsdiagramm.

Im Forschungsprojekt UNICAR*agil* teilt sich in der Architektur der Prototypen entgegen der vereinfachten Darstellung nach Abbildung 4-22 der Trajektorienregler ein Steuergerät mit anderen Diensten. Für eine Hardware-Architektur scheint semantische Äquivalenz gegenüber der Softwarearchitektur nicht erreichbar, da in der Regel mehr als eine Softwarekomponente mit einer Hardwarekomponente (z. B. einem Steuergerät) verbunden ist. Doch auch wenn eine Hardwarekomponente physikalisch nicht trennbar ist, kann sie logisch getrennt werden, indem bspw. die Zuordnung von Speicher oder Rechenleistung zu einer Softwarekomponente festgelegt wird. ISO 26262[318] schlägt hierzu bspw. die Nutzung von Speicherschutzeinheiten (eng.: memory protection units) vor. Weiterhin kann die Verwaltung von Ressourcen verbessert und die Abhängigkeit zwischen Softwarekomponenten bspw. aufgrund gemeinsam genutzter Betriebssysteme durch Containerisierung reduziert werden.[319] Falls diese logische Trennung jedoch nicht durch eine ausreichend valide Beschreibung und der Abhängigkeit zu anderen logischen Komponenten möglich ist, ist eine semantisch äquivalente Modularchitektur nicht erreichbar. Burton[320] schlägt daher als weiteren Schritt die Partitionierung von Rechenressourcen vor, um eine konsequente Trennung zu erreichen. Hierbei werden Partitionen auf dem realen Steuergerät erstellt, die jeweils ein virtuelles Steuergerät darstellen. Gegenüber einer Memory[321a,] oder Timing Protection[321b], die bereits

[318] ISO: ISO 26262 - Road vehicles – Functional safety (2018), Teil 6, S. 14.

[319] Tamanaka, G. T. et al.: Fault-tolerant architecture using containers (2022).

[320] Burton, S.: Entwicklungsprozess für funktional sichere Steuergeräte (2015).

[321] AUTOSAR: Overview of Functional Safety Measures in AUTOSAR (2019), a: S.8-22, b: S. 23-30.

für AUTOSAR[322] gefordert und spezifiziert sind, bietet die Partitionierung eine Kombination dieser Schutzmaßnahmen inklusive des Verbergens bspw. von Softwarecodes, der nicht mit anderen ausgetauscht werden soll.

4.6.2 Methode zur argumentativen Testumgebungsreduktion

Der Stand der Technik befasst sich bei der Validierung meist mit einem bestimmten Modul und einer Konfiguration der Testumgebung. Notwendige Schritte zur Identifikation notwendiger Validitätseigenschaften, wie bereits in Kapitel 4.3.2 durch die verbleibende Ungewissheit U10 dargestellt, sind dabei teils unklar. Hieraus leiten sich zwei Herausforderungen ab. Erstens kann die Erreichung von Validität selbst je nach dem zu prüfenden Modul und der für seine Absicherung erforderlichen Umgebung herausfordernd sein (vgl. z. B. Holder et al.[323]). Zweitens ist die Auswahl der richtigen Konfiguration der Testumgebung für eine modulare Absicherung aufgrund der hohen Anzahl möglicher Konfigurationen herausfordernd. Mit einer Anzahl an Modulen *m* und ihrer möglichen Anzahl an Repräsentation *R* (unter der Annahme, dass alle Module die gleiche Anzahl an unterschiedlichen Darstellungen haben), insgesamt R^m mögliche unterschiedliche Testumgebungen. Hakuli und Krug[324] betrachten bspw. neun Module, die jeweils in realer oder virtueller Form in einer Testumgebung repräsentiert werden. Sie nennen jedoch nur sieben verschiedene Kombinationen, die jeweils eine Testumgebung darstellen (vgl. auch Kapitel 2.2). Damit schließen sie 505[325] mögliche Testumgebungskonfigurationen ohne weitere Erläuterungen aus. Daher wird in diesem Abschnitt mit der Entwicklung einer neuen Methode das Ziel verfolgt, die Validität einer vereinfachten bzw. reduzierten Testumgebung gegenüber Testumgebungen ohne diese Vereinfachungen zu argumentieren.

Als Ausgangspunkt wird der Systemtest in realer Umgebung betrachtet. Die Ableitung und Argumentation einer validen Testumgebung für Modultests beginnen daher auf Systemebene mit allen realen Modulen in realer Systemumgebung. Während das Testziel für diese Ebene letztendlich von der Teststrategie abhängt (z. B. kann sie nur zur Validierung anderer Testumgebungen verwendet werden), wird im Folgenden davon ausgegangen, dass das Testziel der Systemebene die Absicherung des Systems ist. Davon ausgehend wird das Module under Test (MuT) gewählt und andere *Module oder Elemente der Umgebung sukzessive reduziert*. Diese Module oder Umgebungselemente können vollständig gestrichen oder durch Repräsentationen ersetzt werden.

Jeder Reduktionsschritt der Testumgebung beeinflusst potentiell die Stimuli auf das MuT sowie das beobachtbare und ggf. bewertete Verhalten aufgrund des Verhaltens des MuTs.

---

[322] AUTOSAR: Requirements on Operating System (2022), S. 17–20.

[323] Holder, M. et al.: Challenges in Radar Sensor Modeling (2018).

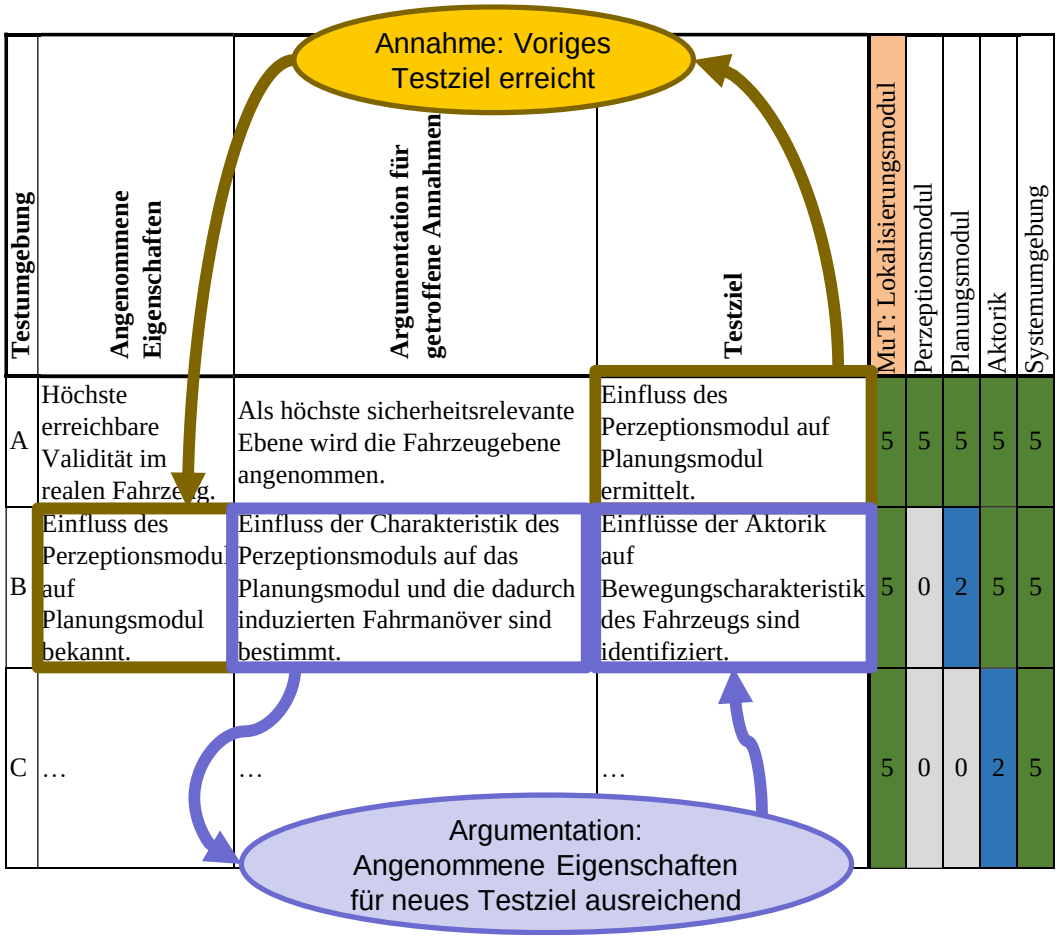
[324] Hakuli, S.; Krug, M.: Virtual Integration in the Development Process (2016), S. 160–161.

[325] Mit $R = 2$ und $m = 9$ ergeben sich mit $2^9 - 7 = 505$ nicht betrachtete Kombination.

Daher wird für jeden Reduktionsschritt eine Analyse und die daraus hervorgehende Begründung benötigt, welche argumentiert, dass das jeweilige Testziel der Testumgebung sowie das übergeordnete Testziel, sicheres Verhalten des MuTs nachzuweisen, nicht beeinflusst wird. Gleiches gilt, wenn das *MuT für die Integration in der Testumgebung* angepasst oder vereinfacht wird. Zum Beispiel kann das MuT Anpassungen erfordern, um Schnittstellen bereitzustellen, die eine Manipulation der Eigenschaften oder Zustände des Moduls ermöglicht, um z. B. Degradationen zu simulieren. Jede Testumgebung besitzt ein *Testziel*, kann allerdings auch mehreren zugeordnet sein. Das übergeordnete Testziel kann auch für die neu definierten Testumgebungen übernommen werden, zu denen diese Testumgebungen beitragen. Testziele können allerdings auch hinzugefügt oder in mindestens zwei Testziele für mindestens zwei verschiedene Testumgebungen zerlegt werden. Dies kann auf eine erforderliche Validierung einer Testumgebung zurückzuführen sein oder darauf, dass eine Testumgebung nur für einen begrenzten Teil der Testfälle geeignet ist. Jede Änderung eines Testziels sollte hinsichtlich ihres Einflusses auf die übergeordneten Testziele analysiert werden. Das Ergebnis der Analyse kann als Argument für die Validität der Testumgebung beitragen. Dementsprechend bringen *Argumente zur Reduktion der Testumgebung neue Testziele* für eine Testumgebung hervor. Die Rückverfolgbarkeit der Testzielableitung z. B. durch Dokumentation der Zerlegungsschritte, unterstützt bei Änderungen an Modulen oder Testumgebungen die Identifikation potentiell notwendiger Anpassungen der Testziele.

Zur übersichtlichen Darstellung der Argumentation wird in der neu entwickelten Methode der vorliegenden Arbeit anstatt einer Argumentationskette eine Argumentationsmatrix verwendet. Das Vorgehen ist beispielhaft anhand von Tabelle 4-1 als Argumentationsmatrix zur Testumgebungsreduktion unter verbleibender Validität dargestellt. Die erste Spalte ordnet den zeilenweise aufgelisteten Testumgebungen eine ID zu. Die zweite Spalte beschreibt in der Tabelle alle Eigenschaften, die der jeweiligen Testumgebung zugeschrieben werden. In diesem Fall sind dies insbesondere die Validitätseigenschaften. In der dritten Spalte werden die Argumente beschrieben, welche die Annahme dieser Validitätseigenschaften unterstützen. Dadurch wird gleichzeitig argumentiert, dass das Testziel, das in der vierten Spalte beschrieben wird, mit den Eigenschaften der entsprechende Testumgebung erreichbar ist. Wenn dieses Testziel durch Tests erreicht wird, kann dieses widerum verwendet werden, um die Argumentation der angenommenen Eigenschaften einer anderen Testumgebung zu unterstützen. Hiermit wird entsprechend die Validierung dieser Testumgebung unterstützt. Die weiteren Spalten geben an, welches Modul in welcher Form in der Testumgebung repräsentiert wird. Je nach Anwendungsfall können dabei unterschiedlich viele Repräsentationsarten unterschieden werden. Hakuli und Krug[324] unterscheiden bspw. nur zwei verschiedene Repräsentationsarten. Im Beispiel von Tabelle 4-1 werden sechs verschiedene Arten unterschieden (Nummerierungen 0 bis 5). Die aus einem Anwendungsbeispiel stammenden Repräsentationsarten werden in Kapitel 6.2.2 detaillierter beschrieben.

Tabelle 4-1: Beispielhaftes Vorgehen zur argumentativen Reduktion der Testumgebung.



Testumgebung	Angenommene Eigenschaften	Argumentation für getroffene Annahmen	Testziel	MuT: Lokalisierungsmodul	Perzeptionsmodul	Planungsmodul	Aktorik	Systemumgebung
A	Höchste erreichbare Validität im realen Fahrzeug.	Als höchste sicherheitsrelevante Ebene wird die Fahrzeugebene angenommen.	Einfluss des Perzeptionsmodul auf Planungsmodul ermittelt.	5	5	5	5	5
B	Einfluss des Perzeptionsmodul auf Planungsmodul bekannt.	Einfluss der Charakteristik des Perzeptionsmoduls auf das Planungsmodul und die dadurch induzierten Fahrmanöver sind bestimmt.	Einflüsse der Aktorik auf Bewegungscharakteristik des Fahrzeugs sind identifiziert.	5	0	2	5	5
C	…	…	…	5	0	0	2	5

Eine Argumentation ist individuell abhängig vom Module under Test, seiner Umgebung oder dem Anwendungsfall des Gesamtsystems. Für eine vollständige Argumentation der Testumgebungsreduktion, sind im gegebenen Beispiel weitere Argumente notwendig, die anhand durchgeführter Analysen oder Tests zu bestätigen sind. Die Validität einer Testumgebung hängt nach Abbildung 4-10 letztlich von den Eigenschaften der Testumgebung und der Sensitivität des MuTs ab. Daher beschränken sich mögliche Argumente für die Reduktion der Testumgebung auf zwei mögliche Betrachtungen. Erstens kann argumentiert werden, dass die geänderte Modulumgebung keine Änderungen der auf das Modul wirkenden Stimuli und keine Änderungen der Weiterverarbeitung und Bewertung des Verhaltens des MuTs bewirken. Zweitens kann gezeigt werden, dass das MuT gegenüber Änderungen der Stimuli nicht sensitiv reagiert. Das bedeutet, das Modulverhalten ändert sich nicht oder nur in einem Umfang, dass die Bewertung des Modulverhaltens nicht beeinflusst wird. Hierbei muss auch die Weiterverarbeitung des Modulverhaltens in der Testumgebung betrachtet werden, die von den Änderungen ggf. mit betroffen ist. Zur Argumentation dieser beiden Schritte können diese mit Hilfe weiterer Testumgebungen unterstützt werden, sodass sich neue Testziele für diese Argumentation ergeben. Dabei ist auch die Zerlegung eines Testziels denkbar, sodass für verschiedene Bestandteile eines Testziels verschiedene Testumgebungen verwendet werden. Die beschriebenen Möglichkeiten zur Argumentation einer Reduktion der Testumgebung sind in folgenden Stichpunkten zusammengefasst und mit Beschreibung jeweils eines Beispiels veranschaulicht:

- Das zu testende Modul bietet eine ausreichende Robustheit gegenüber Parameteränderungen. Infolgedessen können die Repräsentanten anderer Module oder der Umgebung ihren Repräsentationsgrad verringern. Dieses Argument muss durch eine Sensitivitätsanalyse des Moduls und eine angepasste Validierung der Repräsentationen mit realen Modulen oder Umgebungen auf Basis der ermittelten Sensitivitäten gestützt werden.
    - Ein Algorithmus, der beispielsweise ein Umgebungsmodell bei Verarbeitung der Daten unterschiedlicher Sensoren liefert (z. B. durch Training eines neuronalen Netzes auf verschiedenen Sensoren) ist diesen Sensordaten ggü. robust, wenn die Qualität der Umgebungsmodelle weiterhin ausreichend für den spezifizierten Anwendungsfall ist (und damit weiterhin die Anforderungen erfüllt). Die Simulationsumgebung müsste in diesem Fall nur Repräsentanten für die Sensoren darstellen, die in der Bandbreite der verschiedenen Sensoren liegen, für die Robustheit nachgewiesen ist. Diese erhöhte Bandbreite erfordert ggf. geringeren Modellierungsaufwand für den Repräsentanten der Sensoren.
- Sensitivitätsanalysen zeigen, dass spezifische Parameter keinen signifikanten Einfluss auf das zu testende Modul haben. Der Repräsentant muss daher diese spezifischen Parameter nicht oder nur eingeschränkt repräsentieren. Das Argument ist ein Sonderfall für das vorherige Robustheitsargument, sodass eine spezifische Robustheit ggü. einzelner Parameter nachzuweisen ist.
    - Ein Verhaltensplaner kann beispielsweise die Klassen des Fahrzeugtyps aus der Perzeption zu einer einzelnen Klasse vereinfachen. Das Umgebungsmodell benötigt daher nur eine Ausgabe der Klasse Fahrzeug.
- Das Testziel wird in Testziele von mindestens zwei verschiedenen Testumgebungen zerlegt, die gemeinsam das übergeordnete Testziel erfüllen. Weitere Argumente bzw. ein Nachweis der korrekten Dekomposition sind erforderlich.
    - Ein Sensormodul kann bspw. in der Simulation getestet werden, um die korrekte Bestimmung der Zustände anderer Objekte (z. B. Geschwindigkeit oder Position) zu bestätigen. Zusätzlich wird es in einer realen statischen Umgebung getestet, um die Klasse der Objekte zu identifizieren. In diesem Fall sind zusätzliche Testziele vorzusehen, um nachzuweisen, dass die Zerlegung in die beiden Fähigkeiten (Erkennung des Objektzustands und Erkennung der Objektklasse) gültig ist.

Diese drei Argumentationstypen können für spezifische Anwendungen erweitert bzw. detailliert werden. Der Ansatz der argumentierten Testumgebungsreduktion bietet das Potential einer strukturierten Ermittlung von Testumgebungen zur Absicherung von Modulen. Jeder Reduktionschritt der Darstellung der Modulumgebung muss dem Ansatz folgend argumentiert werden. Die Argumente verringern die Ungewissheit darüber, ob die Modultests in der Testumgebung das übergeordnete Testziel auf Systemebene zum Nachweis der Sicherheit unterstützen und decken dabei ggf. die Notwendigkeit weiterer Testziele auf.

4.6.3 Evolutionäre Entwicklung und Standardisierung

Emergente Eigenschaften eines Systems sind per Definition vor dem Zusammenspiel der realen Module nicht vollständig sichtbar. Die Vorhersehbarkeit emergenter Eigenschaften auf Basis der Moduleigenschaften ist damit initial unvermeidbar mit Ungewissheiten verbunden. Eine erhebliche Verringerung dieser Ungewissheiten ist lediglich durch eine vorhergehende Beobachtung des Zusammenspiels der Module in einem Vorgängermodell möglich. Entsprechend ist eine Argumentation erforderlich, dass das bereits beobachtete Zusammenspiel auch auf neue oder geänderte Module übertragbar ist. Zur Wahrung der Gültigkeit der emergenten Eigenschaften ist eine evolutionäre Entwicklung des modularen Systems vorteilhaft. In dieser werden Teile des Systems so weit übernommen, dass bestimmte Eigenschaften sich nicht ändern. So kann bspw. die Architektur vollständig übertragen werden, während sich in einem neuen Modul innerhalb der Architektur nur die Implementierung ändert. Für eine modulare Absicherung wäre die Übertragbarkeit aller Eigenschaften des Systems notwendig, die das Verhalten der einzelnen Module beeinflussen.

In einer konsequent modularen Architektur wird das Verhalten der Module über die Modulschnittstellen beeinflusst und wirkt ebenso über die Modulschnittstellen auf andere Module sowie die Systemumgebung ein. Zur Erreichung einer möglichst hohen Unabhängigkeit von der exakten Implementierung eines Moduls ist die Modulschnittstelle als Blick von außen eine naheliegende Perspektive. Für den Fall, dass sich eine solche Beschreibung mit allen hinreichenden Informationen für eine modulare Absicherung aufstellen lässt, wäre der evolutionäre Entwicklungsansatz durch ein Einfrieren dieser Schnittstellenbeschreibung möglich. Für die Austauschbarkeit über mehrere Entwicklungsgenerationen oder verschiedene Hersteller ist eine Standardisierung sowohl des Formats als auch des Inhaltes dieser Beschreibung denkbar.

Modulare Architekturen nach dem Stand der Wissenschaft und Technik (vgl. Kapitel 3) konzentrieren sich auf die Kompatibilität der Schnittstellen, sodass die Module miteinander kommunizieren und ihre grundlegende Funktion darstellen können. Zur Erfüllung der Ziele aus Kapitel 4 zur Argumentation der Möglichkeit einer modularen Absicherung werden jedoch deutlich detaillierte Informationen über mögliche Stimulationen und daraus folgendem Verhalten benötigt. Der Standard ISO 26262[326] fordert mit der Item-Definition zwar die Erstellung einer umfangreichen Übersicht zur Umgebung und möglichem Einsatz(-verhalten), bezieht jedoch wesentliche Details, die im Rahmen des in der Norm beschriebenen Prozesses erarbeitet und für die Absicherung genutzt werden, nicht mit ein. Darüber hinaus wird weder eine konkrete Struktur für die Item Definition vorgegeben noch ein Prozess zur Erstellung der Item-Definition beschrieben. Reschka[327] beschreibt einen möglichen Prozess zur Erstellung der Item-Definition für ein hochautomatisiertes Fahrzeug. Dieser ist jedoch mit einer

[326] ISO: ISO 26262 - Road vehicles – Functional safety (2018), Teil 3, S. 4-5.

[327] Reschka, A.: Dissertation, Fertigkeiten- und Fähigkeitengraphen von automatisierten Fahrzeugen (2017), S. 101–105.

Black-Box Betrachtung gleichzusetzen, da nur die Sicht auf die Fahrzeugumgebung und das Fahrzeugverhalten beschrieben wird. Zur Beschreibung eines abstrakteren, modulspezifischen Verhaltens und deren Auswirkung auf andere Module sowie dem resultierenden Fahrzeugverhalten ist der Prozess von Reschka daher nur beschränkt anwendbar. Es wird daher eine neue Beschreibungsform für die Schnittstellen von Modulen benötigt, um Module individuell abzusichern. Zur Erreichung einer evolutionären Entwicklung ist außerdem die Standardisierung solcher Schnittstellenbeschreibungen vorteilhaft. Das folgende Kapitel leitet daher eine neue Schnittstellenbeschreibung für eine modulare Absicherung her.

5 Detaillierte semantische Schnittstellenbeschreibung[328]

Kapitel 4 zeigt, dass bisherige Formen zur Spezifikation und zum Test von Modulen für eine modulare Absicherung nicht ausreichen. Die Übertragung des Vorgehens und der Perspektiven von der Systemebene auf die Module kann die bestehenden Methoden ergänzen, scheitert aber immer wieder an möglichen Irrtümern sowie bei deren Ableitung im Dekompositionsprozess. Das Prinzip der geringsten Überraschung nach Bass et al.[329] wird mit bisherigen Methoden nicht erreicht. Zur Vermeidung oder Aufdeckung solcher Irrtümer wird die evolutionäre Entwicklung zusammen mit der Standardisierung von Modulschnittstellen als vielversprechende Lösung identifiziert. Zur Sammlung der zur Absicherung relevanten Informationen in einer Modulschnittstelle bedarf es einer neuen Beschreibungsform, die auf Module hochautomatisierter Fahrzeuge anwendbar ist.

Im Folgenden wird daher eine neue Schnittstellenbeschreibung entwickelt, die neben der Syntax insbesondere die Semantik der über die Schnittstelle transportierten Informationen abbildet. Während die Beschreibung als Vermeidungsmaßnahme für potentielle Irrtümer in der Spezifikation und Implementierung direkt im jeweiligen Prozess abgeglichen werden kann, dient sie zusätzlich zur Definition von Testfällen inklusive der Spezifikation zugeordneter Testumgebungen, sodass potentiell verbliebene Fehlerzustände aufgedeckt werden.

5.1 Ziele der Beschreibung

Der Stand der Technik bietet verschiedene Werkzeuge zur Dokumentation und Kommunikation von Informationen über Module und die zugehörige Systemarchitektur. Anforderungslisten oder eine Item Definition konzentrieren sich auf das einzelne beschriebene Modul. Werkzeuge zur Modellierung der übergreifenden Systemarchitektur existieren auf unterschiedliche Art und Weise, z. B. in Form verschiedener Typen von UML (Unified Modelling Language)-Diagrammen. Die weite Verbreitung von Informationen über andere Module, die Umgebung und deren Wechselwirkungen untereinander ist für eine Absicherung einzelner Module jedoch hinderlich. Für Modulentwickler und -tester ist ein Großteil der Informationen über das restliche System nicht relevant, sodass eine Einarbeitung in die ungefilterten Dokumentationen nicht weiter verwertbaren Arbeitsaufwand verursacht. Im schlechtesten Fall wird dadurch sogar die Bearbeitung relevanter Informationen behindert.

[328] Teile dieses Kapitels wurden bereits in Klamann, B.; Winner, H.: Introducing the detailed semantic interface description (2023) veröffentlicht.

[329] Bass, L. et al.: Software Architecture in Practice, 4th Edition (2021), S. 278.

Für Methoden der Risikoanalyse oder Testfallgenerierung kann ein genaueres Bild vombetrachteten Objekt auch eine Voreingenommenheit gegenüber den Erwartungen bspw. des Sollverhaltens oder potentieller Fehlerursachen verursachen. Es ist für die wissensbasierte Absicherung also keineswegs immer zielführend alle existierenden Informationen über ein Objekt bereitzustellen.

Darüber hinaus erfordern Dokumente oder Modelle bei Änderung eines Moduls eine Überprüfung. Eine komprimiertere und konsistentere Beschreibung der Informationen, die für die Nutzer einer Schnittstelle tatsächlich relevant sind, kann den hierfür notwendigen Aufwand für Einflussanalysen und den daraus identifizierten zu wiederholenden Absicherungsschritten reduzieren. Schnittstellen stellen passend hierfür einen abgestimmten Vertrag nur über die für beide Seiten relevanten Informationen und deren Qualität dar.[330] Äquivalent zur Beschreibungsform einer Operational Design Domain (ODD) strebt die semantische Schnittstellenbeschreibung allerdings eine möglichst einheitliche und auf verschiedene Systeme übertragbare Beschreibung der Umgebung an. Der Einsatz von Algorithmen, deren Struktur nicht vollständig einsehbar ist, erfordert eine solche Umgebungsbeschreibung für hochautomatisierte Fahrzeuge und dementsprechend deren Module.[331] Bestehende ODD Beschreibungen für hochautomatisierte Fahrzeuge sind allerdings zu spezifisch, um sie auf verschiedene Modulschnittstellen zu übertragen. Daher werden Attribute eingeführt, die genügend Spielraum für verschiedene Modulschnittstellen eines HAFs bieten. Darüber hinaus fließen bestehende Beschreibungen von Schnittstellen bei potentieller Übertragbarkeit für die Absicherung von Modulen mit ein. Daraus folgend wird eine neue Schnittstellenbeschreibung erarbeitet, die insbesondere die semantischen Inhalte beschreibt, die an der jeweiligen Schnittstelle übertragen werden. Diese wird zusätzlich durch die Beschreibung von Einflussfaktoren und Auswirkungen unterstützt.

Tabelle 5-1 fasst die grundlegenden Ziele einer solchen Schnittstellenbeschreibung auf Basis des übergeordneten Ziels dieser Arbeit, Module eines HAFs individuell abzusichern, zusammen. Zusätzlich ist die erwartete Auswirkung auf die Umsetzung einer modularen Absicherung bei Nichteintritt des jeweiligen Ziels erläutert. Die Ziele basieren auf den konkreteren, noch offenen Zielen der Argumentationskette nach Kapitel 4 und fassen diese entsprechend zusammen. Die Ziele nach Kapitel 4 sind im Detail in Anhang A zusammengefasst.

---

[330] Broy, M.; Kuhrmann, M.: Einführung in die Softwaretechnik (2021), S. 658.

[331] Abrecht, S. et al.: Deep Learning Safety Concerns in Automated Driving Perception (2023).

Tabelle 5-1: Allgemeine Ziele der detaillierten semantischen Schnittstellenbeschreibung.

Ziele der semantischen Schnittstellenbeschreibung	**Erwartete Auswirkung bei Nichteintritt, der die modulare Absicherung behindert.**
Anwendbar auf alle Modulschnittstellen eines hochautomatisierten Fahrzeugs.	Falls die semantische Schnittstellenbeschreibung nicht auf alle Module, die die Fahraufgabe beeinflussen, anwendbar ist, kann diese nicht für die modulare Absicherung angewendet werden.
Unterstützt die Identifikation der zu erwartenden Stimuli zur Definition von Testfällen und Testumgebungen anderer Module.	Ohne einen vorgegebenen Rahmen notwendiger Informationen können relevante Informationen vergessen werden, sodass notwendige Testfälle und Testumgebungen nicht identifiziert werden.
Stellt Informationen zur Definition von Validierungskriterien bereit.	Auf Modulebene ist ein Testausgang bzw. das beobachtbare Verhalten nicht immer direkt bewertbar. Daher werden Informationen zur Aufstellung von Validierungskriterien benötigt.
Unterstützt die Auslegung und den Test auf Robustheit mit der Spezifikation möglicher Stimuli die inhaltlich oder zeitlich vom Idealverhalten abweichen.	Eine fehlende Spezifikation möglicher Abweichungen von idealen Bedingungen oder Werten an den Modulschnittstellen bleibt für andere Module unbekannt ggü. welchen Abweichungen sie robust sein müssen. Der Test vermuteter Abweichungen ist ineffizient und wird meistens nur unvollständig durchführbar sein.
Stellt nur Informationen bereit, die für die Absicherung von Abonnenten relevant sind.	Bestehende Beschreibungen einzelner Module enthalten alle im Entwicklungsprozess gesammelten Informationen über ein Modul. Diese sind für Entwickler anderer Module nicht immer alle erfassbar, sodass ggf. relevante Informationen maskiert werden.
Stellt Informationen für den Zugriff auf die übertragenen Informationen und deren korrekte Verarbeitung bereit.	Ohne diese Beschreibung könnten Informationen fehlerhaft gelesen oder verarbeitet werden. Tests zur Überprüfung der korrekten Lesbarkeit, Verarbeitung oder Überprüfung der Informationen könnten entsprechend nicht abgeleitet werden.
Bietet eine gemeinsame Kommunikationsplattform für Systemingenieure, Modulentwickler und Tester zur Stärkung des gemeinsamen mentalen Modells.	Irrtümer entstehen u. a. durch unklare oder unvollständige Kommunikation und entsprechendem Aufbau fehlerhafter mentaler Modelle. Ohne eine klare von allen Nutzern verständliche Kommunikationsplattform ist die Wahrscheinlichkeit für Irrtümer daher erhöht.
Bietet einen Rahmen, zur Bewertung der Übertragbarkeit bestehender Datensätze nach Änderungen.	Änderungen an Modulen haben potentiell Auswirkung auf die Gültigkeit aufgenommener Datensätze. Damit wäre nach jeder Änderung die Aufnahme neuer Daten notwendig.

5.2 Analyse bestehender Schnittstellenbeschreibungen

Kapitel 2.3 stellt bereits Grundlagen zur Nutzung und Beschreibung von Schnittstellen vor. Das folgende Kapitel erweitert diese Grundlagen durch bestehende standardisierte Schnittstellenbeschreibung und Veröffentlichungen zu erweiterten Beschreibungsformen bzw. -inhalten. Die Schnittstellenbeschreibungen werden auf deren Eignung zur semantischen Schnittstellenbeschreibung, zur Unterstützung einer modularen Absicherung und im Detail zur Erreichung der definierten Ziele analysiert. Die Ergebnisse der Analyse fließen im darauffolgenden Kapitel in die Entwicklung der neuen Beschreibung mit ein. Das heißt, es werden Attribute teilweise übernommen oder für den Zweck einer modularen Absicherung adaptiert. Zuerst werden bestehende Beschreibungen an der Systemgrenze analysiert, die entsprechend für Module vor Übernahme zu adaptieren sind. Darauffolgend werden Schnittstellenbeschreibungen für Softwarekomponenten analysiert.

5.2.1 Beschreibungen an der Systemgrenze

Die in Kapitel 3.2 vorgestellten Beschreibungsformen der Umgebung eines automatisierten Fahrzeugs mit Hilfe einer Operational Design Domain (ODD) oder einer Szenariobeschreibung sind für Module übertragbar, die ebenfalls Schnittstellen zur Umgebung haben und Teil des automatisierten Systems sind. Module besitzen darüber hinaus Schnittstellen zu anderen Modulen und befinden sich mit dem hohen Umfang verschiedener, nicht vollständig vorhersehbarer Stimuli, wie das automatisierte Fahrzeug in einer komplexen Umgebung. Daher ist ein ähnlich hoher Beschreibungsanspruch für die Umgebung von Modulen notwendig, um die Einflussfaktoren ausreichend genau zu erfassen. Wie bereits erwähnt, ist die Beschreibung der ODD direkt auf Module übertragbar, die direkt von den Parametern eines Szenarios beeinflusst werden, wie z. B. die Umfeldsensoren. Allerdings beschreiben die einzelnen Parameter erstmal nur eine Szene, d.h. die Umgebung wird in einer Momentaufnahme betrachtet. Mori et al.[332] zeigen, dass bereits geringe Änderungen der Szene deutlich unterschiedliche Perzeptionsergebnisse hervorrufen können. Daher ist eine konkrete Beschreibung der Parameter, insbesondere unter Berücksichtigung des Zeitverhaltens entscheidend für die Ableitung relevanter Testfälle und für die Festlegung von deren Bestehens- oder Versagenskriterien.

Der Standard OpenScenario[333] bietet die Möglichkeit bspw. das Zeitverhalten innerhalb der Daten zu berücksichtigen. Die Standardisierung kann dazu beitragen eine große gemeinsame

[332] Mori, K. T. et al.: The Inadequacy of Discrete Scenarios (2022).

[333] ASAM e.V.: ASAM OpenSCENARIO® 2.0.0 (2022).

Datenbank zu erstellen, in der Erfahrungen aus der Praxis, d.h. z. B. zuvor unbekannte kritische Szenarien gesammelt werden. Allerdings besteht weiterhin eine Lücke zwischen der Beschreibung z. B. eines funktionalen Szenarios und den konkreten Szenarien einer OpenScenario Datenbank. Die funktionalen Szenarien (in OpenScenario) oder einer Straße (in OpenDrive[334]) stellen bisher keine Anomalien dar. Variationen, die eigentlich unzulässig sind und deswegen von den Standards nicht betrachtet werden, sind in Realität teilweise möglich. Es kann bspw. sein, dass Fahrstreifenmarkierungen abgenutzt bzw. unvollständig sind oder dass mehrere Verkehrsschilder widersprüchliche Regeln vorgeben. Der Test solcher Anomalien ist für die Sicherheitsargumentation unbedingt notwendig, wenn diese in der ODD eines hochautomatisierten Fahrzeugs nicht auszuschließen sind. Die Standards bieten zwar die Möglichkeit solche Anomalien frei zu modellieren, allerdings geht hierdurch der Vorteil der Standardisierung verloren eine gemeinsame Wissensplattform auch für Anomalien aufzubauen. Im Standard OpenDrive kann bspw. eine zwischendrin fehlende Markierung einzeln modelliert werden. Es kann jedoch angenommen werden, dass Entwickler dazu neigen, die Fahrstreifenmarkierung auf einmal für einen ganzen Streckenabschnitt zu erstellen. Die direkte Integration einer Option für Anomalien dieser Fahrstreifenmarkierungen wäre dabei einfacher und vermeidet, dass eine solche Anomalie in Tests nicht betrachtet wird.

5.2.2 Beschreibung von Softwareschnittstellen

Die Interface Definition Language (IDL)[335] beinhaltet eine breite Spanne möglicher Schnittstelleneigenschaften. Es handelt sich um eine Beschreibungssprache, die die Definition einer Schnittstelle unabhängig von einer Programmiersprache oder einem Betriebssystem ermöglicht. IDL bietet eine umfassende Definition der allgemeinen Syntax. Darüber hinaus definiert sie standardisierte Annotationen, die nicht erforderlich sind, um Informationen zu rekonstruieren, sondern diese näher zu spezifizieren. Diese Annotationen bieten eine grundlegende Semantik, haben jedoch vorwiegend das Ziel zusätzliche Informationen über die Schnittstelle senden zu können.

Die NASA[336a] stellt in ihrem System Engineering Handbook ein extra Kapitel für das Schnittstellenmanagement und die Dokumentation durch ein Interface Control Document vor. Die Vorschrift zur Schnittstellendokumentation basiert auf der Aussage, dass "die meisten Integrationsprobleme durch unbekannte oder unkontrollierte Aspekte einer Schnittstelle entstehen"[337]. Dennoch stellt das Handbuch keine Empfehlungen für eine spezifische Schnittstellenbeschreibung bereit. Die Interface Design Description (IDD) aus dem Software

[334] ASAM e.V.: ASAM OpenDRIVE (2021).

[335] Object Management Group: Interface Definition Language (2018).

[336] NASA: Expanded Guidance for NASA Systems Engineering (2016), a: S. 250–257 b: S. 256.

[337] Eigene Übersetzung nach NASA[336b]

Engineering Handbook der NASA enthält eine umfangreichere Liste von Attributen, die in sieben Gruppen unterteilt sind.[338] Die Attribute sind ähnlich wie die der IDL, sodass diese ebenfalls vorwiegend die Syntax der Schnittstellen beschreiben. Obwohl die zusätzliche Definition der "Sicherheitskritikalität" für jede Schnittstelle vorgeschlagen wird, werden keine Empfehlungen oder Beispiele zur Identifizierung und Beschreibung von sicherheitsrelevanten Attributen in Bezug auf das Verhalten der Schnittstelle dargestellt.

Die ISO 26262 empfiehlt "genau definierte Schnittstellen"[339] zur Erreichung von Modularität.[340a] Detaillierte Merkmale für die Beschreibung werden jedoch nur für Hardware-Software-Schnittstellen (HSI) vorgeschlagen.[340b] HSI unterscheidet zwischen Merkmalen (z. B. Beschreibung verschiedener Modi), Diagnosefähigkeiten und zusätzlichen Elementen (z. B. Speichertyp). Das im Rahmen des Forschungsprojekts UNICAR*agil*[341] entwickelte Konzept der serviceorientierten Architektur (ASOA) geht auf die Notwendigkeit ein, Beschreibungen der HSI bei jeder Änderung von Software- oder Hardwarekomponenten zu ändern. ASOA fungiert als Middleware und unterstützt verschiedene Computerplattformen und eingebettete Systeme. Allerdings stellt ASOA nicht sicher, dass sich die Schnittstellen zwischen den Diensten und deren Verhalten nicht ändert, falls sich ein Dienst ändert. Dieser Hauptaspekt für die Absicherung von Modulen hochautomatisierte Fahrzeuge wird daher auch mit ASOA bisher nicht adressiert.

Die AUTOSAR-Standards definieren auch Schnittstellen von Diensten, z. B. für die Umfeldwahrnehmung.[342a] Verschiedene Typen spezifizieren dabei die Schnittstelle. Jeder Typ wird durch einen Namen, seine Art, mögliche Unterelemente, seine Quelle und eine kurze Beschreibung dargestellt. Für einige Typen werden auch der mögliche Wertebereich oder eine zusätzliche Beschreibung angegeben. Die Beschreibungen der AUTOSAR-Schnittstellen beschränken sich jedoch auf nicht mehr als zwei Sätze. Stattdessen listet der Standard viele verschiedene Schnittstellentypen auf. Hiermit ist das umfangreiche Verhalten an einer Modulschnittstelle nicht vollständig beschreibbar. Anomalien werden in einigen der Signale (z. B. Lokalisierungsdaten) als extra Typen betrachtet.[342b] Anstatt jedoch eine breitere Palette an Anomalien (vgl. IEEE 1044[343]) oder die Informationsqualität (vgl. Batini & Scannapieco[344]) zu beschreiben, sind diese Typen lediglich auf die Genauigkeit beschränkt.

---

[338] Haigh, F. D.: IDD - Interface Design Description (2020).

[339] Eigene Übersetzung nach ISO 26262[340a]

[340] ISO: ISO 26262 - Road vehicles – Functional safety (2018), a: part 4, p. 10, b: part 4, p. 12, pp. 31-32.

[341] Mokhtarian, A. et al.: Dynamic Service-oriented Software Architecture (2020).

[342] AUTOSAR: Specification of Sensor Interfaces (2020), a: -, b: S. 29-31.

[343] IEEE Computer Society: IEEE Std 1044 - Software Anomalies (2010).

[344] Batini, C.; Scannapieco, M.: Data and information quality (2016).

Bachmann et al.[345] weisen darauf hin, dass sich bestehende Schnittstellenbeschreibungen lediglich auf technische Aspekte der Kommunikation zwischen Elementen in Form einer Syntax konzentrieren. Daher stellen sie eine Vorlage für eine Schnittstellenbeschreibung vor, die neun Attribute zur Beschreibung der Syntax und der Semantik enthält. Ohne den Fokus auf die modulare Absicherung und die Abhängigkeiten innerhalb eines modularen Systems zu legen, sind die vorgestellten Attribute zwar zur Ableitung von Testfällen für die Verifizierung nützlich, beinhalten jedoch nicht die Berücksichtigung von Szenarien, die Antizipation des restlichen Systemverhaltens oder die Beschreibung des Verhaltens an einer Schnittstelle.

Bass et al.[346a] übernehmen weitgehend die Attribute von Bachmann et al.[345] Sie betonen, dass die jeweiligen Stakeholder bei der Dokumentation der Schnittstellen zu berücksichtigen sind, sodass nicht zwingend jede Information über eine Schnittstelle notwendig ist. Für Systemingenieure und Tester sehen sie jedoch alle verfügbaren Informationen über eine Schnittstelle als wichtig an. Gegenüber Bachmann et al. nennen Bass et al. die Beschreibung der Fehlerbehandlung oder wie fehlerhafte Informationen an einer Schnittstelle erkannt werden können als wichtiges Element einer Schnittstellenbeschreibung.[346] Allerdings beschränkt sich der darauffolgende Vorschlag zur Beschreibung darauf, wie die Schnittstelle mitteilt, dass ein Irrtum aufgetreten ist. Hierzu ist zwar eine vorhergehende Identifikation dieser Irrtümer notwendig, dies ist jedoch nicht das Ziel der Autoren. Es fehlt daher eine Struktur zur Identifikation und Beschreibung solcher Irrtümer. Die von Bass et al.[346b] vorgeschlagenen Kompatibilitätsforderungen nach Änderungen der Schnittstelle lässt außerdem die Semantik außer Acht und berücksichtigt dem gegebenen Beispiel zufolge nur die Syntax.

Die vergleichsweise neue Norm ISO 23150[347a] beschreibt logische Schnittstellen, die den Austausch von Informationen zwischen Sensoren zur Umgebungserfassung bei hochautomatisierten Fahrzeugen und weiterverarbeitenden Instanzen ermöglichen. Hierbei werden auch mögliche semantische Informationen vorgegeben, wie bspw. die Klasse oder den Zustand von Objekten. Die Semantik bezieht sich jedoch lediglich auf die Angabe der Inhalte einzelner Parameter. Mögliche Kombinationen dieser Parameter zu einem möglichen Modulverhalten werden nicht beschrieben. Auch Einflussfaktoren auf die angegebenen semantischen Informationen sind nicht Teil der Beschreibung.[347b] Die Norm berücksichtigt die Angabe systematischer und zufälliger Irrtümer. Hierbei wird sich jedoch auf Irrtümer aufgrund von Messfehlern, d.h. während der Laufzeit bezogen. Mögliche Irrtümer in der Spezifikation oder Implementierung spielen keine Rolle.[347c] Die Norm zeigt jedoch bereits, in welchem Umfang und Detailgrad eine Schnittstellenbeschreibung mindestens erforderlich ist, um notwendige Informationen aus der Umgebung eines HAFs zu erfassen.

---

[345] Bachmann, F. et al.: Documenting Software Architecture: Documenting Interfaces (2002).

[346] Bass, L. et al.: Software Architecture in Practice, 4th Edition (2021), a: 285–288, b: S. 276-277.

[347] ISO: ISO 23150 - Data communication between sensors (2021), a: -, b: S. 18 ff., c: S. 220-226.

5.2.3 Defizite bestehender Schnittstellenbeschreibung

Der Stand der Technik und der Forschung zeigt, dass sich die Definition von Schnittstellen auf die Funktionsfähigkeit der Kommunikation zwischen Elementen konzentriert. Es wird zwar erwähnt, dass Sicherheitsaspekte berücksichtigt werden sollten, aber der Einfluss des Schnittstellenverhaltens auf das Systemverhalten und damit auf die Sicherheit wird in keiner Schnittstellenbeschreibung miteinbezogen. Aufgrund der geringen Unterschiede zwischen den Schnittstellenbeschreibungen von Bachmann et al.[348] und Bass et al.[346a], die beide auch sicherheitsrelevante Eigenschaften einbeziehen, kann davon ausgegangen werden, dass es diesem Thema an detaillierter Forschung mangelt. Diese Forschungslücke resultiert voraussichtlich daraus, dass die Verwendung von Integrations- und Systemtests in einem üblichen Entwicklungsprozess, die Irrtümer auf Modulebene kompensiert.

Eine mögliche allgemeine Einschränkung bei der Dokumentation von Schnittstellen wird von Wright[349] genannt. Er verweist auf seine Erfahrung, dass alles, was in einem Vertrag steht, immer auch von der Interpretation der genauen Implementierung abhängt. Selbst wenn andere die Implementierung nicht kennen, ist sie durch das Verhalten an der Schnittstelle beobachtbar. Wright folgert, dass sich andere schließlich an diese Beobachtungen anpassen. Er bezieht sich jedoch auf eine große Anzahl von (ggf. unbekannten) Nutzern einer Schnittstelle. Bei einem automatisierten Fahrzeug sind diese Nutzer durch die Hersteller streng kontrolliert und begrenzt. Wright führt an, dass es eine Tendenz gibt, aus Beobachtungen Annahmen zu bilden und sich entsprechend zu adaptieren, solange keine genaue Spezifikation verfügbar ist. Dies bestätigt den Einsatz einer detaillierten Spezifikation von Schnittstellen als möglichen Schlüssel zur Entwicklung und Absicherung von Modulen, die durch diesen Ansatz unabhängig von der Implementierung anderer Module werden.

5.3 Methodik und Aufbau der Beschreibung

Im folgenden Abschnitt wird der Aufbau der neuen detaillierten semantischen Schnittstellenbeschreibung hergeleitet und beschrieben. Wie in bestehenden Schnittstellenbeschreibungen stellen Attribute die Grundlage zur Beschreibung dar. Die Attribute beschreiben die Eigenschaften einer Schnittstelle im Detail. Zur Erhöhung der Übersichtlichkeit und Einordung des Nutzens der einzelnen Attribute werden diese in vier Kategorien unterteilt. Jedes dieser Attribute wird darüber hinaus durch vier verschiedene Beschreibungstypen näher erläutert. Hieraus ergibt sich die in Abbildung 5-1 dargestellte Matrix zur Beschreibung einer Schnitt-

[348] Bachmann, F. et al.: Documenting Software Architecture: Documenting Behavior (2002).

[349] Wright, H.: Hyrum's Law (2022).

stelle. Die detaillierte semantische Schnittstellenbeschreibung wird basierend auf den englischen Begriffen ihrer Attributskategorien (**S**yntax, **S**emantik, **I**nfluencing Factors, **I**mpacts) **S²I²** benannt. Die Begriffe werden im Folgenden erläutert.

		Beschreibungstypen			
		Ausdruck	Reguläre Variabilität	Irregularitäten	Ursachen für Irregularitäten
Attributskategorien	Syntax	…	…	…	…
	Semantik	…	…	…	…
	Einfluss-faktoren	…	…	…	…
	Auswirkungen	…	…	…	…

Abbildung 5-1: Aufbau der detaillierten semantischen Schnittstellenbeschreibung S^2I^2 als Übersichtsmatrix.

Kategorien für Attribute der Schnittstellenbeschreibung

Bestehende Beschreibungsformen unterscheiden zwischen der *Syntax* und der *Semantik* einer Schnittstelle. Das Konzept einer detaillierten semantischen Schnittstellenbeschreibung greift diese Unterteilung auf. Die *Syntax* legt die Struktur einer Nachricht fest. Der Zweck hiervon beinhaltet, dass sowohl der Sender eine Nachricht konstruieren und der Empfänger diese Nachricht rekonstruieren kann, ohne dass der Inhalt der Nachricht verändert wird. Zum Verständnis einer Nachricht ist neben der korrekten Konstruktion und Rekonstruktion die Beschreibung der *Semantik* notwendig. Für hochautomatisierte Fahrzeuge wurde bereits dargelegt, dass deren Absicherung einer Beschreibung der Systemumgebung inklusive des erwarteten Verhaltens anderer Akteure bedarf. Daher wird für eine Schnittstellenbeschreibung, auch die Beschreibung des Verhaltens an den Schnittstellen als Teil der *Semantik* ergänzt.

Die Beschreibung der *Semantik* kann das erwartbare Verhalten an der Schnittstelle darstellen. Für die Identifikation des für die Abonnenten relevanten Verhaltens und zur Erhöhung der Nachvollziehbarkeit dieses Verhaltens trägt die Beschreibung der *Einflussfaktoren* und der *Auswirkungen* dieses Verhaltens bei. Für hochautomatisierte Fahrzeuge ist die sicherheitsrelevante Ebene in der Regel die Fahrzeugebene, sodass die *Einflussfaktoren* und die *Auswirkungen* jeweils auf diese Ebene zurückzuführen sind. Wie in Kapitel 4.2 beschrieben ist die sicherheitsrelevante Ebene allerdings für jedes System bzw. jeden Anwendungsfall eigens zu bestimmen und nicht immer klar von höheren Ebenen trennbar. In Abbildung 5-2 ist die Beschreibung der *Einflussfaktoren* vergleichbar mit Simulationsmodellen zur Stimulation eines Moduls B schematisch dargestellt. Die Beschreibung der *Auswirkungen* ist ver-

gleichbar mit Simulationsmodellen zur Weiterverarbeitung der Modulausgaben. Die Auswirkungen bis zur potentiellen Rückspeisung der Ausgaben eines Moduls A können gemeinsam mit den Attributen der Kategorie *Einflussfaktoren* beschrieben werden. Anhand der Abbildung ist ersichtlich, dass *Einflussfaktoren* und *Auswirkungen* fusioniert oder jeweils noch mal aufgeteilt werden können. Die Aufteilung ist anhand des Fokus auf die Systemebene gewählt, die mit der Umgebung interagiert. Zukünftige Arbeiten und Erweiterungen der Schnittstellenbeschreibungen sowie die Definition weiterer Attribute können zeigen, ob eine Anpassung der gewählten Granularität Vorteile bietet.

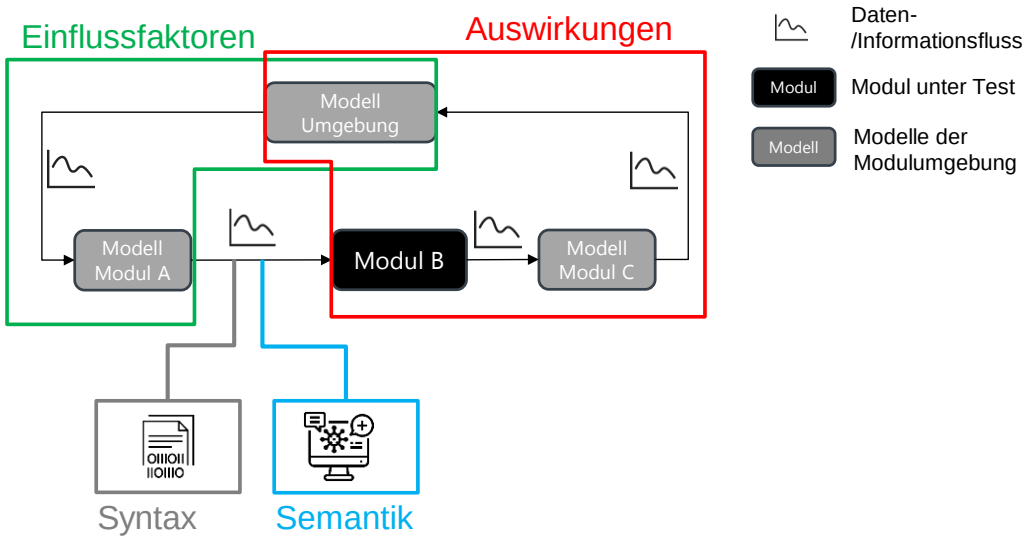


Abbildung 5-2: Schematische Darstellung der Kategorien für die Attribute von S^2I^2 bezogen auf eine Simulationsumgebung mit Modellen der Modulumgebung.

Die Beschreibung der *Einflussfaktoren* und *Auswirkungen* bezieht sich vorwiegend auf die *Semantik*. Deren Attribute lassen meist eine Variabilität zu, die entsprechend beeinflussbar ist und Auswirkungen auf die Modulumgebung hat. Die Attribute der Syntax lassen dagegen weniger Variabilität zu, da bereits geringe Änderungen dazu führen können, dass eine Nachricht nicht mehr rekonstruierbar ist. Abbildung 5-3 zeigt die Zusammenhänge der vier Beschreibungskategorien in einem UML Diagramm.

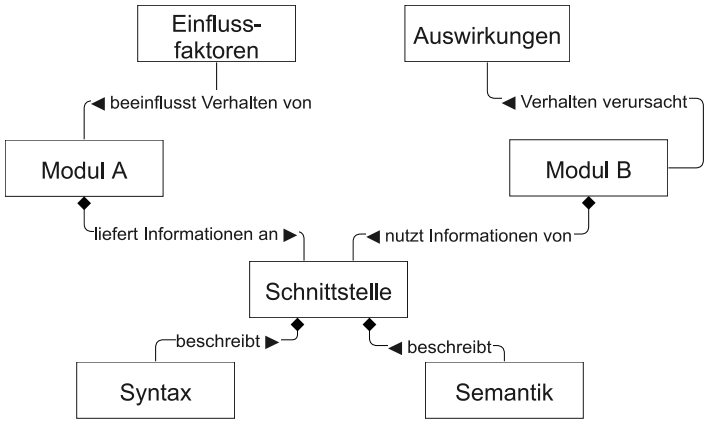


Abbildung 5-3: Zusammenhänge der vier Beschreibungskategorien von S^2I^2 dargestellt als UML-Klassendiagramm.

Beschreibungstypen der Attribute

Die Beschreibung der Attribute erfolgt initial durch natürliche Sprache und zugehörige Werteangaben. Dies bietet die höchste Flexibilität sowie den geringsten Aufwand bei der Erstellung und dem Lesen der Inhalte. Die Beschreibung der Attribute innerhalb der vier Kategorien wird in vier verschiedene Beschreibungstypen unterteilt, um die Übersichtlichkeit zu erhöhen und wichtige Beschreibungsinhalte hervorzuheben. Der grundlegende Beschreibungstyp wird als *Ausdruck* bezeichnet und mit Informationen versehen, die Zweck und Inhalt des Attributs betreffen.

Bachmann et al.[350] schlagen die Dokumentation der Variabilität einer Schnittstelle im Allgemeinen vor. Gleichzeitig betonen sie, dass daraus insbesondere der Einfluss der Variabilität auf die Semantik hervorgehen soll. Für unterschiedliche Konfigurationen eines Fahrzeugs oder einzelner Module ist es jedoch auch möglich, dass die Syntax sich differenziert darstellt, da z. B. bestimmte Werte in bestimmten Konfigurationen nicht benötigt werden. Ebenso können Einflussfaktoren und Auswirkungen je nach Konfiguration unterschiedlich ausfallen oder allgemein eine bekannte Bandbreite aufweisen (z. B. für die zeitliche Dauer eines Vorgangs). Diese Informationen werden in der Schnittstellenbeschreibung S^2I^2 als *reguläre Variabilität* bezeichnet und aufgrund ihrer beschriebenen Relevanz auf die Attribute angewendet. Die reguläre Variabilität beinhaltet alle Änderungen, die während des Betriebs für ein Attribut auftreten können, aber weiterhin ein gewünschtes Sollverhalten darstellen und die nicht bereits unter *Ausdruck* genannt sind. Die Angaben können über den Betrieb hinaus erweitert werden, wenn z. B. bereits Änderungen an anderen Komponenten geplant sind, aus denen sich Änderungen an der Beschreibung der Attribute ergeben. Durch diese Angabe können abonnierende Module diese Änderungen bereits in der Absicherung in Form eines erweiterten Gültigkeitsbereichs berücksichtigen. Die Änderungen können dadurch ggf. größer als ohne diese Angabe ausfallen, ohne dass abonnierende Module eine erneute Absicherung erfordern.

Zur Vervollständigung schließt sich an eine Beschreibung des regulären Verhaltens auch die Beschreibung eines irregulären Verhaltens an. Der Stand der Technik bezieht sich dabei bisher auf die Beschreibung von Irrtümern oder Fehlerzuständen. Allgemeiner lässt sich dies unter dem Begriff und dem zusätzlichen Beschreibungstyp *Irregularitäten* zusammenfassen. Der Begriff wird bewusst terminologisch von Fehlerbegriffen abgegrenzt, da ein irreguläres Verhalten nicht zwingend ein fehlerhaftes Verhalten darstellt. Insbesondere unter der Betrachtung einzelner Module kann nicht direkt eingesehen werden, ob ein bestimmtes Verhalten im Systemverbund ein fehlerhaftes Verhalten hervorruft. Mit der Angabe des irregulären Verhaltens können andere Module bewerten, ob diese ggü. dem Verhalten robust sind und das Sollverhalten auf Systemebene hervorrufen. Die Angabe der Irregularitäten ist wichtig,

[350] Bachmann, F. et al.: Documenting Software Architecture: Documenting Interfaces (2002), S. 12.

um abonnierende Module auf Robustheit auszulegen und ihr Verhalten darauf zu testen. Daraus lassen sich ggf. die Notwendigkeit zusätzlicher Sicherheitsfunktionen oder zuvor unbekannte Abhängigkeiten aufdecken (z. B. Common Cause Failures[351]).

Die Angabe von *Ursachen für Irregularitäten* stellt den vierten Beschreibungstyp dar und wird ebenfalls auf alle Attribute angewendet, sofern potentielle Ursachen bekannt sind. Die Angabe der Ursachen kann dazu dienen mögliche Gegenmaßnahmen zu identifizieren, die bspw. durch Implementierung von Sicherheitsfunktionen eine entsprechende Ursache verhindert oder diese während der Laufzeit erkennt und eine Reaktion einleitet, um ein gefährliches Verhalten zu verhindern. Darüber hinaus besteht die Möglichkeit, dass Ursachen für Irregularitäten bekannt sind, die jeweils ausgelöste Irregularität selbst jedoch nicht beschreibbar ist. Zur Prüfung dieser Funktionen oder zur Aufdeckung verbliebener Schwachstellen können die Angaben der Ursachen helfen, passende Testfälle zu definieren. Bei Erzeugung der Testfälle mit Hilfe von Simulationsmodellen kann es einfacher sein, dieses Modell entsprechend einer Ursache zu konfigurieren, anstatt bspw. zu versuchen einen bestimmten Werteverlauf zu erzeugen. Ein Beispiel hierfür ist eine verschmutzte Kamera als Ursache für fehlende Einträge in der Objektliste. Die Erzeugung der Objektliste ist durch die Angabe der Ursache potentiell einfacher modellierbar, als durch die direkte Manipulation der Einträge in der Objektliste. Die jeweiligen Modultester werden damit auch in der Bewertung unterstützt, ob ein Modell für einen bestimmten Test und die jeweilige Art von Irregularität valide ist. Die Ursachen aus mehreren S^2I^2 Dokumenten können darüber hinaus direkt für eine Analyse gemeinsamer Ursachen (eng.: Common Cause Analysis) genutzt werden.

Im Folgenden ist die Wahl von vier Beschreibungstypen für S^2I^2 nochmals zusammengefasst. Der Beschreibungstyp *Ausdruck* bietet bereits einen weiten Raum zur Beschreibung eines Attributs in natürlicher Sprache oder in formalen Beschreibungsformen. Die spezifische Angabe der *regulären Variabilität* stellt sicher, dass in Entwicklung und Test verschiedene Varianten oder Bedingungen berücksichtigt werden, die bekannt aber z. B. aufgrund ihrer Seltenheit nicht spezifiziert oder kommuniziert werden. Hierbei kann die Angabe der Ursachen einer Variabilität ebenso unterstützen. Allerdings ist für Abonnenten vorwiegend die Beschreibung dieser Variabilität notwendig. Die Angabe der Ursachen für Test und Entwicklung ist ggf. hilfreich, aber zweitranging, wenn die Beschreibung der Variabilität bereits vorhanden ist.

Für *Irregularitäten* hingegen werden die *Ursachen* extra aufgeführt, da einerseits nur die Irregularitäten beschrieben werden können, zu denen die jeweiligen Ursachen nicht bekannt sind, andererseits können Ursachen bekannt sein, für welche die dabei auftretende Irregularität ggf. nicht ausreichend beschreibbar sind. Die abonnierten Module haben dann entsprechend die Herausforderung, diese Abweichungen zu erkennen bzw. robust gegen diese zu

[351] Versagensfälle, die durch eine Kombination mehrerer Irrtümer oder Fehlerzustände entstehen, die allerdings alle dieselbe Grundursache haben.

sein. Hierzu kann durch die Angabe der Ursache ggf. auf zusätzliche Quellen zurückgegriffen werden.

Methodik zur Ermittlung der Attribute

Die in Kapitel 5.2 analysierten Schnittstellenbeschreibungen des Stands der Technik dienen als erste Basis zur Ermittlung der Attribute der vier vorgestellten Kategorien. Bereits existierende Attribute werden teilweise übernommen. Falls die Angabe eines Attributs nicht eindeutig ist oder die Möglichkeit als hoch eingeschätzt wird, dass bestimmte Informationen innerhalb eines Attributs zur Erreichung der Ziele von S^2I^2 nicht hinreichend dokumentiert werden, werden sie weiter verfeinert. Zusätzlich leiten sich Attribute anhand der ungelösten Ziele aus Kapitel 4 ab, die deren Erfüllung unterstützen. Die Referenzierung, der zur Herleitung der Attribute konkret verwendeten Ziele erfolgt in den folgenden Kapiteln direkt oder anhand der Übersicht aller Attribute von S^2I^2 in Anhang B.1.

Als Besonderheit werden in S^2I^2 auch Anomalien in der Beschreibung von Irregularitäten und ihrer Ursachen berücksichtigt. Daher basieren Attribute auch auf Anomalien, die sich bereits in der Literatur finden. Hierzu tragen insbesondere die 15 IQ(Informationsqualität)-Dimensionen nach Hildebrand et al.[352], die Beschreibungen von Batini und Scannapieco[353] sowie die Norm IEEE 1044[354] bei. Die jeweiligen Informationsquellen sind im Detail in Anhang B.1 mit den Attributen verknüpft.

Die folgenden Kapitel sind entsprechend der vier Attributskategorien aufgeteilt. Darin werden die einzelnen Attribute beschrieben und ihr Zweck begründet, die nach dem Stand der Technik verändert oder ergänzt wurden. Hierbei liegt der Fokus auf der Absicherung einzelner Module, sodass die Attribute als Informationsquelle für deren Implementierung und Testfallgenerierung einsetzbar sind.

5.3.1 Syntax

Harel und Rumpe[355] beschreiben die Syntax als eine Beschreibung der Notation, so dass sie von anderen Diensten oder Entwicklern verarbeitet werden kann. Dennoch ist die Kenntnis der Notationsregeln zur eindeutigen Rekonstruktion der Informationen nicht ausreichend. Daher werden als Teil der Kategorie *Syntax* folgende Attribute für ihre Rekonstruktion ergänzt. IDL der Object Management Group[356] enthält hierzu eine Liste möglicher Annotatio-

[352] Hildebrand, K. et al.: Daten- und Informationsqualität (2021).

[353] Batini, C.; Scannapieco, M.: Data and information quality (2016).

[354] IEEE Computer Society: IEEE Std 1044 - Software Anomalies (2010).

[355] Harel, D.; Rumpe, B.: Modeling Languages: Syntax, Semantics and All That Stuff (2000), S. 5.

[356] Object Management Group: Interface Definition Language (2018), S. 109–117.

nen. Die für eine modulare Absicherung anwendbaren Annotationen sind *Name*, *Identifikationsname*, *Versionsnummer*, *Standardwert*, *Einheit*, *Position* und der *verfügbare Wertebereich*.

Eine Einheit ist im Allgemeinen ein standardisiertes Referenzsystem, mit dem eine Zahl eine grundlegende Bedeutung erhält, die von einem Empfänger verstanden werden kann, der die Definition des Referenzsystems kennt. Allerdings beinhaltet nicht jede Einheit bereits eine vollständige Definition der verwendeten Referenz. Zum Beispiel kann die Position einer Trajektorie, deren Werte in der Einheit Meter angegeben werden, mit verschiedenen Koordinatensystemen dargestellt werden. Außerdem muss definiert werden, für welchen Punkt der Fahrzeugkarosserie die Position der Trajektorie gilt. Daher benötigt eine Schnittstelle ggf. die zusätzliche Beschreibung des *Referenzsystems* als Ergänzung zur Einheit für die bereitgestellten Werte oder Informationen.

Informationen können hinsichtlich ihrer allgemeinen *Klassifizierung des Informationsflusses* unterschiedlich bereitgestellt werden. Der Bereich der Signalanalyse bietet Grundlagen für eine solche Klassifizierung, insbesondere wenn die Informationen über Signale bereitgestellt werden. Beispielsweise können Signale kontinuierlich oder diskontinuierlich, periodisch oder nichtperiodisch, deterministisch oder stochastisch sein, wie Woyczyński[357] aufzeigt.

In Anhang B.1 sind insgesamt 17 verschiedene Attribute für die *Syntax* aufgeführt. Je nach Implementierung der Schnittstelle können zusätzliche Attribute erforderlich sein, damit die Teilnehmer die Informationen vollständig und unverfälscht rekonstruieren können. Hierbei wird nur die höchste Ebene des Open Systems Interconnection-Modells der Norm ISO/IEC 7498[358] betrachtet, da der Schwerpunkt auf der Anwendung der Schnittstelle liegt. Für eine vollständige Absicherung müssen auch die unteren Schichten beschrieben und getestet werden. Der Schwerpunkt liegt jedoch auf der Semantik einer Schnittstelle, die aufgrund von Spezifikationsirrtümern ungewünschtes Verhalten erzeugt.

5.3.2 Semantik

Die Kategorie *Semantik* enthält Attribute, die notwendig sind, um ausreichend gültige Stimuli, die über die Schnittstelle übertragen werden, zu charakterisieren und schließlich zu erstellen, z. B. für Modultests.

Das erste Attribut beschreibt *Zweck und Bedeutung* der Schnittstelle. Es enthält eine Beschreibung, wofür die über diese Schnittstelle transportierten Informationen aus Modul- und Systemsicht verwendet werden. Das Attribut *verfügbarer Wertebereich* in der Kategorie *Syntax* beschreibt, welche Werte technisch noch über eine Schnittstelle sendbar sind. Einige

[357] Woyczyński, W. A.: Statistics for signal analysis (2006), S. 1–3.

[358] ISO: ISO/IEC 7498 - Basic reference model (1994).

sendbare Werte können jedoch nicht notwendig oder semantisch ungültig sein. Angaben im Attribut *gültiger Wertebereich* bieten daher die Möglichkeit, diese Werte auszuschließen.

Die weitere Informationsverarbeitung durch Module, die eine Schnittstelle abonniert haben, ist nicht nur von einem einzelnen Wert abhängig, sondern wird auch durch dessen zeitlichen Verlauf beeinflusst. Neben der häufigen Referenz der Zeit sind auch andere Referenzen möglich wie bspw. die zurückgelegte Strecke. Die daraus folgende Angabe eines *Werte- oder Informationsverlaufs* ist äquivalent zur Beschreibung von Parametern innerhalb eines Szenarios auf Systemebene. Für hochautomatisierte Fahrzeuge können hier insbesondere die bereits vorgestellten Möglichkeiten zur Beschreibung von Szenarien zum Einsatz kommen. In S^2I^2 lassen sich diese jedoch für eine spezifische Schnittstelle beschreiben und damit potentiell reduzieren, wie vergleichbar durch Amersbach und Winner[359] aufgezeigt. Für Modultests ist es notwendig, mögliche Verläufe und Auslöser bzw. *Trigger* zu beschreiben, die diese Verläufe verursachen. Ein einfaches Beispiel für einen Trigger und dessen ausgelösten Werteverlauf beschreibt Viehof[360a] anhand der ausgelösten Reaktion der Gierrate auf einen Lenkwinkelsprung. In der Signalverarbeitung gibt es mehrere Ansätze, den Verlauf von Signalen, z. B. im Zeit- oder Frequenzbereich, detailliert zu beschreiben (vgl. z. B. Yarlagadda[361]). Das Ziel von S^2I^2 ist jedoch erstmal einen Überblick über die zu erwartenden Verläufe zu geben. Wie genau die Spezifikation der zu erwartenden Signale zu verfassen ist, sollte den Entwicklern und Testern überlassen bleiben. Diese können einschätzen, ob die Spezifikation ausreichend präzise und vollständig für die weitere Entwicklung und die Testvorhaben ist, sodass ggf. unnötiger Dokumentationsaufwand zur Erzeugung noch genauerer Beschreibungen vermieden wird. Die Beschreibung kann zur weiteren Detaillierung auch mit Datenbanken oder (Simulations-)Modellen verbunden werden. Initial ist eine allgemeine Beschreibung des Verlaufs notwendig, die diese Modelle mit den darin enthaltenen Ungewissheiten beschreibt (vgl. z. B. die von Viehof beschriebene Berücksichtigung der Ungewissheit bei der Validierung[360b]).

Für Module mit Einsatz von neuronalen Netzen ist eine Beschreibung möglicher Verläufe allerdings nur eingeschränkt möglich, da deren Verhalten sehr unterschiedlich und umfangreich sein kann. Dies hängt bspw. Von den genutzten Trainingsdaten ab. Das Verhalten bei Abweichungen von diesen Trainingsdaten kann für große, komplexe Netze nach dem Stand der Technik nicht zuverlässig vorhergesagt werden.[362] Eine Spezifikation des Verhaltens kann jedoch neben der üblichen Form der natürlichen Sprache auch in Form von Modellen oder von Datensätzen bis hin zu äquivalenten ggf. standardisierten Modulen erfolgen.

---

[359] Amersbach, C.; Winner, H.: Overcome the parameter space explosion (2019).

[360] Viehof, M.: Dissertation, Objektive Qualitätsbewertung durch statistische Validierung (2018), a: S. 56, b: S. 106-112.

[361] Yarlagadda, R. K.: Analog and digital signals and systems (2010).

[362] Abrecht, S. et al.: Deep Learning Safety Concerns in Automated Driving Perception (2023).

Zwischen Informationen können auch einschränkende Bedingungen bestehen. Es ist z. B. möglich, dass bestimmte Kombinationen an Informationen ungültig sind (z. B. muss ein Objekt im Stillstand eine Geschwindigkeit gleich Null haben) oder Informationen über den zeitlichen Verlauf voneinander abhängen (z. B. Verlauf der Geschwindigkeit zur abgeleiteten Beschleunigung). Die Beschreibung von informellen *Einschränkungen oder Konsistenzkriterien* ist ein weiteres Attribut, das zusätzlich mit anderen Schnittstellen verbunden werden kann.

Einschränkungen können auch zwischen Informationen innerhalb der betrachteten Schnittstelle bestehen, die als Attribut *Interdependenzen* eingeführt wird. Konsistenzkriterien oder Abhängigkeiten zwischen Informationen beruhen auf dem Vorhandensein eines gewissen Grades an *Redundanzen* in den Informationen. Eine Beschreibung der redundanten Informationen und des Zwecks der Redundanz ist daher ein weiteres Attribut der Semantik.

Während die Syntax bereits Attribute enthält, die das grundlegende Zeitverhalten beschreiben, können folgende weitere Faktoren die Semantik beeinflussen. In Steuerungsanwendungen ist eine essentielle Eigenschaft die Latenzzeit, welche die Zeit zwischen einem Ereignis und der Reaktion auf dieses Ereignis darstellt. Baun[363] definiert Latenz in der Netzwerktechnik als die Zeit, die benötigt wird, um eine Nachricht von einem Ende des Netzwerks zu einem anderen zu transportieren. Für die Schnittstellenbeschreibung ist die Reaktion, das Auftreten oder der Empfang der Information relevant, also wann sie an der Schnittstelle verfügbar ist. Der Zeitpunkt, zu dem die Information gesendet wird, gilt als Ereignis. Je nach Zweck der Schnittstelle können mehrere Ereignisse (also unterschiedliche Sendeorte) und damit auch mehrere Latenzzeiten für die weitere Verarbeitung der Information relevant sein. Die Ereignisse sollten im Attribut *Latenzen* beschrieben und mit den zugehörigen Latenzzeiten verknüpft werden. Eine gesonderte Latenzzeit wird durch das Attribut *Aktualität* beschrieben. Diese stellt das Alter der Information bzw. die Latenzzeit zwischen dem Zeitpunkt, an dem das Ereignis ursprünglich eingetreten ist, und dem Zeitpunkt, an dem es empfangen wird, dar. Die *Aktualität* kann für die Entscheidungen anderer Module, wie auf diese Ereignisse zu reagieren ist, relevant sein. Es ist insbesondere für die Verhaltensplanung von automatisierten Fahrzeugen von Bedeutung, beispielsweise entscheidet das Ereignis darüber, wer an einer 4-Wege-Kreuzung Vorfahrt hat (vgl. Verkehrsregeln des kalifornischen Department of Motor Vehicles[364]). Wenn Kreuzungsbereiche teilweise verdeckt sind, werden Objekte deutlich später detektiert (z. B. in vorherigen Tests). Für den Verhaltensplaner kann dies den Anschein erwecken, als sei das Ego-Fahrzeug früher als das andere Objekt an der Kreuzung angekommen und habe die Vorfahrt genommen. Die Bestimmung der *Aktualität*

[363] Baun, C.: Computernetze kompakt (2020), S. 30.

[364] California Department of Motor Vehicles: Driver handbook (2020), S. 36.

ist während der Laufzeit selten möglich, da z. B. Ground-Truth[365, 366] Zustandsinformationen von anderen Objekten nicht verfügbar sind. Es besteht ggf. jedoch die Möglichkeit die *Aktualität* zu schätzen. Daher sollten Aktualitätsmaße bereits während der Entwicklung und der Tests bestimmt werden.

Die *Verfügbarkeit* der Schnittstelle oder des Signals kann von einem gestarteten System oder Modul abhängen, wie lange es von einem bestimmten Ausgangspunkt aus dauert, oder welche äußeren Bedingungen vorher eintreten müssen. Beispielsweise kann es notwendig sein, dass ein Sensor beim Start kalibriert werden muss, bevor er ausreichend genaue Informationen sendet. Die *Vor- und Nachbedingungen* sowie andere *Unterbrechungsmechanismen* sind zwei weitere Attribute, die die Verfügbarkeit der Informationen an einer Schnittstelle charakterisieren. Sie beschreiben das Zeitverhalten in der Zeit vor und nach dem Empfang, der Verarbeitung und der Ausgabe der Informationen sowie die Bedingungen, die zu einer Unterbrechung der Informationsausgabe führen.

Informationen über bekannte mögliche Irrtümer an einer Schnittstelle können ggf. nicht den Irregularitäten eines Attributs zugeordnet werden. Daher werden im Folgenden bestehende Informations- oder Datenqualitätskategorien (u. a. als Gesamtübersicht beschrieben von Wang und Strong[367] und in der Norm ISO/IEC 25012[368]), die auch auf Schnittstellen anwendbar sind, als zusätzliche Attribute der Semantik eingeführt. Häufig beschriebene Fehlerarten sind die *Genauigkeit* und die *Präzision* eines Wertes. Handelt es sich bei einem Parameter nicht um eine Größe, die mit einem Referenzsystem verbunden ist, kann stattdessen auf den Begriff *Korrektheit* verwiesen werden, wie z. B. von Batini und Scannapieco[369] beschrieben. Metriken zur Beschreibung der Korrektheit können eine Beschreibung in natürlicher Sprache sein und in verschiedene Stufen unterteilt werden, um zumindest eine Ordinalskala für die Vergleichbarkeit bereitzustellen. So kann beispielsweise ein Fußgänger fälschlicherweise als Verkehrsschild klassifiziert werden und somit eine niedrigere Korrektheitsbewertung erhalten als die ebenfalls inkorrekte Klassifizierung als Tier. Die Bewertung basiert auf der vorhergesagten Bewegung dieses Objekts, der daraus resultierenden Verhaltensentscheidung und den daraus resultierenden Auswirkungen auf die Sicherheit. Darüber ist eine Art Instabilität der Semantik denkbar, d.h. die Informationsqualität schwankt. Diese Phänomene und ihre Ursachen werden unter dem Attribut *Informationsvolatilität* beschrieben. Schließlich können Informationen zwar verfügbar, aber unvollständig sein. Das dazu eingeführte Attribut *Vollständigkeit* ist jedoch nur auf eine Zusammenstellung an Informationen anwendbar, in der es eine Unvollständigkeit der Information geben kann.

---

[365] Bezeichnung für die wahren Informationen oder eine Referenz, die für den Anwendungsfall ausreichend geringe Unsicherheiten ggü. den wahren Informationen aufweist.[366]

[366] ISO: ISO/DIS 21448: Safety of the intended functionality (2022), S. 17.

[367] Wang, R. Y.; Strong, D. M.: What Data Quality Means to Data Consumers (1996).

[368] ISO: ISO/IEC 25012 - Data quality model (2008).

[369] Batini, C.; Scannapieco, M.: Data and information quality (2016), S. 45–46.

Ebenen der Semantik

Schnittstellen eines Moduls werden erst während des Entwicklungsprozesses final definiert. Sie legen die Grenzen eines Moduls fest und bestimmen die Informationen, die an andere Module weitergegeben werden. Ähnlich den Ebenen der Systemarchitektur gibt es auch unterschiedliche Ebenen von Schnittstellen.

Feingranulare Schnittstellen bieten die Möglichkeit, technische Details zu beschreiben, ohne die Aufmerksamkeit von Entwicklern und Testern durch zu viele unterschiedliche Elemente der Schnittstelle zu stark zu beanspruchen. Dennoch bieten feingranulare Schnittstellen einen geringeren Umfang an Semantik als grobgranulare Schnittstellen. Die feinste technisch relevante Granularität einer Schnittstelle wäre bspw. die Berücksichtigung von elektrischen Strömen oder übertragenen Bits. Die detaillierte semantische Schnittstellenbeschreibung für eine modulare Absicherung soll das Verhalten an der Schnittstelle und ihre Verbindung zu anderen Modulen und der Systemebene beschreiben. Die zweckmäßige Granularität für eine solche Schnittstelle soll eine atomare[370] Semantik beinhalten, die von der Stärke der Verbindungen und dem Zweck einer Schnittstelle abhängt. Zum Beispiel kann die Drehung eines Rades durch die Rotationsgeschwindigkeit ausreichend genau beschrieben sein. Die Schnittstelle „Trajektorie“ in UNICAR*agil* enthält hingegen 20 Elemente bzw. Zellen zur Beschreibung der beabsichtigten Bewegung des Fahrzeugs zur Übermittlung an den Fahrdynamik- und Trajektorienregler.[371] Falls also der Ausdruck der Attribute nicht beschrieben werden kann, wurde die Schnittstelle möglicherweise zu feingranular gewählt. Dennoch kann es sinnvoll sein, einzelne Attribute zusätzlich für einzelne Elemente zu beschreiben, die Teil einer komplexeren, übergeordneten Schnittstelle sind. Dies wird in Anhang B.2 anhand eines Beispiels für die Gesamttrajektorie (höhere Ebene der Semantik) als Struktur und die Beschleunigung, die eine einzelne Zelle der Trajektorie darstellt (niedrigere Ebene der Semantik), veranschaulicht.

5.3.3 Einflussfaktoren

Die nächste Kategorie von S^2I^2 beschreibt typische Einflussfaktoren auf die Semantik einer Schnittstelle. Erstens können sich bei der *Initialisierung*, dem *Herunterfahren* oder anderen definierten *Modi* oder *Moduswechseln* die Ausgaben, von denen während der Laufzeit im normalen Betriebsmodus unterscheiden. Daher wird in S^2I^2 neben der Angabe des jeweiligen Einflussfaktors auch deren Einflüsse auf die Semantik, wie z. B. den Werteverlauf bei einem Betriebsmoduswechsel, beschrieben. Ebenso muss für den Fall, dass unterschiedliche *Konfigurationen* des Moduls oder des Gesamtsystems und anderer Module existieren, der Ein-

[370] „Atomar“ wird als Begriff aus der Physik übertragen, um eine nicht weiter (logisch) zerteilbare Einheit zu beschreiben.

[371] Homolla, T.: Dissertation, Gekapselte Trajektorienfolgeregelung für autonomes Fahren (2023), S. 159.

fluss auf die Semantik beschrieben werden. Alternativ können Konfigurationen auch in anderen Versionen des Schnittstellendokuments beschrieben sein. Dies ist insbesondere dann zu empfehlen, wenn die Abweichungen von der Ursprungskonfiguration hoch sind. Im Sinne einer Absicherung hochautomatisierter Fahrzeuge beschreibt das Attribut *Systemszenarien* Szenarien auf Systemebene, die signifikant unterschiedliches Verhalten an der betrachteten Schnittstelle hervorrufen. Einige Systemszenarien können bereits im Attribut *Trigger* für den Werteverlauf berücksichtigt werden. Die jeweilige sicherheitsrelevante Hierarchieebene erfordert jedoch eine zusätzliche Berücksichtigung in einem eigenen Attribut. Aufgrund der Erwartung einer hohen Menge an Systemszenarien existieren bereits formale Beschreibungsformen (vgl. Szenariobeschreibung für automatisierte Fahrzeuge, z. B. von Bagschik[372]). Darüber hinaus wird das *Verhalten anderer Module* in S^2I^2 beschrieben, die die Semantik einer Schnittstelle maßgeblich beeinflussen. Dies gilt ebenso für Module oder Systeme der Systemumgebung. So ist beispielsweise die Liste der erkannten Objekte abhängig von der aktuellen Position, dem Kurs oder der Position der Objekte in der Umgebung. Dies scheint trivial, führt aber weiter zu dem Einfluss, dass weitere Objekte hinter einem erkannten Objekt verdeckt werden. Eine Übersicht solcher signifikanten Einflüsse unterstützt deren Berücksichtigung z. B. in Simulationsmodellen. Ein erster Vorschlag zur formalen Beschreibung dieser Abhängigkeiten wird von Kohls[373] beschrieben. Schließlich können der *Vorverarbeitungs- und Transferkanal,* die *Transportmethode* und die *vorherige Quelle* die Semantik beeinflussen und werden daher als zusätzliche Attribute eingeführt. Die Einflussfaktoren können innerhalb des Attributs beschrieben oder mit anderen semantischen Attributen, wie dem Werteverlauf verbunden werden.

Die Beschreibung von Einflussfaktoren findet sich in ähnlicher Form in Risikoanalysen wieder wie bspw. der Fehlermöglichkeits- und Einflussanalyse (FMEA). Die FMEA ermittelt auf Basis möglicher Fehlerarten potentielle Ursachen z. B. unter Nutzung der Funktionsarchitektur des Systems. Dementsprechend werden Einflussfaktoren auf das identifizierte fehlerhafte Verhalten ermittelt. Verhalten, das wie spezifiziert auftritt wird daher nicht betrachtet, sodass Irrtümer der Spezifikation in einer FMEA nicht aufdeckbar sind. Außerdem werden nicht explizit die Schnittstellen des Systems, sondern Komponenten und deren Implementierung und Funktionsbeschreibung betrachtet, was mit den in Kapitel 5.1 beschriebenen Nachteilen einhergeht.

5.3.4 Auswirkungen

Bei einzeln abgesicherten Modulen muss sichergestellt werden, dass der Ausgang eines Moduls kein unsicheres Verhalten auf Systemebene verursacht. Dazu ist das *erwartete Verhal-*

[372] Bagschik, G.: Dissertation, Systematischer Einsatz von Szenarien (2022), S. 74–78.

[373] Kohls, S. K.: Master Thesis, Abhängigkeiten zwischen Modulen (2022), S. 35–51.

ten auf Systemebene zu beschreiben, das durch Stimuli der spezifizierten Schnittstelle hervorgerufen wird. Die Beschreibung des erwarteten Verhaltens soll zunächst anderen Modulen als Information darüber dienen, was von dem Modul erwartet wird, und um Missverständnisse oder andere Irrtümer während der Entwicklung zu vermeiden. Solche Vorhersagen sind jedoch nur bei sehr einfachen Systemen genau genug, ohne das gesamte System dynamisch zu betreiben. Daher sollte geprüft werden, ob die Vorhersagen für die Entwicklung und Prüfung ausreichend valide sind oder ob Simulationsmodelle oder sogar Systemtests erforderlich sind, um die Vorhersagbarkeit zu erhöhen. Es sind mehrere Iterationen von S^2I^2 zu erwarten, einschließlich Tests in der Simulation und im realen System, bis diese Vorhersage valide ist. Beim schnittstellenbasierten Testen wird der Wertebereich in der Regel in kleinere Bereiche bzw. Äquivalenzklassen aufgeteilt. Innerhalb der Äquivalenzklassen wird davon ausgegangen, dass die Stimuli das gleiche resultierende Verhalten erzeugen. In S^2I^2 werden die Informationen an der Schnittstelle entsprechend im Attribut *Äquivalenzklassen* beschrieben, abhängig von deren Auswirkungen auf andere Module. Auf Basis dieses Zwecks ist es hilfreich, die weiteren *Verarbeitungsschritte durch andere Module* zu beschreiben. Zur Unterstützung dieser Beschreibungen sind *mögliche Module / Abonnenten* aufzuführen, die die Informationen nutzen bzw. die Schnittstelle abonnieren.

Zur Bewertung der Auswirkung von Modulausgaben und deren Weiterverarbeitung durch andere Module bis hin zur sicherheitsrelevanten Hierarchieebene ist die Beschreibung einer *Relevanzmetrik* von Vorteil. Informationen oder Charakteristika der Ausgaben haben möglicherweise keine Relevanz für die weitere Verarbeitung und das Systemverhalten. Planerzentrierte Metriken für automatisierte Fahrzeuge werden bspw. von Philion et al.[374] erforscht. Diese Metriken gewichten Informationen der Umfeldwahrnehmung, die für die Verarbeitung durch einen Verhaltensplaner relevant sind, höher als weniger relevante Informationen.

Die Beschreibung von Auswirkungen findet sich in ähnlicher Form in Risikoanalysen wie bspw. der Fehlermöglichkeits- und Einflussanalyse (FMEA) wieder. Die FMEA ermittelt mögliche Fehlfunktionen und analysiert deren Auswirkungen auf das Systemverhalten. Die Ausgangslage ist dabei die Beschreibung einer Systemstruktur z. B. durch Prozessdiagramme. Schnittstellen werden dementsprechend nur indirekt betrachtet. Darüber hinaus konzentriert sich die FMEA auf Abweichungen von der spezifizierten Funktion, das heißt ausschließlich auf Irrtümer bei der Implementierung. Entspricht also ein Modulverhalten an einer Schnittstelle dem spezifizierten Verhalten, wird dies in der weiteren Analyse der Auswirkungen nicht betrachtet. Irrtümer in der Spezifikation sind mit dieser Vorgehensweise nicht aufdeckbar.

[374] Philion, J. et al.: Evaluate Perception Models Using Planner-Centric Metrics (2020).

5.4 Anwendungsbeispiel

Nach der Einführung der verschiedenen Attribute von S^2I^2 wird in diesem Abschnitt die Anwendung anhand von zwei Beispielen aus dem Projekt UNICAR*agil* demonstriert. Beide Beispiele enthalten nur exemplarische Beschreibungen ihrer Attribute. Sie sind weder vollständig, noch enthalten sie durchgehend reale Beschreibungen aus UNICAR*agil*. Die Beispiele demonstrieren jedoch eine repräsentative Beschreibung von S^2I^2. Für das erste Beispiel wird die Trajektorie gewählt, die vom Verhaltens- und Trajektorienplaner erzeugt und vom Trajektorienregler abonniert wird. Hierfür wird die eigentliche Trajektorie vereinfacht. Statt der 20 Einträge mit verschiedenen Parametern wird lediglich die Gesamttrajektorie als übergeordnete Schnittstelle und ein Eintrag, der Beschleunigungsbetrag, beschrieben. Datentechnisch werden die Trajektorie als Struct und der Beschleunigungsbetrag als Cell bezeichnet. Die genaue Spezifikation der Trajektorie im Projekt wird in der Dissertation von Homolla[375] beschrieben. Auf den folgenden Seiten werden beispielhafte Attribute von S^2I^2 für jede der vier Kategorien in Tabelle 5-2 bis Tabelle 5-5 dargestellt. In Tabelle 5-4 mit Attributen der Einflussfaktoren wird auf die Spalte zur Beschreibung der Beschleunigung (d.h. die Spalte „Ausdruck Cell“) verzichtet, da diese durch die Beschreibung der Trajektorie bereits abgedeckt ist. Die Trajektorie wird für alle Attribute in Anhang B.2 beschrieben.

Die Trajektorie stellt ein Beispiel aus dem Bereich der Informationstechnik dar, wofür S^2I^2 ausgelegt ist. Trotzdem lässt sich die Beschreibung teilweise auch in der Mechanik einsetzen. Dies wird im zweiten Beispiel in Anhang B.3 anhand der Schnittstelle zwischen einer Motorwelle und einem Getriebe demonstriert. Im Beispiel wird offensichtlich, dass einige Attribute nicht anwendbar sind, da sie ursprünglich aus der Softwareperspektive motiviert sind. Andere Attribute sind jedoch auch für mechanische Schnittstellen anwendbar. Für eine modulare Absicherung unter Einbezug mechanischer Schnittstellen werden jedoch ergänzende Attribute und Beschreibungsformen benötigt.

Im Beispiel für die Attribute der Syntax nach Tabelle 5-2 fällt auf, dass Einträge zur Beschreibung der regulären Variabilität, der Irregularitäten und zu Ursachen für Irregularitäten teils nicht anwendbar oder nicht bekannt sind. Dies resultiert daraus, dass die Beschreibungstypen zur Beschreibung von Abweichungen von idealen Werten bzw. von einem Sollverhalten geschaffen sind. Die Syntax beinhaltet allerdings einen Großteil an Attributen, bei denen Abweichungen von Sollwerten zu einem vollständigen Funktionsverlust führen können, da Nachrichten nicht mehr rekonstruierbar sind. Durch dieses direkt beobachtbare Verhalten und die geringen Variationsmöglichkeiten der Attribute sind solche Irrtümer allerdings gut aufdeck- und vermeidbar. Dies ist bspw. am Datentyp ersichtlich, dessen Änderung dazu führen würde, dass die Nachricht nicht mehr korrekt konstruiert bzw. rekonstruiert wird. Anders verhält sich dies bei Angabe der Quelle einer Nachricht, die sich ggf. ändert, wenn

[375] Homolla, T.: Dissertation, Gekapselte Trajektorienfolgeregelung für autonomes Fahren (2023), S. 159.

ein anderes Modul die Informationen bereitstellt. Abhängig von der verwendeten Middleware kann die Information zur Quelle einer Nachricht notwendig sein, um diese zu abonnieren. Dabei kann es auch zur angegebenen Irregularität kommen, dass die Nachricht gleichzeitig von zwei verschiedenen Quellen bereitgestellt wird, bspw. falls die Middleware Dienste fehlerhaft verknüpft. Empfänger der Nachrichten können sich ggf. auf solche Irregularitäten einstellen, indem bspw. ein weiterer Indikator zur Identifikation der aktuell gültigen Nachricht eingeführt wird. Darüber hinaus gibt es im Beispiel für die Trajektorie keine Zuordnung einer Einheit oder eines Referenzsystems. Diese treffen jeweils nur auf einzelne Einträge bzw. Cells der Trajektorie zu und sind jeweils unterschiedlich. Die Klassifikation des Informationsflusses ist im Beispiel dagegen für alle Cells gleich, sodass die Angabe nur für das Struct erfolgt.

Im Beispiel für die Attribute der Semantik nach Tabelle 5-3 ist ersichtlich, dass für Zweck und Bedeutung die reguläre Variabilität und Irregularitäten als nicht bekannt angegeben sind, obwohl dies üblicherweise keine dynamische Eigenschaft ist. Allerdings kann sich die Bedeutung und der Zweck je nach Nutzung durch andere Module und abhängig von der jeweiligen Einsatzumgebung auch ändern, sodass die Beschreibungstypen auf das Attribut anwendbar sind. Die Angabe eines gültigen Wertebereichs ist für die Trajektorie als Struct dagegen nicht anwendbar. Der gültige Wertebereich hilft aber den Empfängern sich auf entsprechende Werte einzustellen. Mit Angabe der regulären Variabilität kann Robustheit gegenüber solchen Sprüngen geschaffen oder alternativ abgestimmt werden, dass solche Sprünge unzulässig sind. Ebenso kann anhand der Angabe möglicher Irregularitäten eine Überwachung dieser Werte und daraus folgend ggf. eine Sicherheitsfunktion aktiviert werden. Bei der Angabe des Werte- und Informationsverlaufs der Trajektorie in der Tabelle wird ferner auf Datenbanken verwiesen, die diese Trajektorien bspw. anhand von Manövern kategorisieren. Blödel[376] unterscheidet hierfür bspw. zwischen neun Standardmanövern und erstellt daraus konkrete Trajektorien. Für den Beschleunigungsbetrag sind dagegen beispielhafte Werteverläufe beschrieben. Dies dient der Demonstration einer sprachlichen Beschreibung. Auch hier kann allerdings die Nutzung einer Datenbank an Werteverläufen nützlich sein, um die Verläufe detaillierter und übersichtlicher zu beschreiben.

Die Angabe der Aktualität unterstützt die verarbeitenden Module bei der Bewertung der Qualität der Informationen. Eine veraltete Trajektorie verursacht ggf. die Gefahr, dass Objekten nicht mehr rechtzeitig ausgewichen werden kann. Hieraus können Maßnahmen eingeleitet werden, die z. B. einen sicheren Stopp des Fahrzeugs auslösen. Die Information der regulären Variabilität der Aktualität kann zusätzlich dazu dienen, die verarbeitenden Module darauf zu prüfen, ob diese mit den entsprechenden Schwankungen durchgehend das gewünschte Sollverhalten erzeugen. Gleichermaßen gilt dies für die Angabe der Verfügbarkeit. Deren Beschreibung der regulären Variabilität vermeidet bspw., dass ein Prozess im Trajektorienregler zu früh startet und somit eine nicht korrekte Trajektorie abfährt oder der Verarbeitungsprozess fälschlicherweise abgebrochen wird. Die Angabe der Irregularitäten sorgt

[376] Blödel, A.: Master Thesis, Verfahren zur Generierung von Trajektorien (2021), S. 41–50.

wiederum dafür, dass reale Irrtümer erkannt werden können und darauf entsprechend reagiert werden kann. Die Angabe der Ursachen für solche Irregularitäten unterstützt die Generierung von Szenarien, in denen solche Irregularitäten auftreten. Hierdurch fällt einerseits die Simulation solcher Szenarien leichter, da z. B. die möglichen Zeiten genauer bestimmbar sind, andererseits lassen sich dadurch mögliche Kombinationen an Bedingungen identifizieren, die das Verhalten des Trajektorienreglers zusätzlich beeinflussen.

Das Beispiel zur Beschreibung der Attribute der Kategorie Einflussfaktoren nach Tabelle 5-4 demonstriert, welche Verarbeitungsschritte anderer Module oder des Systems Einfluss auf die Semantik der Trajektorie haben. Die Angabe der Initialisierungsprozesse des Systems beeinflussen direkt die zuvor beschriebene Verfügbarkeit der Trajektorie. Dieser direkte Einfluss ist auch in Anhang B.2 anhand der modellierten Attribute nach Abbildung B-1 ersichtlich. Die Irregularitäten und deren Ursachen sind dabei auf Systemebene oder in Bezug auf andere Module beschrieben. Die Attribute Moduswechsel und Systemszenarien haben direkten Einfluss auf den Werte- und Informationsverlauf der Trajektorie. Die Beschreibung der Ausdrücke, Variabilität und Irregularitäten unterstützt deren Modellierung z. B. für eine Simulationsumgebung. Die vorherige Modellierung der Attribute in einem Diagramm unterstützt dabei die Identifikation weiterer Einflussfaktoren oder dessen Kombinationen von der Systemebene.

Im Beispiel der Attribute der Kategorie Auswirkungen nach Tabelle 5-5 sind mit der Angabe der Äquivalenzklassen für eine Trajektorie und für den Beschleunigungsbetrag einer Trajektorie zwei stark unterschiedliche Schnittstellenebenen ersichtlich. Der Beschleunigungsbetrag ist in Äquivalenzklassen eingeteilt, die um den Betrag 0 feiner als für höhere oder niedrigere Werte diskretisiert sind. Dies wird so demonstriert, da es möglich ist, dass für den Bereich um den Betrag Null eine höhere Auftretensdichte von Irrtümern erwartet wird. Für höhere und niedrigere Werte werden Irrtümer des Trajektorienreglers zwar als wahrscheinlicher angesehen, gleichzeitig wird aber erwartet, dass Tests einzelner Grenzwerte die Irrtümer bereits aufdecken. Für die Trajektorie ist kein Wertebereich definiert. Allerdings lässt sich die Trajektorie bspw. in Manöver unterteilen, von denen erwartet wird, dass sich diese auf andere Module und auf Systemebene vergleichbar auswirken. Diese Manöver können ggf. weiter unterteilt werden z. B. anhand der dabei initiierten Beschleunigungswerte oder Geschwindigkeiten. Die Unterteilung wird dabei anhand der Auswirkungen auf andere Module oder das System bestimmt. Dies geht einher mit dem Beispiel zur Beschreibung des Verhaltens auf Systemebene. Im Beispiel ist das allgemein erwartete Verhalten in Bezug auf Trajektorien und der Beschleunigungsbeträge angegeben. Die reguläre Variabilität bezieht sich auf den definierten Akzeptanzbereich der Ist-Beschleunigung. Die Irregularität führt den nicht akzeptierten Bereich auf. Ursachen hierfür können sehr umfangreich und stark unterschiedlich sein. Zur systematischen Herleitung sollte daher eine eigene Gefährdungs- und Risikoanalyse durchgeführt werden. In Bezug auf Schnittstellen und deren Interaktion mit dem System eignet sich hierzu bspw. die systemtheoretische Prozessanalyse (STPA).

Tabelle 5-2: Ausschnitt zur Beschreibung der Kategorie Syntax beispielhaft für die Schnittstelle Trajektorie. (n/a = nicht anwendbar, n/b = nicht bekannt)

Attribut	Ausdruck Struct	Ausdruck Cell	Reguläre Variabilität	Irregularitäten	Ursachen für Irregularität
Name	Trajektorie	Soll-Beschleunigungsbetrag	n/a	n/a	n/a
Identifikations-name	trajectory	.acc_enu_ mag_m_s2	n/a	n/a	n/a
Versionsnum-mer	0.1	n/a	Änderungen der Dezimal-stelle bedeuten, dass S^2I^2 von einer Änderung nicht betroffen ist.	n/a	n/a
Datentyp	Struct	Sequence double	n/a	n/b	n/b
Signallänge	n/a	64 bit	n/a	n/b	n/b
Standardwert	n/a	0	n/a	n/a	n/a
Quelle	Verhaltens- und Trajektori-enplaner (VTP)	n/a	Die Quelle kann sich bei Modusänderungen oder nach Updates ändern.	Eingabe aus zwei verschie-denen Quellen gleichzeitig.	Falsche Verknüpfung von Diensten durch Middle-ware.
Einheit	n/a	m/s^2	n/a	n/a	n/a
Referenzsystem	n/a	Polarkoordinatensystem, Ursprung in der Mitte zwi-schen Verbindungslinie der Räder.	n/a	n/a	n/a
Eltern / Struct	n/a	trajectory	n/a	n/a	n/a
Position (auf Struct)	n/a	8	n/a	n/a	n/a
Größe / Anzahl	9 verschiedene Parameter	Vektor mit 51 Elementen	n/a	n/a	n/a
Verfügbarer Wertebereich	n/a	[0; 100] m/s^2	n/a	n/b	n/b
Diskretisierung	n/a	0.01 m/s^2	Keine	n/b	n/b
Klassifikation des Informati-onsflusses	Periodisch, diskontionuier-lich	n/a	n/a	n/a	n/a

Tabelle 5-3: Ausschnitt zur Beschreibung der Kategorie Semantik beispielhaft für die Schnittstelle Trajektorie. (n/a = nicht anwendbar, n/b = nicht bekannt)

Attribut	Ausdruck Struct	Ausdruck Cell	Reguläre Variabilität	Irregularitäten	Ursachen für Irregularität
Zweck / Bedeutung	Die Trajektorie stellt die geplanten Posen des Fahrzeugs über die Zeit dar.	Betrag der Soll-Beschleunigung in Polarkoordinaten für jedes Trajektorienelement.	n/b	n/b	n/b
Gültiger Wertebereich	n/a	[0; 15] m/s^2	Einzelner Wert springt auf 100 m/s^2, falls Ableitung der Geschwindigkeit mit $\Delta t = 0$.	Mehr als ein Wert über dem gültigen Bereich innerhalb eines Zeitraums von 5 s.	Ein Irrtum bei Initialisierung der Startposition führt zu Sprüngen in der Zielgeschwindigkeit.
Werte- oder Informationsverlauf	-Beschreibung von Trajektorien, die auf repräsentativen Manövern basieren.-	Abhängig von Trajektorie. Sprunghafte Änderungen > 5 m/s^2 bei Notmanövern.	Beschleunigungssprung bei Anfahren und Anhalten bis zu 2 m/s^2.	Sprunghafte Änderungen > 5 m/s^2 obwohl kein Notmanöver initiiert.	Falsch-Positive Detektion von dynamischen Objekten.
Trigger	Situationsänderungen; Entscheidungsänderungen	n/a	n/b	n/b	Falsche Verknüpfung von Diensten durch ASOA.
Interdependenzen	Position, Geschwindigkeit, Beschleunigung.	n/a	n/b	n/b	n/b
Redundanzen	Position, Geschwindigkeit, Beschleunigung.	Ableitung der Geschwindigkeit über die Zeit.	n/b	n/b	n/b
Aktualität	500 ms, nachdem sich eine Situation auf Systemebene geändert hat, die zu einer neuen Trajektorie führt.	n/a	+/- 150 ms	Bis zu 1500 ms	Objekt wird nach Änderung der Situation vergleichsweise spät erkannt.
Verfügbarkeit	Verfügbar 5 Sekunden nach Start des Trajektorienplaners.	n/a	Die Startprozedur kann sich ändern, so dass die Zeitdauer kürzer oder länger sein kann.	Verfügbar erst nach mehr als 5 Sekunden.	Keine gültigen Lokalisierungsdaten verfügbar.
Unterbrechungsmechanismen	Eine gültige Trajektorie kann nicht berechnet werden.	Abweichungen bei der Berechnung vom gewünschten Wert.	n/b	n/b	n/b
Informationsvolatilität	Trajektorie ändert sich, wenn sich Umgebungsmodell oder Verhaltensentscheidung ändert.	n/a	Trajektorie ändert sich kontinuierlich durch Aktualisierungen des Umfeldmodells.	Trajektorie ändert sich plötzlich mit hoher Abweichung der Posen zur vorigen Trajektorie.	Umfeldmodell oder Verhaltensentscheidung ändert sich plötzlich in hohem Umfang (z. B. Notbremsung).
Vollständigkeit	Letzte 10 Elemente der Trajektorie nicht erforderlich.	n/a	0 bis 10 der letzten Elemente können fehlen.	Es fehlen mehr als 10 Elemente.	Berechnung der neuen Trajektorie zu langsam.

Tabelle 5-4: Ausschnitt zur Beschreibung der Kategorie Einflussfaktoren bsph. für die Schnittstelle Trajektorie. (n/a = nicht anwendbar, n/b = nicht bekannt)

Attribut	Ausdruck Struct	Reguläre Variabilität	Irregularitäten	Ursachen für Irregularität
Initialisierung	Nach dem Start findet eine Initialisierung statt, bis das Umgebungsmodell aufgebaut und die erste Trajektorie berechnet ist.	Die Initialisierung dauert 10 bis 30 Sekunden.	Mehr als 30 Sekunden für die Initialisierung erforderlich bei erhöhter Ungewissheit im Umweltmodell z. B. aufgrund von Regen.	Das Fahrzeug befindet sich nicht in der spezifizierten ODD.
Moduswechsel	Automation→ Sicheres Anhalten: Neue Trajektorie, basierend auf gewünschter Notbahn.	Ruckartige Bewegung aufgrund inkonsistenter Bewegungsdaten ggü. der vorherigen Trajektorie.	n/b	n/b
Systemszenarien	Fußgänger betritt Fußgängerüberweg → Trajektorie mit Verzögerung vor Fußgängerüberweg.	Regen verringert Reibkoeffizient. → Verzögerungsanforderung übersteigt verfügbare Bremskraft.	Fußgänger betritt plötzlich die Straße. → Trajektorie beinhaltet sprungartig Ausweichmanöver.	Fußgänger versucht den Bus zu erwischen.
Vorverarbeitungs- und Transferkanal	Transformation von Sollpositionen in ein lokales Koordinatensystem durch Dienst zur Vorverarbeitung.	Vorverarbeitung nicht erforderlich, da andere Dienste das gleiche Referenzsystem nutzen.	Ungenauigkeiten der Trajektorien-Posen durch Transformation.	Verwendung der falschen Bezugspunkte für die Transformation.

Tabelle 5-5: Ausschnitt zur Beschreibung der Kategorie Auswirkungen bsph. für die Schnittstelle Trajektorie. (n/a = nicht anwendbar, n/b = nicht bekannt)

Attribut	Ausdruck Struct	Ausdruck Cell	Reguläre Variabilität	Irregularitäten	Ursachen für Irregularität
Äquivalenzklassen	Pfade, die sich durch verschiedene Manöver unterscheiden.	[-10; -8; -5; -2; -0.5; -0.25; 0; 0.25; 0.5; 1; 2] m/s^2	n/b	n/b	n/b
Verhalten auf Systemebene	Fahrzeuge sollen sich entsprechend der Trajektorie fortbewegen.	Die Beschleunigung wird durch Dynamikmodule umgesetzt.	Negative Abweichung erlaubt, um das Positionsziel zu erreichen. Positive Abweichung bis zu 0.2 m/s^2,wenn geforderte Geschwindigkeit nicht um mehr als 0,25 m/s überschritten wird.	Verzögerung nicht wie gewünscht durchgeführt.	Die Bremsen sind zu heiß, um ein ausreichendes Bremsmoment zu erzeugen.
Mögliche Abonnenten	Fahrdynamik- und Trajektorienregler	n/a	Andere Module innerhalb des Systems.	n/b	n/b
Relevanzmetrik	Metrik zur Bewertung, ob Trajektorie ohne Stabilitätsprobleme ausführbar ist.	n/a	n/b	n/b	n/b

5.5 Fazit und Diskussion

Die detaillierte semantische Schnittstellenbeschreibung (S^2I^2) beschreibt Modulschnittstellen mit in vier Kategorien unterteilten Attributen zur unabhängigen Spezifikation und Testfallgenerierung von Modulen. Im Gegensatz zu bestehenden Schnittstellenbeschreibungen beschreibt S^2I^2 die Semantik im Detail, die erforderlich ist, um Modultests abzuleiten oder eine fehlerhafte Spezifikation zu vermeiden. Die Attribute der Semantik beschreiben, welche Werte oder Zustände an den Schnittstellen auftreten können. Anhand der Attribute der Kategorie Einflussfaktoren wird beschrieben, was diese Charakteristiken der Semantik verursachen. Die Attribute der Kategorie Auswirkungen beschreiben, wie die Charakteristiken auf andere Module des Systems, das System als Ganzes oder andere Systeme bzw. deren Module wirken. Es wird gezeigt, wie S^2I^2 auf eine Schnittstelle aus dem Bereich der Informationstechnik angewendet werden kann und wie es auf eine Schnittstelle aus dem Bereich der Mechanik nur bedingt anwendbar ist. Zusammenfassend lässt sich sagen, dass S^2I^2 bei folgenden Anwendungen von Nutzen ist und damit die in Kapitel 5.1 formulierten Ziele unterstützt. Eine Übersicht der Ziele mit Erläuterungen, wie diese mit S^2I^2 unterstützt werden liefern folgende Stichpunkte:

- Vermeiden von Irrtümern in der Spezifikationsphase von Modulen und deren Schnittstellen durch detaillierte Angaben der erwarteten Charakteristika an einer Schnittstelle. Während (die Entwickler der) Herausgeber (eng.: Publisher) beschreiben, welche Charakteristika sie garantieren und mit welchen potentiellen Abweichungen unter welchen Bedingungen zu rechnen ist, können Abonnenten (eng.: Subscriber) diese Angaben prüfen und bei Bedarf eine Anpassung der Schnittstelle fordern.
- Ermittlung spezifischer Modultestfälle, die Abhängigkeiten zwischen Modulen und die Verbindung zur Systemebene berücksichtigen durch detaillierte Angaben der erwarteten Charakteristika an einer Schnittstelle. Auf Basis dieser Angaben können Modultests entworfen werden, die die beschriebenen Charakteristika einer Schnittstelle und Bedingungen, in denen diese auftreten repräsentieren.
- Identifikation von Gefahren durch induktive Verfahren der Gefährdungs- und Risikoanalyse. Die Attribute der Einflussfaktoren und Auswirkungen können dabei, wie beschrieben eine FMEA unterstützen. Darüber hinaus können die Attribute S^2I^2 als Beitrag für eine STPA dienen, die ebenfalls auf der Analyse von Schnittstellen basiert, sodass potentiell zusätzliche Gefährdungen aufgedeckt werden.
- Definition und Validierung von Simulationsmodellen auf Basis der in S^2I^2 beschriebenen Charakteristika der Schnittstelle. Die Validierung muss jedoch mit Hilfe von Validierungstechniken erfolgen, wie bspw. von Viehof[377] beschrieben.

[377] Viehof, M.: Dissertation, Objektive Qualitätsbewertung durch statistische Validierung (2018).

- Diagnose während der Laufzeit bzw. die Spezifikation eines Qualitätsvektors für die zugehörige Schnittstelle anhand der angegebenen Charakteristika durch die Attribute der Semantik. Während der Laufzeit kann überwacht werden, ob diese Charakteristika weiterhin erfüllt werden. Im Beispiel zur Beschreibung einer Trajektorie mit S^2I^2 nach Kapitel 5.4 wird erläutert, wie die Angabe der Irregularitäten und deren Ursachen zur Identifikation von Irrtümern dienen. Zusätzlich kann die Angabe der regulären Variabilität genutzt werden, um z. B. eine aggregierte Genauigkeit oder Aktualität für das System zu ermitteln. Anhand der Beschreibung der Einflussfaktoren und Auswirkungen lässt sich auch eine Funktion aufbauen, die Einflüsse auf und Auswirkungen durch die Semantik überwacht. Hierzu müssen die nach S^2I^2 beschriebenen Einflüsse und Auswirkungen in anderen Modulen oder auf Systemebene aufgenommen und mit dem Verhalten an der betreffenden Schnittstelle abgeglichen werden. Driessen[378] zeigt, wie z. B. Vor-/Nachbedingungen, Zeit- oder Wertebereichfähigkeiten als Vertrag in Software-Code definiert werden können. Sein Ansatz hängt jedoch von fehlerfreien Spezifikationen ab. Mögliche Irrtümer während der Spezifikation werden nicht berücksichtigt. Im Forschungsprojekt UNICAR*agil* werden Schnittstellen einer dienstbasierten Architektur mit funktionalen Daten und Qualitätsdaten verwendet.[379] Die Herausforderung besteht jedoch weiterhin darin, das Sollverhalten an den Schnittstellen zu spezifizieren. Eine Betrachtung der Auswirkungen auf das Verhalten des Gesamtsystems ist dabei weiterhin erforderlich. S^2I^2 bietet hierzu ein strukturiertes Rahmenwerk.
- Bewertung der Gültigkeit bestehender Informationen und Datensätze nach Änderungen auf Basis notwendiger Änderungen der Spezifikation mit S^2I^2. Durch Abgleich der Attribute nach Änderungen kann eine Einschätzung erfolgen, ob diese Änderung Einfluss auf andere Module hat.

Die Anwendung von S^2I^2 auf zwei exemplarische Schnittstellen zeigt, dass es zu einer systemischen Kapselung von Modulen beiträgt, um deren individuelle Absicherung zu unterstützen. Die Ziele nach Kapitel 4 werden jeweils durch die Einführung von einzelnen Attributen adressiert. Die Ziele zur Beschreibung der Zusammenhänge innerhalb des modularen Systems werden insbesondere durch die Attribute der *Einflussfaktoren* und *Auswirkungen* in Zusammenhang mit den Attributen der *Semantik* erreicht. Das Ziel bekannte Ungewissheiten, Variabilität oder geplante Änderungen zu berücksichtigen, wird durch die Beschreibungsart *reguläre Variabilität* adressiert. Mögliche Auswirkungen von Ungewissheiten bzw. Anomalien werden durch eine Beschreibung von *Irregularitäten* berücksichtigt. Bekannte Gründe für solche Irregularitäten werden unter dem Beschreibungstyp *Ursachen für Irregularitäten* dokumentiert.

[378] Driessen, T.: Dissertation, Modularity by Design (2019), S. 173–183.

[379] Mokhtarian, A. et al.: Dynamic Service-oriented Software Architecture (2020).

Zur Beschreibung von Modulschnittstellen mit S^2I^2 ist ein umfassendes Systemverständnis erforderlich, um bspw. die Abhängigkeiten zum beschriebenen Schnittstellenverhalten zu ermitteln. Die Beschreibung ist daher nicht allein von Modul- oder Schnittstellenentwicklern durchführbar. Systemarchitekten und andere Modulentwickler sind in den Prozess der Entwicklung von S^2I^2 einzubeziehen, um ein ausreichendes Systemwissen abzubilden. Hilfreich ist dabei auch eine vollständige Rückverfolgbarkeit von der Beschreibung von Attributen darzustellen. Das erforderliche Fachwissen und dementsprechend das Potential für neue Irrtümer nach Moduländerungen kann dadurch verringert werden. Es sollte also dokumentiert sein, warum Informationen relevante Bestandteile von S^2I^2 sind. Die Rückverfolgbarkeit hilft den Entwicklern einer Schnittstelle, den Einfluss von Änderungen in der Implementierung zu bewerten, ohne die Hilfe von Systemarchitekten oder anderen Modulentwicklern in Anspruch nehmen zu müssen.

Die Beschreibung hat im derzeitigen Stand jedoch die folgenden Einschränkungen in Bezug auf eine mögliche Erreichung der modularen Absicherung und erfüllt Teile der in Kapitel 5.1 formulierten Ziele noch nicht bzw. nicht vollständig:

- Die Verwendung von S^2I^2 erfordert einen hohen Aufwand zur Ermittlung der Beschreibungen der dargestellten Attribute. Viele Attribute erfordern Analysemethoden, wie Sensitivitätsanalysen, Identifikation von Abhängigkeiten oder die Beschreibung möglicher Werteverläufe. Zusätzlich muss dieser Aufwand für alle Modulschnittstellen des betrachteten Systems betrieben werden. Die Softwarearchitektur von UNICAR*agil* enthält bereits mehr als 100 verschiedene Modulschnittstellen, die für das automatisierte Fahren benötigt werden. Im Folgenden werden daher zwei Möglichkeiten zur Aufwandsreduktion skizziert. Zum einen können nur Schnittstellen berücksichtigt werden, die für das sicherheitsrelevante Verhalten der Module relevant sind. Bisher fehlen hierzu jedoch systematische Ansätze, sodass eine Klassifizierung relevanter Schnittstellen auf Basis von Expertenwissen erfolgen muss. Zweitens sollten Vorlagen auf der Basis bisheriger Entwicklungen oder bestehender Literatur bereitgestellt werden. Zum Beispiel sollte die Beschreibung möglicher Verläufe auf grundlegenden Beschreibungen aus der Literatur im Bereich der Signalverarbeitung basieren. Die Schnittstellenbeschreibungen sollten zusätzlich in einen rekursiven Lernzyklus überführt werden, um den Aufwand in zukünftigen Generationen zu verringern. Darüber hinaus wird erwartet, dass bei den ersten Anwendungen keine hinreichend vollständigen Schnittstellenbeschreibungen erreichbar sind. Potential zur Aufwandsvermeidung besteht außerdem in der Kombination mit Risikoanalysen, wie FMEA oder STPA. Diese erfordern, wie erwähnt bspw. ebenfalls eine Strukturanalyse zur Identifikation von Einflussfaktoren bzw. Ursachen und Auswirkungen.
- Die Attribute von S^2I^2 sind möglicherweise nicht für alle Schnittstellen bzw. die Absicherung der entsprechenden Module relevant. Es würden also Informationen bereitgestellt, die nicht weiter genutzt werden. Die Entwicklung eines Ansatzes zur systematischen Reduktion von Attributen, die für Module im Rahmen ihrer Absicherung nicht

relevant sind, ist eine offene Forschungsfrage. Bisher ist die Auswahl dieser Attribute auf Expertenwissen angewiesen.

- Für eine effektive Anwendung von S^2I^2 müssen unterschiedliche Granularitäten bzw. Perspektiven der Modulschnittstellen berücksichtigt werden. Neben der Analyse verschiedener Perspektiven durch Experten würde der Prozess auch hier von einer systematischen Vorgehensweise profitieren.
- Die Nutzung von S^2I^2 zur Erreichung einer modularen Absicherung wird voraussichtlich nicht ausreichen. S^2I^2 bietet einen grundlegenden Rahmen zur Definition von Modellen der Modulumgebung, ohne an eine spezifische Implementierung dieser Umgebung gebunden zu sein. Die von der spezifischen Umgebung erzeugten Stimuli sollten allerdings anhand von S^2I^2 verifiziert sein, um sicherzustellen, dass der damit spezifizierte Vertrag zwischen den Modulen erfüllt wird.
- Nach Änderungen an der Umgebung kann S^2I^2 ebenfalls Änderungen erfordern, sodass die Absicherung der Module nicht mehr sichergestellt ist. Bei Änderungen an der Umgebung ist daher eine Analyse der Auswirkungen auf die Module erforderlich, die die entsprechende Schnittstelle abonnieren. S^2I^2 bietet einen Rahmen für diese Überprüfungen. Vereinfachte Beschreibungen der Attribute können nach Änderungen eine größere Wirkung haben. Daher sind diese Vereinfachungen zu dokumentieren und nach Änderungen neu zu bewerten. Insgesamt bleibt die Einstufung der Relevanz von Informationen für S^2I^2 jedoch herausfordernd. Dies gilt insbesondere bei Einsatz von maschinellen Lernverfahren, die sensitiv bereits auf geringe Änderungen der Stimuli reagieren (vgl. Mori et al.[380]).
- Die Beschreibung von Einflüssen auf der Systemebene wird in S^2I^2 bisher in natürlicher Sprache beschrieben. Dies ist dem Ziel der Anwendbarkeit auf alle Varianten von Schnittstellen geschuldet. Nachteilig ist, dass die natürliche Sprache durch ihre hohe Flexibilität nicht immer eindeutig und verständlich ist. Darüber hinaus ist deren Automatisierung (bspw. zur Testautomatisierung) vergleichsweise aufwendig, da Sprachalgorithmen bisher noch hohe Rechenressourcen benötigen und unzuverlässiger arbeiten als bei Erstellung aus einer formaleren Sprache, wie bspw. die Unified Modelling Language (UML). Im Anforderungsmanagement wird die natürliche Sprache formalisiert, um Anforderungen zu strukturieren, sodass dies für S^2I^2 ebenfalls eine potentielle Erweiterung darstellt.[381] Kocerka et al.[382] geben einen Überblick über die dabei verwendeten Formalisierungsmethoden. Lippert et al.[383] führen bspw. als formalisierte Beschreibungsform die Behavior Semantic Scenery Description (BSSD) ein, die Elemente der Systemebene in

[380] Mori, K. T. et al.: The Inadequacy of Discrete Scenarios (2022).

[381] INCOSE: Guide for Writing Requirements (2017), S. 1–2.

[382] Kocerka, J. et al.: Analysing Quality of Textual Requirements (2018).

[383] Lippert, M. et al.: Behavior-Semantic Scenery Description (2024).

Informationen für einen Verhaltensplaner übersetzt, die für die Bestimmung eines sicheren Verhaltens relevant sind. Anhang B.2 demonstriert anhand eines kurzen Ausschnitts, wie S^2I^2 in Diagrammform zusammen mit Abhängigkeiten zwischen den Attributen dargestellt werden kann.

- Die Nachverfolgung von Änderungen anderer Module, die das Übertragungsverhalten von der System- auf die Modulebene beeinflussen, erfordert bisher eine manuelle Neubewertung der Einflüsse. Der Einsatz von Modellen kann dies ebenfalls unterstützen. Allerdings sollten anstelle von besonders realistischen Simulationsmodellen Modelle bevorzugt werden, die das Übertragungsverhalten nur so weit abbilden, dass die Stimuli, wie nach S^2I^2 beschrieben erzeugt werden. Andernfalls steigt die Abhängigkeit zwischen dem Modell und den realen Modulen und erfordert ggf. Änderungen an den Modellen nach Änderungen an den realen Modulen.
- Die Attribute von S^2I^2 basieren auf den in Kapitel 4 erlangten Erkenntnissen und Zielen für eine modulare Absicherung sowie der Nutzung bestehender Schnittstellenbeschreibungen der Literatur. Daher wird empfohlen, auch die Attribute von S^2I^2 durch Anwendung eines rekursiven Lernzyklus anzupassen oder zu ergänzen.
- Die beispielhafte Anwendung von S^2I^2 in dieser Arbeit auf nur eine Schnittstelle eines hochautomatisierten Fahrzeugs kann nicht als repräsentativ für alle anderen Schnittstellen angesehen werden. Die Trajektorie hängt nicht von so vielen Einflussfaktoren ab, wie z. B. die Umfeldwahrnehmung eines hochautomatisierten Fahrzeugs. Daher ist die Identifizierung relevanter Informationen, die Teil von S^2I^2 sein sollten, für den Trajektorienregler weniger anspruchsvoll als für die Umfeldwahrnehmung eines automatisierten Fahrzeugs. Während Methoden für die Absicherung hochautomatisierter Fahrzeuge noch nicht ausreichend entwickelt sind, ist auch S^2I^2 keine vollständige Lösung, sondern ein unterstützendes Element. Die Schnittstellenbeschreibung stärkt für den Fall, dass die gesamte Absicherung des Systems erreichbar ist, die Argumentation für eine modulare Absicherung.

6 Demonstration am Beispiel der modularen Architektur im Projekt UNICAR*agil*

Im folgenden Kapitel wird die Anwendung und Zweckmäßigkeit der Methoden, die in Kapitel 4 mit Hilfe der Argumentationskette abgeleitet wurden und die in Kapitel 5 vorgestellte semantische Schnittstellenbeschreibung demonstriert. Dazu wird die Spezifikation einer modularen Architektur sowie die Möglichkeit zur Ableitung von Modultests am Beispiel des Prototypenfahrzeugs des Forschungsprojekts UNICAR*agil* vorgestellt. Integrationstests dienen zur Aufdeckung verbliebener Fehlerzustände und Irrtümer nach den Modultests. Hierbei wird verdeutlicht, wie S^2I^2 diese Fehlerzustände und Irrtümer zu vermeiden hilft.

6.1 Beschreibung des Projekts und der Architektur

Das Forschungsprojekt UNICAR*agil* verfolgt das Ziel, die evolutionär geprägten Strukturen der Automobilindustrie zu überwinden und disruptive, modulare Systemarchitekturen für automatisierte Fahrzeuge zu entwickeln. Die Hard- und Softwarearchitektur des Projekts ist konsequent modular aufgebaut, um ein hohes Potential zur Aktualisierung und Erweiterbarkeit des Systems zu schaffen. Dies bietet die Chance, dass eine größere Vielfalt an möglichen Anwendungsfällen für die Fahrzeuge ohne einen entsprechend höheren Entwicklungsaufwand einhergeht. Außerdem ist das modulare Konzept die Basis zur Umsetzung des in dieser Arbeit verfolgten Ansatzes einer modularen Absicherung.[384]

6.1.1 Funktionale-, E/E- und Dienstarchitektur

Im Projekt werden drei verschiedene Architektursichten betrachtet: die funktionale Architektur, die E/E-Architektur und die Dienstarchitektur. Die grundlegende funktionale Architektur folgt der Darstellung nach Abbildung 6-1. Diese beschreibt die Funktionen auf Basis der eingesetzten Komponenten sowie Aufgaben und teilt sich auf oberster Betrachtungsebene in die Cloud, die Leitwarte und das Fahrzeug auf. Die Cloud sammelt und verarbeitet Informationen des Verkehrsgeschehens über eine gesamte Flotte an Fahrzeugen sowie der Infrastruktur. Die Leitwarte überwacht die einzelnen Fahrzeuge der UNICAR*agil* Flotte und kann bei Bedarf mit Insassen kommunizieren und das Fahrzeug steuern. Das A-Modell innerhalb des Fahrzeugs folgt mit den dargestellten Funktionen dem Modell nach Rasmussen[385]. Herausstechend ist dabei die Brücke zwischen Stabilisierungs- und Führungsebene mit einer Funktion, die ein sicheres Verhalten auch bei Degradation auf dieser

[384] Woopen, T. et al.: UNICARagil - Disruptive Modular Architectures (2018), pp. 3-6.

[385] Rasmussen, J.: Distinctions in Human Performance Models (1983).

niedrigeren Ebene sicherstellt. Diese Funktion wird im Folgenden als „Sicheres Anhalten“ bezeichnet.

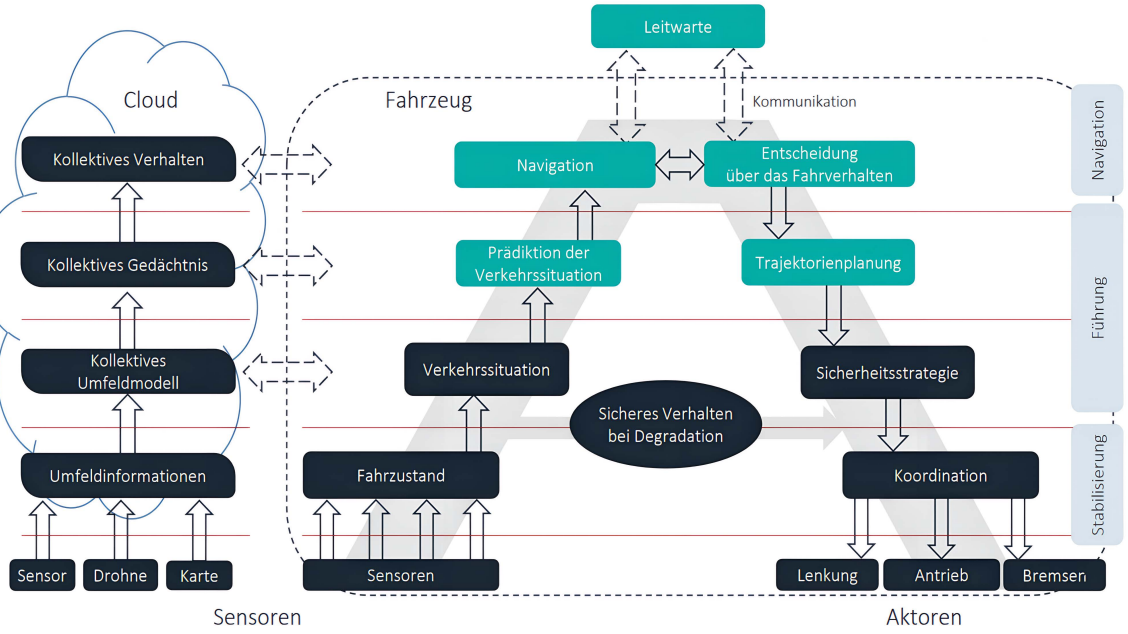


Abbildung 6-1: Funktionale Architektur von UNICAR*agil* in Form eines A-Modells.[386]

Die elektrische/elektronische Architektur (E/E-Architektur) der Fahrzeuge mit den Modulen, die vorwiegend für die automatisierte Fahrfunktion benötigt werden, ist in Abbildung 6-2 dargestellt. Die vier Sensormodule enthalten jeweils Kameras, Radare und Lidare zur Umfelderfassung. Sie sind an jeder Ecke des Fahrzeugs positioniert, um eine 360° Abdeckung der Umfeldwahrnehmung des Fahrzeugs zu ermöglichen. Im Großhirn werden die Informationen der Sensormodule fusioniert, um ein sicheres Verhalten des Ego-Fahrzeugs zur Zielerreichung zu ermitteln und daraus die Solltrajektorie sowie eine Notbahn zu berechnen. Während die Solltrajektorie für den normalen Betrieb bestimmt ist, wird die Notbahn immer parallel zur Verfügung gestellt. Hierdurch sollen Ausfälle oder Irrtümer in der Verarbeitungskette bis zur Berechnung der Trajektorie kompensiert werden. Wird ein Ausfall oder ein Irrtum erkannt, wird nur noch die zuletzt berechnete und gespeicherte Notbahn abgefahren. Die Funktion „Sicheres Anhalten“ nutzt währenddessen eine sekundäre Umfeldsensorik, um die Notbahn kontinuierlich auf mögliche Kollisionen zu prüfen. Auf Basis dieser Kollisionsprüfung wird eine Nottrajektorie berechnet. Das Stammhirn ist für die Umsetzung des beabsichtigten Fahrverhaltens verantwortlich, da es die Recheneinheit u. a. für die Funktion Sicheres Anhalten (SiA) sowie für den Fahrdynamik- und Trajektorienregler (FTR) ist. Die vier Dynamikmodule empfangen die vom FTR berechneten Stellgrößen. Sie beinhalten Antriebsmotor sowie Brems- und Lenkaktoren. Die Lenkaktoren sind in der

[386] Eckstein, L. et al.: UNICARagil news 4. Ausgabe (2021).

Lage, die Lenkwinkel der Räder auf bis zu 90° einzustellen, was dynamische Manöver wie z. B. Seitwärtsbewegungen ermöglicht.[387]

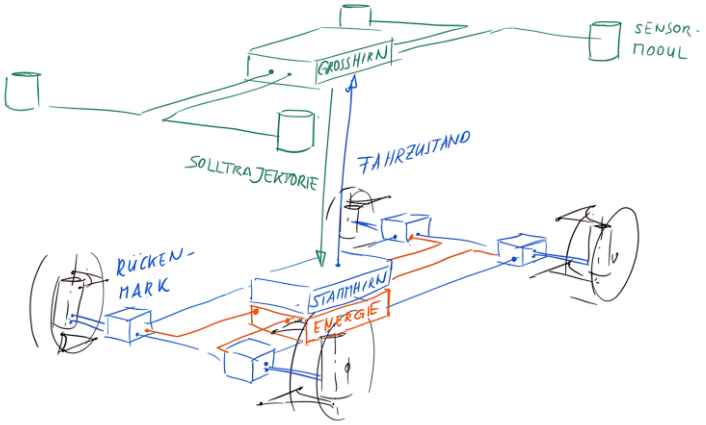


Abbildung 6-2: Skizze der E/E-Architektur des UNICAR*agil*-Fahrzeugkonzepts, das auf der Struktur des menschlichen Gehirns basiert.[388]

Die Modularität der Software der UNICAR*agil*-Fahrzeuge wird durch eine dienstorientierte Softwarearchitektur realisiert.[389] Dienste sind gekapselte Softwarekomponenten, die auf der Grundlage der funktionalen Architektur definiert werden.[390] Zusammen mit der automatisierten dienstorientierten Architektur ASOA können die Dienste während der Laufzeit gekoppelt werden. Daher sind verschiedene Verknüpfungen der Dienste innerhalb desselben Systems möglich, wodurch die Darstellung unterschiedlicher Funktionalitäten ermöglicht wird. Darüber hinaus können die Dienste aktualisiert, geändert oder neue Dienste implementiert werden, ohne dass Änderungen an den Schnittstellen anderer Dienste oder an den Hardwarekomponenten erforderlich sind.[391] Die Definition der Dienste erfolgt iterativ auf Basis, der im Projekt identifizierten, notwendigen Funktionen. Hierzu zählen neben den Diensten zum automatisierten Fahren auch Komfortfunktionen für die Insassen, wie z. B. die Innenraumklimatisierung.

6.1.2 Eignung der Architektur zur modularen Absicherung

Zur Einordnung, ob die Architektur von UNICAR*agil* grundsätzlich zur Durchführung einer modularen Absicherung geeignet ist, werden die Ziele der Argumentationskette aus Kapitel 4 herangezogen, die eine Bewertung bereits ohne detaillierte Informationen über die einzelnen Komponenten der Architektur zulassen. Hierzu zählen die Prinzipien zur Gestaltung

[387] Martens, T. et al.: UNICARagil Dynamics Module (2020).

[388] Woopen, T. et al.: UNICARagil news Juli 2020 (2020), S. 56.

[389] Mokhtarian, A. et al.: Dynamic Service-oriented Software Architecture (2020).

[390] Farcas, C. et al.: Addressing the Integration Challenge (2010), S. 564.

[391] Kampmann, A. et al.: Dynamic Service-Oriented Software Architecture (2019).

modularer Architekturen (vgl. die Beschreibung der Prinzipien in Kapitel 2.2), die Ziele zur Reduktion der Komplexität, zur Stimulation und Bewertbarkeit, sowie das neu eingeführte Ziel zur Erreichung semantischer Äquivalenz zwischen Architektursichten. Die Ziele zur Reduktion der Komplexität sind allerdings nicht direkt anhand der existierenden Architektur bewertbar. Die Reduktion der Komplexität wird stattdessen durch die in Kapitel 5 entwickelte Schnittstellenbeschreibung adressiert.

Die Prinzipien zur Gestaltung modularer Architekturen sind lediglich richtungsweisende Empfehlungen, sodass deren Einhaltung nur qualitativ geprüft werden kann. Aufgrund der im Mittelpunkt stehenden Herausforderung Dienste und ihre Funktionen eines hochautomatisierten Fahrzeugs abzusichern, werden die Gestaltungsprinzipien für Softwarearchitekturen betrachtet und deren Begrifflichkeiten verwendet. Die E/E-Architektur wird als grobgranularere Architektur lediglich zum Vergleich herangezogen.

Das Separation-of-Concerns Prinzip kann als erfüllt angesehen werden, da die einzelnen Dienste so zu gestalten sind, dass sie eigenständig programmiert, getestet und betrieben werden, sofern die spezifizierten Stimuli erzeugt werden. In der eingesetzten ASOA meldet sich jeder Dienst eigenständig als verfügbar und wird bei Bedarf aktiviert. Die Funktionsbeschreibung und die sich daraus ergebenden Anforderungen sind darüber hinaus direkt mit den Diensten verknüpft, da die Dienstarchitektur von der Funktionsarchitektur abgeleitet ist.

Das Prinzip der hohen Kohäsion ist in Bezug auf die E/E-Architektur erfüllt, da die Einteilung der Hardwaremodule auf der Einteilung der Fahraufgabe, wie sie ein Mensch verarbeitet, basiert. Die Sensormodule beinhalten bspw. die Komponenten zur Wahrnehmung, während die Dynamikmodule die Komponenten zur Umsetzung der Bewegung beinhalten. Die Einteilung der Dienste ist dagegen deutlich feingranularer und orientiert sich an den einzelnen Teilfunktionen der Fahraufgabe. Dies ist für den flexiblen Austausch und potentielle Updates beabsichtigt, allerdings erfüllt die Dienstarchitektur damit nur beschränkt das Prinzip der hohen Kohäsion. Zwar ist die Kohäsion innerhalb eines Dienstes höher als zu anderen Diensten, jedoch ist diese in der Dienstarchitektur von UNICAR*agil* diffus. Manche Dienste besitzen untereinander teils deutlich stärkere Abhängigkeiten als zu anderen Diensten.

Aufgrund des Prinzips der ASOA erreicht die Dienstarchitektur jedoch eine lose Kopplung zwischen den Diensten. Diese besitzen zwar unterschiedlich starke Abhängigkeiten zueinander, können jedoch relativ unabhängig voneinander betrieben werden.

Das Prinzip des Information Hiding wird ebenfalls durch die verwendete Struktur in ASOA erreicht. Dienste geben Informationen als Garantien in Form von Daten aus, die wiederum von anderen Diensten auf Basis von Anforderungen abonniert werden. Ein Informationsaustausch erfolgt daher nur über die abgestimmten und definierten Schnittstellen als Garantien und Anforderungen. Die Dienste erfahren untereinander nichts von den jeweiligen internen Abläufen.

Bewertung der semantischen Äquivalenz der Architektursichten

Die Dienstarchitektur ist direkt von der funktionalen Architektur nach Woopen et al.[392] und Buchholz et al.[393] abgeleitet. Dienste sind gekapselte Einheiten, die modulare Eigenschaften besitzen. Eine Teilfunktion wird dabei jedoch teilweise von mehr als einem Dienst erfüllt, sodass keine semantische Äquivalenz zwischen funktionaler und Dienstarchitektur erreicht wird. Im Projekt soll z. B. eine Funktion dargestellt werden, die eine Rückfallebene zur Perzeption und Planung darstellt. Diese Funktion wird in der funktionalen Architektur lediglich mit dem Begriff „Sicheres Anhalten“ beschrieben. Hierfür werden aber sowohl Dienste zur Verarbeitung der verschiedenen Sensordaten sowie zur Planung einer Trajektorie benötigt. Die funktionale Architektur und Dienstarchitektur sind daher nicht semantisch äquivalent zueinander.

Die E/E-Architektur von UNICAR*agil* nach Abbildung 6-2 folgt dem Gedanken einer Reduktion der Steuergeräte zu einer gegenüber heutigen Serienfahrzeugen geringen Anzahl an Steuergeräten oder Rechnern, die jeweils für eine Vielzahl unterschiedlicher Dienste als Rechen- und Speicherressource zur Verfügung stehen. Die E/E-Architektur folgt dabei nicht einer konsequenten Zentralisierung aller Rechenressourcen zu einem einzelnen Rechner, sondern strebt lediglich die Clusterung von Rechenressourcen zu Modulen an. Die Rechner besitzen jeweils trotzdem eine höhere Abhängigkeit zwischen den beinhalteten Rechen- und Speicherressourcen, die sich auf eine höhere Abhängigkeit zwischen den darauf laufenden Diensten überträgt. Trotz der konsequenten Modularisierung wird daher eine semantische Äquivalenz auch zwischen E/E- und Dienstarchitektur nicht erreicht. Weitere Maßnahmen, wie die Containerisierung von Softwarekomponenten und Partitionierung von Hardwareressourcen inklusive von Memory und Timing Protection Maßnahmen können dies, wie in Kapitel 4.6.1 erläutert, zukünftig unterstützen. Die verstärkte Zentralisierung von Steuergeräten erfordert auf diesen Gebieten jedoch weitere Forschung, um einer modularen Absicherung zu genügen, die auf Systemtests zur Absicherung der beschriebenen Maßnahmen verzichtet.[394, 395]

6.1.3 Definition der Modularchitektur

Die Architektursichten erfüllen, wie bereits erläutert, nicht durchgängig die Prinzipien zur Gestaltung modularer Architekturen sowie die semantische Äquivalenz zwischen Architektursichten. In Kapitel 4.6 wurde hergeleitet, dass Module in verschiedenen Architektursichten semantisch äquivalent zueinander sein müssen, um eine modulare Absicherung zu ermöglichen. Das folgende Kapitel definiert daher eine Modularchitektur, die diese

[392] Woopen, T. et al.: UNICARagil - Disruptive Modular Architectures (2018).

[393] Buchholz, M. et al.: Automation of the UNICARagil vehicles (2020).

[394] Hackelsperger, M. et al.: Master Controller für das softwaredefinierte Fahrzeug (2022).

[395] Bhange, A. et al.: Kernelemente einer SW-Architektur für Cross-Domänen-Rechner (2023).

Eigenschaften erfüllt. Dienste erfüllen zum Teil vergleichsweise einfache Funktionalitäten (z. B. für die Lichtsteuerung), oder es wurden noch keine Bewertungsmetriken für ihre Ergebnisse gefunden (z. B. eine direkte Bewertungsmetrik für die Punktwolken des Dienstes zur Vorverarbeitung von Rohdaten eines LIDARs). Mit dem Fokus dieser Arbeit auf automatisierte Fahrfunktionen ist die Modularchitektur im Vergleich zur Dienstarchitektur vereinfacht und grobgranularer definiert. Grobgranularer bedeutet an dieser Stelle, dass teils mehrere Dienste zu einem Modul zusammengefasst werden. Gründe hierfür können sein, dass Dienste eine gemeinsame Funktion darstellen, die die gleichen Hardwareressourcen nutzen oder die Ausgaben einer der Dienste nicht bewertbar sind. Mit der Weiterverarbeitung durch die inkludierten Dienste des betrachteten Moduls wird erst eine bewertbare Ausgabe erzeugt. Die Zusammenfassung von Diensten, die gleiche Hardwareressourcen nutzen, unterstützt darüber hinaus die Erreichung semantischer Äquivalenz zwischen Software- und Hardwarearchitektur. Die Modularchitektur ist jedoch feingranularer als die vorgestellte Hardwarearchitektur. Die Wahl der gleichen Grenzen wie die der Hardwarearchitektur hätte u. a. die folgenden Nachteile:

- Die Beschreibung eines Moduls kann unübersichtlich werden, wenn verschiedene Funktionen von Diensten, die derselben Hardwarekomponente zugeordnet sind, gemeinsam beschrieben werden.
- In Modultests müssen alle Dienste angesprochen werden, auch wenn ein Testfall nur für den Test einer einzelnen Funktion des Moduls vorgesehen ist. Dies erzeugt zusätzliche Kombinationsmöglichkeiten und dementsprechend höhere Testdatensätze.
- Änderungen an einer Funktion oder einem Dienst erfordern die erneute Absicherung aller Dienste des Moduls. Für Funktionen, die möglicherweise gar nicht von der Änderung betroffen sind, bedeutet dies einen erhöhten Testaufwand gegenüber der Einordung in einem eigenen Modul.

Aufgrund dieser Nachteile wird auf die Einhaltung der Hardwaregrenzen zur Definition der Modularchitektur verzichtet. Die semantische Äquivalenz zwischen Soft- und Hardwarearchitektur kann stattdessen bspw. durch Kapselung der Rechenressourcen erreicht werden. Die verwendete ASOA-Middleware adressiert dies durch variable Verteilung der Rechenressourcen. ASOA unterstützt dabei verschiedene Computerplattformen und ermöglicht die Kommunikation zwischen Diensten, unabhängig davon, welche Hardwarekomponenten ihnen zugewiesen sind.[396] Die Hardwareressourcen werden von den jeweiligen Betriebssystemen verwaltet.[397] Hierbei müssen für sicherheitsrelevante Funktionen immer ausreichend Rechenressourcen zur Verfügung stehen. Die Interaktionen zwischen Hardware und Software sind nicht Gegenstand dieser Arbeit, da sie sich auf die funktionalen Interaktionen von

[396] Kampmann, A. et al.: Dynamic Service-Oriented Software Architecture (2019).

[397] Baun, C.: Operating Systems (2020), S. 15–19.

Softwarekomponenten konzentriert. Die Hardware-Software-Interaktionen sind jedoch sicherheitsrelevant und folglich relevant für eine modulare Absicherung. Eine genauere Analyse wird aber z. B. bereits in der Norm ISO 26262[398] detailliert behandelt.

Im Folgenden werden zwei Maßnahmen aus dem Projekt UNICAR*agil* vorgestellt, bei dem einmal die Granularität der Modularchitektur verringert und einmal erhöht wird. Dies demonstriert, wie Modulgrenzen anzupassen sind, um den Prinzipien modularer Architekturen zu folgen, die Ziele einer modularen Absicherung und eine semantische Äquivalenz zwischen den Architektursichten zu erreichen.

Anpassung der Modulgrenzen zur Trennung unterschiedlicher Funktionen

Die Module in UNICAR*agil* werden zunächst durch die Hardwaregrenzen (vgl. Abbildung 6-2) definiert. Dies hat zur Folge, dass das Modul Stammhirn Dienste beinhaltet, die stark unterschiedliche Funktionen erfüllen. Außerdem wird sich die Funktion „Sicheres Anhalten" voraussichtlich häufiger als die anderen Funktionen ändern, da sie für neu aufgedeckte relevante Szenarien (u. a. sogenannte „Corner Cases"[399]) aktualisiert werden soll. Daher wird das Modul in weitere Module zerlegt, sodass diese jeweils geringere Funktionsumfänge enthalten. Die zusätzliche Abhängigkeit zwischen den kleineren Modulen über die gemeinsam genutzten Hardwareressourcen erfordert zusätzliche Analysen und Tests.

Anpassung der Modulgrenzen zur Bewertbarkeit

Mit der von Lippert et al.[400] entwickelten Behavior Semantic Scenery Description (BSSD) ist die Bewertung der Entscheidungen eines Verhaltensplaners möglich. Hierdurch ist direkt zu erkennen, ob ein Irrtum bei der Interpretation der Situation oder des gewählten Verhaltens begangen wird. Allerdings liefert der Verhaltensplaner keine Ausgaben, die mithilfe von BSSD bewertet werden können. Daher werden der Verhaltensplaner und der Trajektorienplaner zu dem Modul Verhaltens- und Trajektorienplaner (VTP) zusammengefasst. Die Trajektorie als Ausgabe dieses Moduls kann schließlich ausgewertet werden, z. B. durch eine subjektive Einschätzung des Verhaltens durch Experten oder durch die Verwendung von Kritikalitätsmetriken (vgl. z. B. Westhofen et al. [401]).

Modularchitektur von UNICARagil

Abschließend wird die in Abbildung 6-3 dargestellte Modularchitektur der UNICAR*agil* Prototypen vorgestellt. Die Module sind entweder dem Großhirn, dem Stammhirn oder ihrer eigenen individuellen Hardware, die nicht mit anderen Modulen geteilt wird, zugeordnet. Für das Modul „Sicheres Anhalten" sind zusätzlich zwei darin enthaltene Dienste dargestellt. Für diese Dienste wäre auch eine individuelle Absicherung vorstellbar, d.h. deren Einstufung

[398] ISO: ISO 26262 - Road vehicles – Functional safety (2018), parts 4-6.

[399] ISO: ISO/TR 4804 - Safety and cybersecurity for automated driving systems (2020), S. 2.

[400] Lippert, M. et al.: Behavior-Semantic Scenery Description (2024).

[401] Westhofen, L. et al.: Criticality Metrics for Automated Driving (2022).

als eigenständige Module. Zur Reduktion der Vielfalt verschiedener Testaufbauten wird dies jedoch nicht weiterverfolgt. Umgekehrt könnte die sekundäre Umfeldsensorik auch Teil des Moduls „Sicheres Anhalten“ sein. Dies erfordert jedoch die Berücksichtigung der realen Sensoren, da validierte Sensormodelle für die Simulation nicht verfügbar sind. Die Modultests des Moduls „Sicheres Anhalten“ hingen dann stärker von einem funktionsfähigen Fahrzeug oder einer Vehicle-in-the-Loop Testumgebung ab.

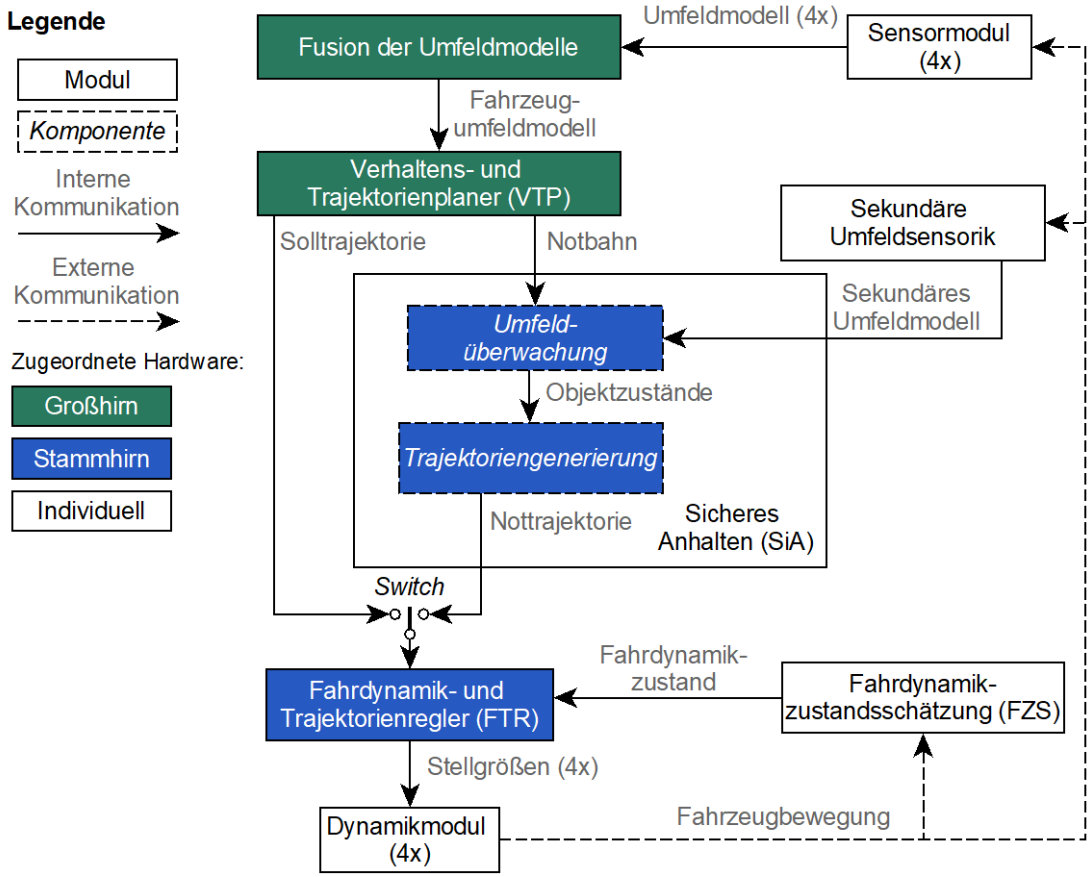


Abbildung 6-3: Modularchitektur der UNICAR*agil*-Prototypen mit Modulen, die Hauptfunktionen zum automatisierten Fahren beinhalten. Die Zahlen in Klammern geben die Anzahl einer Entität an.

Diese Arbeit verfolgt nicht das Ziel, neue Konzepte im Architekturdesign oder eine modulare Architektur für automatisierte Fahrzeuge zu entwickeln. Daher wird die betrachtete Modularchitektur nur auf Basis der beschriebenen Ziele und Prinzipien definiert, jedoch nicht darauf verifiziert. Hierbei erfolgen auch keine weiteren Iterationsschleifen, da die Architektur der Fahrzeuge bereits zu Projektbeginn eingefroren wird. Darüber hinaus stammen die Dienste und ihre Schnittstellen teilweise aus Vorentwicklungen anderer Forschungsprojekte, sodass diese nicht veränderbar sind. Stattdessen werden Beispiele aus den Modultests herangezogen, um verbleibende Defizite der UNICAR*agil* Architektur zu veranschaulichen. Aufgrund der begrenzten Anzahl an Testfällen dient dieses Vorgehen nur als erste Einschätzung darüber, ob die Architektur von UNICAR*agil* für eine modulare Absicherung geeignet ist.

6.2 Testprozess

Der im Folgenden beschriebene Testprozess dient im Forschungsprojekt UNICAR*agil* zur Absicherung der Module und des integrierten Gesamtsystems zur Durchführung der Demonstration der Fahrzeuge unter Abwesenheit eines unangemessenen Risikos. In der vorliegenden Arbeit wird der Testprozess zur Demonstration der Gültigkeit, der in Kapitel 4 und Kapitel 5 entwickelten Ansätze für eine modulare Absicherung genutzt. Es wird erläutert, dass die in Kapitel 4 hergeleitete Argumentationskette keine offensichtlichen Lücken aufweist. Darüber hinaus wird analysiert, ob die im Testprozess aufgedeckten Fehlerzustände und Irrtümer mit der in Kapitel 5 entwickelten detaillierten semantischen Schnittstellenbeschreibung S^2I^2 vermeidbar oder bereits auf Modulebene aufdeckbar sind.

6.2.1 Testmethodik

Die Ergebnisse der bisherigen Arbeiten zeigen, dass zum Verzicht auf Integrationstests Ungewissheiten im Entwicklungs- und Testprozess von Modulen aufgedeckt werden müssen. In der Industrie sind diese Prozesse gut etabliert und werden von einer größeren Anzahl an Ingenieuren durchgeführt als bei dem in dieser Arbeit untersuchten Forschungsprojekt. Zur Aufdeckung von Irrtümern, die in den Entwicklungsprozessen der Industrie ebenso auftreten können, sollten die betrachteten Module so weit wie möglich auf modularer Ebene entwickelt und getestet werden. Daher werden die Testfälle der betrachteten Module mit den vorgestellten Ansätzen bestimmt und in modularen Testumgebungen durchgeführt.

6.2.2 Testumgebungen

Im Forschungsprojekt steht nur eine beschränkte Auswahl möglicher Testumgebungen zur Verfügung. Darüber hinaus sind die zu testenden Module erst zu einem späten Zeitpunkt im Projekt so weit testbar, dass sie bewertbare Testergebnisse liefern. Daher sollten die Varianten an Testumgebungen in ihrem Umfang und Erstellungsaufwand möglichst gering ausfallen. Trotzdem werden ausreichend valide Testumgebungen benötigt, um den Betrieb der Fahrzeuge möglichst risikoarm zu gestalten und eine hohe Aussagekraft der Ergebnisse für die Demonstration der Ansätze zur modularen Absicherung zu erlangen. Daher wird die in Kapitel 4.6.2 vorgestellte Methode zur Argumentation der Testumgebungsreduktion ausgehend von Systemtests im realen Fahrzeug beispielhaft angewendet.

Betrachtet wird der *Fahrdynamik- und Trajektorienregler (FTR)* als Module under Test (MuT). Tabelle 6-1 listet die angenommenen Repräsentationsgrade der Module auf, die Teil der jeweiligen Testumgebung sind. Die zugeordneten Zahlen und Farben finden sich in den folgenden Tabellen zur Beschreibung der Testumgebungen wieder.

Tabelle 6-1: Beschreibung der angenommenen Repräsentationsgrade der Module.

0	Nicht implementiert
1	Nicht validiertes Modell (Closed Loop)
2	Injizierte validierte oder reale Daten (Open Loop)
3	Validiertes Modell (Closed Loop)
4	Reale Softwaremodule in simulierter Umgebung
5	Reale Softwaremodule auf realer Hardware

Das Beispiel nach Tabelle 6-2 beinhaltet auf der obersten Ebene das gesamte System sowie seine Umgebung. Zur Vereinfachung werden in den Spalten lediglich die als relevant eingestuften Module für das MuT, sowie zwei bekannte Einflüsse aus der Umgebung aufgenommen. Jede Zeile stellt eine mögliche Testumgebung dar. Anhand der Argumente lässt sich ableiten, ob die Testumgebung benötigt wird oder die Argumente bereits ohne weitere Tests in der jeweiligen Testumgebung ausreichen, um das Risiko des Einsatzes des MuTs im System ausreichend gering zu halten.

Tabelle 6-2: Beispielhafte Argumentation eingesetzter Testumgebungen für den FTR in UNICARagil. Repräsentationsgrade der Module nach

Testumgebung Nr.	**Angenommene Eigenschaften**	**Argumentation für getroffene Annahmen**	**Testziel**	FZS	Sensormodule	VTP	MuT: FTR	Dynamikmodule	Bordnetz	Kommunikationsnetz	Systemumgebung
1	Vollständiges reales Fahrzeug		Sicherheitsargumentation ist erbracht; Simulationsmodelle sind validiert	5	5	5	5	5	5	5	5
2	Alle relevanten Komponenten und Parameter inkludiert.	Einfluss der Charakteristiken der Sensormodule auf die VTP sind bestimmt, sodass darauffolgende Ausgaben des VTP bekannt sind.	Sicherheitsargumentation ist erbracht	5	0	5	5	5	5	5	5
3	Mögliche Trajektorien sind klar spezifiziert.	Mögliche Trajektorien ändern sich nicht für unterschiedliche Umfeldwahrnehmungen, wenn die Trajektorien klar spezifiziert sind.	MuT verhält sich robust ggü. elektrischen Schwankungen und deren Einfluss auf das Verhalten des Stammhirns.	5	0	2	5	5	5	5	5
4	MuT ist robust ggü. elektrischen Schwankungen	Die Hardware ist ASIL-D-zertifiziert, sodass keine größeren Abweichungen der getesteten elektrischen Schwankungen erwartet werden.	Validierung des Kommunikationsnetzes in der virtuellen Umgebung.	5	0	0	5	5	0	5	5
5	Modell des Kommunikationsnetzes validiert.	Das MuT wird durch Modelle mit validierten Schnittstellen stimuliert.	Validierung der Simulationsmodelle der Dynamikmodule.	5	0	0	5	5	0	1	5

Testumgebung Nr.	Angenommene Eigenschaften	Argumentation für getroffene Annahmen	Testziel	FZS	Sensormodule	VTP	MuT: FTR	Dynamikmodule	Bordnetz	Kommunikationsnetz	Systemumgebung
6	Modelle der Dynamikmodule sind validiert.	Die Simulationsmodelle der Dynamikmodule sind auf einem Prüfstand und auf Systemebene validiert.	MuT verhält sich robust ggü. des realen Verhaltens der Lokalisierung (FZS).	5	0	0	5	3	0	1	5
7	MuT ist robust ggü. des realen Verhaltens der Lokalisierung (FZS).	MuT wurde mit einer Vielzahl von Lokalisierungsdaten gemäß Spezifikation und mit aufgezeichneten realen Daten getestet.	Validierung von seltenen Umgebungseinflüssen (auf Basis des Simulationsmodells von IPG Carmaker)	1	0	0	5	3	0	1	5
8	Das Modul ist robust gegenüber schwankenden übertragbaren Kräften.	Das Modul wird auf einer breiten Variation von Reibwerten und Straßenprofilen getestet. Modelle, die Reibwerte und Straßenprofile liefern, sind durch Realdaten validiert (durch den Anbieter der Simulationssoftware).	Zeige die Echtzeitfähigkeit des MuTs an.	1	0	0	5	3	0	1	1
9	Echtzeitfähigkeit für dedizierte Hardware nachgewiesen.	Die Echtzeitfähigkeit des Moduls auf seiner dedizierten Hardware wird in der vorherigen Testumgebung bestätigt.	Absicherung der Funktionen des MuTs in ausreichend valider Testumgebung.	1	0	0	4	3	0	1	1

Neben den Testumgebungen für Modultests des *Fahrdynamik- und Trajektorienreglers (FTR)* werden Integrationstests durchgeführt, um verbleibende Fehlerzustände und Irrtümer aufzudecken. Tabelle 6-3 zeigt eine Übersicht der hierfür eingesetzten Testumgebungen. Die unterste Testumgebung H stellt dabei einen minimalen Aufbau dar, um einen geschlossenen Regelkreis (eng.: closed loop) für die *FTR* darzustellen. Testumgebung G schließt das Steuergerät mit ein, auf dem die *FTR* im Fahrzeug betrieben wird. In Testumgebung F wird das Modul *Sicheres Anhalten (SiA)* inkludiert, das Trajektorien auf Basis vordefinierter Notbahnen und Objekten in der simulierten Umgebung erzeugt. Hierbei wird die Testumgebung lediglich beschrieben und zu Modul- oder Integrationstests eingestuft. Testumgebung E beinhaltet statt *SiA* die *Verhaltens- und Trajektorienplanung* (VTP) zur Erzeugung von Trajektorien auf Basis einer Route und Objekten der simulierten Umgebung. Aufgrund nicht verfügbarer Sensormodelle existiert für die Simulation keine weitere Integrationsstufe, die den Regelkreis bis zu den Sensormodulen erweitert. Stattdessen wird mit Testumgebung D_a und D_b das reale Fahrzeug genutzt, um die Schnittstelle zwischen *FTR* und *Dynamikmodulen* zu verifizieren (D_b) sowie die zwischen menschlichem Fahrer und *Dynamikmodulen* anhand des Sidesticks und dem Bremspedal (D_a). Die Testumgebung D_a dient entsprechend nur zur Freigabe der Erprobung der Fahrzeuge. Äquivalent zur Testumgebung H werden in Testumgebung C vordefinierte Trajektorien abgefahren. Ebenso werden äquivalent zur Testumgebung F in Testumgebung B zuvor definierte Notbahnen abgefahren. In Testumgebung A

können äquivalent zur Testumgebung E vordefinierte Routen abgefahren werden. Allerdings ist im beschriebenen Testaufbau die Umfeldwahrnehmung noch nicht aktiv. Die Umfeldwahrnehmung ist Teil späterer Integrationsschritte, sodass diese zur Zeit der Aufnahmen für die Demonstration der modularen Absicherung nicht verfügbar war. Von dessen Integration werden allerdings keine grundlegend neuen Erkenntnisse für die vorliegende Arbeit erwartet, da das Zusammenspiel aus Umfeldwahrnehmung und Planung bereits in der Simulation und im Fahrzeug zumindest für das Modul *Sicheres Anhalten* dargestellt wird. Zur Wahrung des Verständnisses der Tabelle wird auf das gleiche Layout und die gleichen Einstufungen der Modulrepräsentation wie in Tabelle 6-2 zurückgegriffen. Gegenüber Tabelle 6-2 wird in Tabelle 6-3 zusätzlich das Modul *Sicheres Anhalten* sowie das zugehörige Modul *Sekundäre Umfeldsensorik* aufgeführt.

Tabelle 6-3: Eingesetzte Testumgebungen zur Aufdeckung verbleibender Fehlerzustände und Irrtümer nach Modultests. Repräsentationsgrade der Module nach Tabelle 6-1.

Testumgebung Nr.	**Beschreibung der Testumgebung**	**Modul-(M) / Integrationstest (I)**	FZS	Sensormodule	Fusion der Umfeldmodelle	VTP	Sekundäre Umfeldsensorik	Sicheres Anhalten	FTR	Dynamikmodule	Bordnetz	Kommunikationsnetz	Systemumgebung	Sidestick & Bremspedal
A	UNICAR*agil* Fahrzeug, das fähig ist eine vorgegebene Route abzufahren, ohne auf Objekte reagieren zu können.	I	5	0	0	5	0	0	5	5	5	5	5	0
B	UNICAR*agil* Fahrzeug, das fähig ist eine vorgegebene Notbahn abzufahren, und auf Objekte innerhalb dieser Bahn zu reagieren.	I	5	0	0	0	5	5	5	5	5	5	5	0
C	UNICAR*agil* Fahrzeug, das fähig ist eine vorgegebene Trajektorie abzufahren.	I	5	0	0	0	0	0	5	5	5	5	5	0
D_a	UNICAR*agil* Fahrzeug im manuellen Betrieb mit Sidestick und Bremspedal.	I	0	0	0	0	0	0	0	5	5	5	5	5
D_b	UNICAR*agil* Fahrzeug, in dem Dynamikmodule durch zuvor generierte Stellgrößen stimuliert werden.	I	0	0	0	0	0	0	0	5	5	5	5	0
E	Software in the Loop, in dem eine vorgegebene Route abgefahren werden kann.	I	1	0	0	4	0	0	4	1	0	0	1	0
F	Software in the Loop, in dem eine vorgegebene Notbahn abgefahren und auf Objekte innerhalb dieser Bahn reagiert werden kann.	I	1	0	0	0	1	4	4	1	0	0	1	0
G	Hardware in the Loop, in dem der FTR auf der dedizierten Hardware läuft und vorgegebene Trajektorien abfahren kann.	M	1	0	0	0	0	0	5	1	0	1	1	0
H	Software in the Loop, in dem der FTR vorgegebene Trajektorien abfahren kann.	M	1	0	0	0	0	0	4	1	0	1	1	0

6.2.3 Modultests

Die Modultests decken Fehlerzustände in der Implementierung und teilweise in der Spezifikation der Module auf. Die Modulentwickler des Projekts korrigieren daraufhin aufgedeckte Irrtümer und Fehlerzustände vor der weiteren Integration. Krause[402] stellt Methoden zur Generierung von Modultestfällen für das Dynamikmodul vor und leitet daraus Testfälle ab, die auf Modulebene mit Hilfe von Simulationsmodellen oder einem Hardware-in-the-Loop Prüfstand nach Testumgebung „F“ nach Tabelle 6-3 durchgeführt werden können. Modultestfälle und Testergebnisse für die Absicherung der Fahrdynamik- und Trajektorienregelung (FTR) werden von Blödel[403] in Form identifizierter Manöver auf Basis von Risikoanalysen beschrieben. Mit einer Analyse der spezifischen Implementierung des Reglers werden diese Testfälle zur Verifizierung der funktionalen Anforderungen an die FTR von Homolla[404] ergänzt und durchgeführt. Smits[405] leitet Anforderungen und Testszenarien zur Absicherung des Verhaltens- und Trajektorienplaners her. Smits identifiziert hierzu mögliche Abweichungen vom Sollverhalten, das in der verhaltenssemantischen Szeneriebeschreibung (BSSD) nach Lippert et al.[406] spezifiziert ist. Die Modultests des Verhaltens- und Trajektorienplaners werden in der Testumgebung „E“ nach Tabelle 6-3 und Tabelle 6-2 durchgeführt. Die Testumgebung beinhaltet die reale Implementierung des FTR, da die Generierung eines validen Modells des FTR zusätzlichen Aufwand erfordern würde. Die Solltrajektorien können zwar direkt aufgezeichnet und ausgewertet werden, es ergeben sich jedoch Einflüsse durch das spezifische Verhalten des FTR, die über das Fahrverhalten des Fahrzeugs an den VTP rückgekoppelt werden. Daher wird die Testumgebung „E“ als solche zur Durchführung von Integrationstests eingestuft.

Auf Basis der abgeleiteten Modultests von Krause[402], Blödel[403], Homolla[404] und Smits[405] lassen sich ergänzend zu den bereits in Kapitel 6.2.2 vorgestellten Limitationen der Architektur von UNICAR*agil* die folgenden Punkte identifizieren, welche die Erreichung der Ziele aus Kapitel 4 einschränken:

- Die hydraulischen Bremsen der Dynamikmodule sind nicht einzeln betätigbar.
    - Die Bremsdrücke von je zwei Rädern sind direkt gekoppelt. Dadurch sind auch die Einflussfaktoren auf das Bremsmoment miteinander gekoppelt und können nicht mehr unabhängig voneinander betrachtet werden. Die Regelung der Bremsmomente erfordert daher Wissen über diese Kopplung, die widerum fahrzeugspezifisch ist.

[402] Krause, J.: Bachelor Thesis, Tests für die Freigabe des Dynamikmoduls (2021).

[403] Blödel, A.: Master Thesis, Verfahren zur Generierung von Trajektorien (2021).

[404] Homolla, T.: Dissertation, Gekapselte Trajektorienfolgeregelung für autonomes Fahren (2023), S. 101–141.

[405] Smits, L.: Master Thesis, Testfälle für die Verhaltens- und Trajektorienplanung (2022).

[406] Lippert, M. et al.: Behavior-Semantic Scenery Description (2024).

- Valide Stimuli der Sensormodule zur Simulation der Fusion der Umfeldwahrnehmung sowie des Verhaltens- und Trajektorienplaners (VTP) sind nicht verfügbar.
- Eine Metrik zur Bewertung von Umgebungsmodellen, welche die Einflüsse auf die weitere Verarbeitung (z. B. im VTP) berücksichtigt, ist nicht verfügbar.
    - Der VTP kann sensitiv auf Veränderungen der Wahrnehmungsmerkmale reagieren und muss somit gemeinsam mit den Modulen für die Perzeption getestet werden.
    - Das Modul Sicheres Anhalten kann äquivalent sensitiv auf Veränderungen der sekundären Umfeldsensorik reagieren. Das Sichere Anhalten und die sekundäre Umfeldsensorik müssen daher gemeinsam getestet werden.
- Verhaltensplanung und Trajektorienplanung können aufgrund fehlender Schnittstellen nicht einzeln bewertet werden.
    - Die inidividuelle Bewertung der Verhaltensplanung würde direkt aufdecken, ob die Verkehrsregeln korrekt identifiziert und verstanden werden. Die Bewertung anhand der ausgegebenen Trajektorien erfordert dagegen zusätzliche Bewertungsmodelle und erzeugt Ungewissheit über die Kausalfaktoren des beobachteten Verhaltens. Nalic et al.[407] stellen hierzu bspw. eine Metrik vor, die eine gewählte Trajektorie mit Berücksichtigung möglicher Alternativen auf Basis kinematischer Randbedingungen bewertet. Mit jedem positiven oder negativen Testausgang ist jedoch unklar, ob das erzeugte Verhalten des Fahrzeugs auf Basis des korrekten oder inkorrekten Verständnisses der Situation erfolgt oder ob dieses durch die darauffolgende Erzeugung der Trajektorien noch mal verfälscht wird. Schöner und Antona-Makoshi[408] schlagen daher bspw. die Ausgabe interner Informationen über die jeweiligen Verhaltensentscheidung vor. Damit ist prüfbar, ob alle relevanten Informationen aufgenommen werden und zu welcher Verhaltensentscheidung diese führen. Die Autoren verwenden dabei den Begriff taktische Sicherheit, um ein vorausschauendes Verhalten zur Vermeidung potentiell gefährlicher Situationen zu beschreiben.
- Die Schnittstelle zum Dynamikmodul ist in den Prototypen von UNICAR*agil* für den manuellen und den automatisierten Fahrmodus unterschiedlich. Auch wenn diese keine Auswirkungen auf eine Absicherung der automatisierten Fahrfunktion hätte, ergeben sich Nachteile bei der Absicherung der Prototypen für die Erprobung und Demonstration. Allgemein ergeben sich durch die Verwendung gleicher Schnittstellen folgende Vorteile:
    - Gleiche Schnittstellen ermöglichen es, Stimuli für die Schnittstelle früher zu testen.
    - Die Verwendung gleicher Schnittstellen reduziert die Anzahl der Testfälle durch eine geringere mögliche Vielfalt an Stimuli.
    - Der Beschreibungsaufwand für Schnittstellen verringert sich durch weniger Variationen. Infolgedessen können die Beschreibungen der Schnittstellen höhere

[407] Nalic, D. et al.: Criticality Assessment Method for Automated Driving Systems (2023).

[408] Schöner, H.-P.; Antona-Makoshi, J.: Testing for Tactical Safety of Autonomous Vehicles (2021).

Robustheit erlangen, da zu deren Erstellung und Pflege mehr Ressourcen, wie bspw. Zeit und Personal, bereitgestellt werden können.

6.2.4 Integrationstests

Integrationstests werden als Tests definiert, in denen mehr als ein reales Modul verwendet wird. Ein Modul gilt als real, wenn es keine Änderungen gegenüber dem für den Betrieb des Systems vorgesehenen Modul besitzt, die das Modulverhalten signifikant ändern. Als signifikante Änderungen gelten für Integrationstests solche, die Einfluss auf das Verhalten der im betrachteten Integrationsstand beteiligten Module haben. Daher sind Änderungen an den Modulen zulässig, sofern ihr Einfluss bewertet und nicht signifikant ist.

Die in diesem Abschnitt vorgestellten Integrationstests, welche die Testumgebungen nach Tabelle 6-3 nutzen, folgen einem Bottom-up-Ansatz. Für die im realen Fahrzeug durchgeführten Integrationstests liegt der Schwerpunkt zunächst auf den Dynamikmodulen und ihren zugehörigen Steuergeräten (Testumgebungen D_a und D_b). Danach folgt die Integration des Fahrdynamik- und Trajektorienreglers (FTR), der die Dynamikmodule anspricht und durch vordefinierte Trajektorien stimuliert wird (Testumgebung C). Im letzten betrachteten Integrationsschritt wird der Verhaltens- und Trajektorienplaner (VTP) ergänzt, der durch eine vordefinierte Route stimuliert wird, jedoch noch kein Umfeldmodell erhält (Testumgebung A).

Das Hauptziel der Integrationstests besteht darin, verbleibende Fehlerzustände in und Irrtümer bei der Modulspezifikation und -implementierung aufzudecken, die erst bei der Interaktion mit der realen Umgebung zu beobachtbaren Fehlern führen. Die Modulspezifikation basiert auf der Systemarchitektur bzw. der Systemspezifikation. Daher sollen die Integrationstests auch Irrtümer bei der Systemspezifikation aufdecken.

Die Integrationstests konzentrieren sich auf Szenarien, in denen eine Interaktion der verschiedenen Module ausgelöst wird. Die Interaktionen werden durch eine breite Variation verschiedener Informationen und die Einbringung möglicher Störungen oder Irrtümer in die Kommunikation erreicht. Dennoch decken die Integrationstests auch Fehlerzustände in der Modulimplementierung auf, da im Rahmen des Forschungsprojekts nur wenige Module bereits auf Modulebene ausführlich getestet werden. Die Fehlerzustände und Irrtümer werden daher danach differenziert, ob diese in Modultests nach bestehenden Methoden aufdeckbar wären oder nicht.

Dynamikmodule

Aufgrund fehlender Prüfstandkapazitäten wird das Dynamikmodul nicht als Modul getestet. Daraus folgt für das Projekt die Notwendigkeit ausführlicherer Integrationstests. Auf Integrationsebene lassen sich trotzdem Fehlerzustände und Irrtümer ermitteln, die die Vorteile der Anwendung der in Kapitel 5 entwickelten semantischen Schnittstellenbeschreibung demonstrieren.

Als erstes wird hierzu die Testumgebung „D_a“ nach Tabelle 6-3 für Integrationstests der Dynamikmodule mit Verwendung der manuellen Steuerung via Sidestick und Bremspedal betrachtet. Die Stimulation erfolgt daher durch einen menschlichen Fahrer. Die manuelle Steuerung wird im Projekt genutzt, um die Prototypenfahrzeuge in der Werkstatt, für den Transport oder zu einer Ausgangsposition auf dem Testfeld für einen geplanten Testfall manövrieren zu können. Sie ist nicht Teil der Integration der automatisierten Fahrfunktion. Die manuelle Steuerung wird jedoch als erste Plattform der Integrationstests zur Identifizierung von Fehlerzuständen oder Irrtümern verwendet. In der Testumgebung „D_b“ werden die Eingaben dagegen automatisch eingespielt. Deren Erstellung erfolgt zuvor virtuell in der Simulation der FTR.

Testfälle, die versuchen ein sicherheitsrelevantes Versagen auszulösen, werden auf einer Hebebühne durchgeführt. Testfälle, die Einflüsse aus der realen Umgebung erfordern, werden auf einem Testgelände ausgeführt. Die Testfälle für die Dynamikmodule (die Beschreibung der Testfälle finden sich in Anhang C.1) sind anhand der folgenden Informationen abgeleitet[409]:

- Beschreibung der Dynamikmodule, des Sidesticks und Bremspedals sowie der internen Kontrolllogik durch natürliche Sprache, Datenarchitekturdiagrammen und Simulink-Modell der Steuerungslogik
- Anforderungen an die Dynamikmodule
- Anwendungsfälle und Fahrmanöver für manuelles Fahren
- Anwendungsfälle und Manöver für das automatisierte Fahren
- Abgeleitete Risiken auf Basis der Sicherheitsziele des Systems mit der Methode nach Klamann et al.[410] zur risikobasierten Testfallgenerierung auf Modulebene

Tabelle 6-4 fasst die beobachteten sicherheitsrelevanten Abweichungen vom Sollverhalten zusammen, die sich in den Integrationstests des Dynamikmoduls zeigen. Zusätzlich sind die aufgedeckten Fehlerzustände und mögliche Irrtümer, die während der Entwicklung potentiell zum jeweiligen Fehlerzustand geführt haben, angegeben. Die letzte Spalte verweist wiederum auf Attribute oder Kombinationen an Attributen von S^2I^2, die eine Vermeidung oder Entdeckung bereits auf Modulebene ermöglichen.

---

[409] Krause, J.: Bachelor Thesis, Tests für die Freigabe des Dynamikmoduls (2021), S. 41–43.

[410] Klamann, B. et al.: Pass-/Fail-Criteria for Particular Tests (2019).

Tabelle 6-4: Abweichung vom Sollverhalten bei Integrationstests des Dynamikmoduls beobachtet in den teilintegrierten Testumgebungen D_a und D_b nach Tabelle 6-3 im manuellen Fahrmodus (MD Abk. für „Manual Driving“). Ursachen sind in Form von Fehlerzuständen und den dazu zugeordneten Irrtümern in Spezifikation oder Implementierung angegeben. Zur Demonstration des Einsatzes der Attribute von S^2I^2 ist beschrieben, wie Attribute zur Vermeidung oder Entdeckung der Irrtümer beitragen. Die notwendigen Attribute sind in Fett hervorgehoben.

Test ID	**Abweichung vom Sollverhalten**	**Fehlerzustand in Implementierung**	**Potentielle Irrtümer bei Spezifikation oder Implementierung**	**Vermeidung oder Entdeckung mit S^2I^2**
MD 3-2	Fehlermodus leitet kein Bremsmanöver ein.	Bremsmanöver für Fehlermodus nicht implementiert.	Anforderung des Bremsmanövers nicht eindeutig an Entwickler kommuniziert.	Angabe und Verifikation des **Werte-/Informationsverlaufs** in verschiedenen **Modi.**
MD 3-3	Unterbrechung des Bremspedalsignals führt nicht zum Fehlermodus.	Keine Verbindung zu Fehlermodus vorhanden, für die Bedingung, dass fehlendes Bremspedalsignal erkannt ist.	Die Bedingungen für Fehlermodi sind nicht im Detail spezifiziert.	Spezifikation und Test von **Irregularitäten des Wertebereichs** in verschiedenen **Modi** in Simulation der Steuerungslogik.
MD 4-2	Die Räder kommen bei einer Beschleunigung auf 15 km/h und gleichzeitiger hydraulischer Bremsung nicht zum Stehen.	Anforderungen an Antriebsmoment werden trotz hydraulischer Bremsung umgesetzt. Antiblockiersystem reduziert das Bremsmoment aufgrund hoher Radbeschleunigung, wenn auf Hebebühne nur die Rotationsträgheit der Räder dem Bremsmoment entgegenwirkt.	Die Anforderung wurde nicht definiert, da das hydraulische Bremsmoment höher ist als das Antriebsmoment. Ein gleichzeitiges Eingreifen des Anti-Blockier-Systems wurde nicht bedacht.	Spezifikation von Wertebereichen des Einflussfaktors durch das **Verhalten anderer Module** (Massenträgheit) und deren Test in der Simulation.
MD 5-1	Fahrzeug kann nicht durch einmaliges Ziehen des Schalters der Feststellbremse angehalten werden.	Die ursprüngliche Implementierung der Feststellbremse öffnet die Bremse während der Fahrt, wenn sie nicht vom Fahrer gehalten wird.	Die Funktionalität der Feststellbremse wurde nicht vollständig spezifiziert / die Anforderung wurde nicht formuliert.	Spezifikation des Anwendungsfalls in Form von **Systemszenarien** und deren Ableitung zu Modulanforderungen. Test der abgeleiteten Systemszenarien in der Simulation der Steuerung der Feststellbremse.
MD 6-1	Keine Möglichkeit der Rekuperation als Sekundärbremse bei Überschreitung der Geschwindigkeitsgrenze.	Alle Drehmomentanforderungen werden ignoriert, wenn die Drehzahlgrenze überschritten wird, unabhängig von der angeforderten Drehmomentrichtung.	Anforderung zur Verwendung der Rekuperation als Sekundärbremse bei Implementierung der Drehzahlbegrenzung nicht definiert.	Spezifikation und Test von Degradationszuständen als Teil der **Modi** in Kombination mit den **Wertebereichen** der verschiedenen Parameter sowie des **erwarteten Systemverhaltens**.

Test ID	Abweichung vom Sollverhalten	Fehlerzustand in Implementierung	Potentielle Irrtümer bei Spezifikation oder Implementierung	Vermeidung oder Entdeckung mit S^2I^2
MD 7-1	Die Geschwindigkeitsbegrenzung wird bei hohen Beschleunigungsanforderungen überschritten.	Es ist weder eine automatische Bremsfunktion, zur Geschwindigkeitsbegrenzung implementiert, noch wird das Drehmoment schnell genug reduziert, um eine Geschwindigkeitsüberschreitung zu vermeiden.	Bestehende Anforderungen berücksichtigen nur die Situation nach einer Geschwindigkeitsüberschreitung, nicht die Vermeidung einer Geschwindigkeitsüberschreitung.	Spezifikation und Test inkonsistenter aber möglicher Informationen / Daten durch Angabe **reguläre Variabilität** und mögliche **Irregularitäten**.
MD 8-1	Die primäre Hydraulikbremse lässt sich nicht betätigen.	Fehlermodus wird eingenommen, wenn das angeforderte Bremsmoment verringert wird, während die Feststellbremse geöffnet wird. Der Fehlermodus tritt auf, da vermeintlich unplausible Sensordaten erkannt werden.	Gleichzeitige Zustandsänderungen der hydraulischen Bremse und der Feststellbremse werden nicht berücksichtigt.	Spezifikation von Zuständen und Werten unter Berücksichtigung möglicher Parameterkombinationen und deren Test z. B. mit einem t-weisen Ansatz.

Neben dem Spezialfall der Dynamikmodultests werden in Tabelle 6-5 Testergebnisse aus den restlichen Testumgebungen nach Tabelle 6-3 vorgestellt. Die Integrationstests decken Schwächen der vorherigen Modultests auf. Die reine Betrachtung von Integrationstests verschleiert jedoch systematische Schwächen der Modultests, die lediglich zufällig bestimmte Fehlerzustände und Irrtümer aufdecken. Daher werden im Folgenden auch Ergebnisse aus den reinen Modultests herangezogen, um den Einsatz von S^2I^2 zur Absicherung von Modulen zu demonstrieren. Die vollständige Beschreibung der Testfälle ist in Anhang C.2 aufgeführt.

Tabelle 6-5: Abweichung vom Sollverhalten der Fahrdynamik- und Trajektorienregelung sowie der Verhaltens- und Trajektorienplanung. Die Test ID gibt die verwendete Testumgebung als Buchstabe und die Nummer des Tests innerhalb dieser Testumgebung an. Zur Demonstration des Einsatzes der Attribute von S^2I^2 ist beschrieben, wie Attribute zur Vermeidung oder Entdeckung der Irrtümer beitragen. Die notwendigen Attribute sind in Fett hervorgehoben.

Test ID	Abweichung vom Sollverhalten	Fehlerzustand in Implementierung	Potentielle Irrtümer bei Spezifikation, Implementierung oder Modultest	Vermeidung oder Entdeckung mit S^2I^2
G1	Die Lenkwinkel der Räder bleiben im Stillstand in der vorherigen Position, obwohl bereits eine neue Richtung in der Trajektorie festgelegt ist. Die Räder werden daher beim Weiterfahren ruckartig eingeschlagen.	Die FTR berechnet einen Lenkwinkel von Null, wenn die Geschwindigkeit der Radaufstandsfläche gleich Null ist.	Obwohl die Schnittstelle korrekt angegeben wurde, fehlte eine Beschreibung der Interpretation der Werte (Richtung der Geschwindigkeit und Beschleunigung).	Spezifiziere erwartete **Auswirkungen auf Systemebene** auf Basis der **Äquivalenzklassen von Modulausgaben**. *Oder* Identifizere **Zustandsänderungen** und Beschreibung **erwarteter Auswirkungen**.

Test ID	Abweichung vom Sollverhalten	Fehlerzustand in Implementierung	Potentielle Irrtümer bei Spezifikation, Implementierung oder Modultest	Vermeidung oder Entdeckung mit S^2I^2
G2	Simulation stoppt plötzlich.	Trajektorie enthält einzelne unplausible Werte	Trajektorie wird nicht auf Plausibilität geprüft. *Oder* FTR ist nicht ausreichend robust gegen einzelne Unplausibilitäten.	Spezifiziere **mögliche unplausible/inkonsistente Werte** bzw. Anomalien, die während des Betriebs auftreten können.
G3	Am Ende des abklingenden Slaloms ändern sich die Lenkwinkel der Räder im Stillstand kontinuierlich, während der daraus resultierende seitliche Versatz nahezu Null ist.	Die Abklingfunktion für die Slalomamplitude geht nicht schnell genug gegen Null. Dies führt dazu, dass die Geschwindigkeit und die Beschleunigung gering sind, während sich ihre Richtung entsprechend ändert, was zu unnötig wechselnden Lenkwinkeln führt.	Spezifikation zu kleinen Werten bzw. an den Grenzen des Wertebereichs nicht vorhanden.	Spezifiziere die erwarteten **Auswirkungen** unter **verschiedenen Wertebereichen** und **Werteverläufen** und an deren **Grenzen**. *Oder* Identifiziere **Anwendungsfälle oder Szenarien** und ordne das **Modulverhalten** dabei zu.
E1	Bremsreaktion auf sich von hinten nähernde Objekte bis zum vollständigen Stillstand.	Nur Absolutabstand zu anderen Objekten berücksichtigt.	In frühen Entwicklungsständen wurden nur wenige einfache Szenarien berücksichtigt.	**Definiere Systemszenarien für Module** bereits zu Beginn der Entwicklung.
E2	Bei Einschermanöver eines dynamischen Objektes beschleunigt Egofahrzeug und überschreitet sogar das Geschwindigkeitslimit.	Szenarien, die zum beschriebenen Versagen führen, werden bei Berechnung der Zielgeschwindigkeit nicht berücksichtigt.	In frühen Entwicklungsständen wurden nur wenige einfache Szenarien berücksichtigt.	**Definiere Systemszenarien für Module** bereits zu Beginn der Entwicklung.
D1	Egofahrzeug fährt in verkehrte Richtung.	Richtungsvektoren für Beschleunigung, Geschwindigkeit und Ausrichtung beginnen immer bei null, statt bei aktueller wahrer Ausrichtung.	Keine oder fehlerhafte Schnittstellenverifikation durchgeführt. *Oder* Fehlende, fehlerhafte oder inkonsistente Schnittstellenbeschreibung.	Spezifiziere die Verknüpfung zwischen **Schnittstellenwerten und dem Systemverhalten.**
D2	Permanente etwa 2 m große laterale Regelabweichung von gewünschtem Fahrstreifen.	Nutzung unterschiedlicher Referenzsysteme zwischen Karte und Trajektorienplaner.	Transformationsmethode für Koordinatensystem wurde nicht spezifiziert.	Spezifiziere das **Referenzsystem** für die über eine Schnittstelle übertragenen Informationen und verifiziere dessen Implementierung in Tests.

Test ID	Abweichung vom Sollverhalten	Fehlerzustand in Implementierung	Potentielle Irrtümer bei Spezifikation, Implementierung oder Modultest	Vermeidung oder Entdeckung mit S^2I^2
D3	Lateraler Abstand zu Objekten bei Vorbeifahrt zu gering.	Objekt wird anderem Fahrstreifen zugewiesen, da die Karte den betreffenden Fahrstreifen innerhalb eines größeren Fahrstreifens enthält. Das Objekt wird dem größeren Fahrstreifen zugeordnet und angenommen, dass sich dieses in der Mitte dieses Fahrstreifens befindet, anstatt die ermittelte Position der Perzeption zu verwenden.	Karte wurde nicht auf Basis der Anforderungen der Prädiktion spezifiziert und aufgebaut.	Spezifiziere für welche **Funktionen und Anwendungsfälle Informationen verwendet** werden und wie dies das **Systemverhalten beeinflusst.**
C1	Permanente Abweichung vom gewünschten Gierwinkel nach Kurvenfahrt.	Invalide Modellannahmen in FTR.	Kombination einer Kurvenfahrt mit bis zu 180° Drehwinkel und darauffolgender Geradeausfahrt nicht in Modultests berücksichtigt.	Teste Kombinationen verschiedener Testfälle.
A1	Sprungartiger Wechsel des Kurswinkels nach 180° Wendung.	Nicht bekannt.	Evaluationskriterien für Kurswinkeländerungen nicht definiert.	Spezifiziere **Evaluationskriterien** für Informationen an einer Modulschnittstelle in verschiedenen Fahrmanövern.

6.3 Fazit & Einordnung der Testergebnisse

Dem Großteil der in Integrationstests aufgedeckten Fehlerzuständen und Irrtümern lassen sich Attribute der in Kapitel 5 eingeführten detaillierten semantischen Schnittstellenbeschreibung zuordnen, mit denen diese bereits durch eine genauere Modulspezifikation vermeidbar oder in Modultests aufdeckbar sind. Lediglich die Irrtümer bzw. Fehlerzustände die in den Tests MD 8-1 und C1 aufgedeckt wurden, wären nicht rein durch S^2I^2 vermeidbar oder aufdeckbar. Allerdings lassen sich diese wie angegeben bereits durch eine konsequente Durchführung der Testmethoden nach dem Stand der Technik aufdecken. Aufgrund des beschränkten Umfangs der Tests im Forschungsprojekt UNICAR*agil* ergeben sich allerdings die folgenden Limitationen bei der Unterstützung einer modularen Absicherung durch S^2I^2:

- Die durchgeführten Integrations- und Modultests sind nur eine Stichprobe zum Test eines frühen Prototyps in einem Forschungsprojekt. Daher können weiterhin Fehlerzustände und Irrtümer existieren, die nicht mit Hilfe von S^2I^2 auf Modulebene vermeid- oder aufdeckbar sind.

- Die Vorschläge zur Nutzung der Attribute erfolgen retrospektiv, sodass eine Voreingenommenheit ggü. notwendiger Spezifikationen und Tests herrscht. Die Attribute von S^2I^2 unterstützen Entwickler und Tester, durch Vorgabe zur Angabe bestimmter Informationen über eine Schnittstelle. Allerdings ist nicht sichergestellt, dass daraufhin auch die korrekten und vollständigen Angaben gemacht und die richtigen Schlüsse zur Vermeidung oder Aufdeckung von Irrtümern gezogen werden.
- Den beobachteten Abweichungen vom Sollverhalten lassen sich eindeutig den Fehlerzuständen zuordnen. Ebenso sind diese durch Elimination des Fehlerzustands in der Implementierung und anschließendem Regressionstest verifiziert. Die Irrtümer basieren dagegen teilweise auf Annahme zur Ursache der Entstehung des Fehlerzustands. Hierdurch können ebenfalls weiterhin Irrtümer existieren, die nicht mit Hilfe von S^2I^2 aufdeck- oder vermeidbar sind.

7 Fazit und Ausblick

Das finale Kapitel fasst die Ergebnisse dieser Arbeit zusammen und diskutiert mögliche Schwachstellen der Argumentationskette für eine modulare Absicherung, sowie der neu entwickelten Ansätze zur Erreichung der abgeleiteten Ziele. Die Arbeit schließt darauffolgend mit einem Ausblick ab, der Möglichkeiten zur Reduktion dieser Schwachstellen und die Potentiale der entwickelten Methoden im Rahmen einer modularen Absicherung als Ganzes aufzeigt.

7.1 Zusammenfassung und Diskussion

Das Ziel der vorliegenden Arbeit, Module individuell abzusichern und damit auf Integrationstests verzichten zu können, motiviert zwei wesentliche Forschungsfragen. Zuerst stellt sich die Frage, wie die Sicherheit von Modulen äquivalent zu einem System ohne Integrationstests argumentiert werden kann. Die zweite Forschungsfrage verfolgt die Entwicklung neuer Ansätze, um die Sicherheitsargumentation einer modularen Absicherung zu unterstützen. Zur Beantwortung der ersten Forschungsfrage wird auf Basis der Analyse des Stands der Technik und Wissenschaft eine Argumentationskette aufgebaut. Die darin aufgestellten Strategien, Ziele und Lösungen argumentieren, dass eine modulare Absicherung ein äquivalentes Sicherheitsniveau, wie die Absicherung auf Systemebene erreichen kann. Hierzu werden verschiedene Pfade mit Hilfe einer GSN-Argumentationskette verfolgt.

Lösungen in einer GSN-Argumentationskette kennzeichnen sich dadurch, dass sie das Erreichen eines Ziels nachweisen.[411] Hierbei wird selten ein mathematischer Beweis erbracht, sondern Methoden (z. B. die Durchführung einer Risikoanalyse) angeführt, die die Annahme, ein zu Ziel zu erreichen, argumentativ unterstützen. Lösungen können daher auch als Ziel angesehen werden, das nicht mehr weiter heruntergebrochen wird. Die abgeleiteten Lösungsansätze sind daher zwar konkrete Maßnahmen, die bei Durchführung zur Erfüllung der übergeordneten Ziele beitragen, können aber mit der Entwicklung detaillierterer Methoden weiter heruntergebrochen werden und dadurch neue Lösungsansätze motivieren oder deren Falsifizierung unterstützen. Strategien einer GSN unterliegen darüber hinaus auch Ungewissheiten, die in vorherigen Arbeiten keine Betrachtung finden. Die Analyse der modularen Absicherung bedarf allerdings der Aufdeckung bestehender Ungewissheiten vor den Integrationstests. Daher wird in dieser Arbeit ein zusätzliches Element zur Darstellung von Ungewissheiten in einer GSN-Argumentationskette eingeführt.

Der erste Pfad argumentiert, dass Module äquivalent zu einem System spezifizierbar sind, wenn die Modulumgebung wie eine Systemumgebung analysiert und beschrieben wird.

[411] SCSC: Goal Structuring Notation Community Standard (2021), S. 12.

Dazu bedarf es der Identifikation von Abhängigkeiten zwischen den Modulen. Die Ungewissheit über Abhängigkeiten kann durch Reduktion der Komplexität verringert werden. Daher wird der Stand der Technik zu Komplexitätseigenschaften für eine modulare Absicherung systematisch ergänzt. Die daraus resultierenden Ziele zur Reduktion der Komplexität stellen allerdings keine verifizierbaren Anforderungen dar, sondern geben lediglich Entwicklungsrichtungen vor. Dies gilt gleichermaßen für die bereits bestehenden Prinzipien zur Architekturgestaltung des Stands der Technik. Dadurch ergibt sich eine verbleibende Ungewissheit über die Vollständigkeit und die Beschreibung von Abhängigkeiten zwischen Modulen. Deren Grundlagen sind insbesondere Ungewissheiten über die korrekte Dekomposition der Module und der Definition ihrer Schnittstellen.

Der zweite Pfad argumentiert zusätzlich, dass die Modultests für die Absicherung vollständig und valide sind. Hieraus ergibt sich, dass neben einer validen Testumgebung insbesondere die Unvollständigkeit der Spezifikation dazu führt, dass Modultests lückenhaft sind. Daher werden eine Item-Definition zur Beschreibung automatisierter Fahrzeuge sowie bestehende Methoden zur Testfallgenerierung hinsichtlich notwendiger Informationen analysiert. Hierbei wird die Vereinfachung getroffen, dass Methoden einer Kategorie die gleichen Informationen benötigen. Zur gewünschten Vollständigkeit der Spezifikation und der Modultests verbleibt außerdem erneut die Ungewissheit über die korrekte Ableitung von Informationen von der System- auf die Modulebene aufgrund einer möglichen fehlerhaften oder unvollständigen Dekomposition.

In einem dritten Pfad wird folglich argumentiert, dass Ungewissheiten im Dekompositionsprozess vermeidbar sind. Hierzu erfolgt die Identifikation potentieller Irrtümer, die Definition von Zielen zu deren Vermeidung und die Zuordnung von Lösungen. Hierzu zählen bestehende Prinzipien zur Architekturgestaltung sowie zur Nachverfolgbarkeit aller Informationen, insbesondere zwischen System- und Modulebene. Neu identifiziert wird das Ziel, dass die Abbilder von Modulen in verschiedenen Architektursichten semantisch äquivalent sind. Die Umsetzung einer semantischen Äquivalenz zwischen Architektursichten bewirkt eine eindeutige Zuordnung der verschiedenen Informationen zu einem Modul.

Insgesamt fünf der acht abgeleiteten Ziele des dritten Pfades betreffen die Beschreibung von Modulschnittstellen bzw. von Abhängigkeiten zwischen Modulen. Die Modulschnittstellen sollten demnach einerseits die Testbarkeit der Module sicherstellen, andererseits Informationen über mögliche Abhängigkeiten und zu erwartendes Verhalten eines Moduls bereitstellen. Anhand dieser Informationen können Module, die ebenfalls Eigner der entsprechenden Schnittstelle sind, gestaltet und getestet werden, sodass möglichst keine Überraschungen aufgrund unbekannter Abhängigkeiten oder Verhaltensweisen auftreten.

Die Beschreibung von Modulschnittstellen wird daher als Schlüssellösung für eine modulare Absicherung identifiziert. Diese vereint die Vorteile einer Grey-Box Betrachtung von Modulen, sodass die Spezifikation nicht zu sehr von der Implementierung eines Moduls abhängt. Dies bietet mit der Möglichkeit zur Standardisierung das Potential einer evolutionären Entwicklung. Die Analyse existierender Schnittstellenbeschreibungen ergibt, dass sich diese darauf fokussieren, wie Informationen dargestellt, transportiert und gelesen werden können.

Bestehende Schnittstellenbeschreibungen definieren daher die Syntax im Detail, bieten aber keinen Rahmen oder Vorgaben zur Beschreibung der Semantik.

In der vorliegenden Arbeit wird daher eine neue Schnittstellenbeschreibung entwickelt, um die gesteckten Ziele zur Beschreibung der Abhängigkeiten und des Verhaltens zu erreichen. Die daraus entwickelte detaillierte semantische Schnittstellenbeschreibung S^2I^2 beinhaltet Attribute, die in vier Kategorien eingeteilt sind. Die Kategorie *Syntax* dient dem Verständnis, wie eine Nachricht strukturiert ist, sodass das sendende Modul eine Nachricht konstruieren und ein empfangendes Modul diese Nachricht rekonstruieren kann. In der Kategorie *Semantik* sind Attribute enthalten, um den Inhalt und die Charakteristika der ausgetauschten Informationen zu beschreiben. Attribute der Kategorie *Einflussfaktoren* beschreiben welche Faktoren die Semantik inwieweit beeinflussen. Die Kategorie *Auswirkungen* beinhaltet wiederum, wie die Semantik einer Schnittstelle andere Module oder das System beeinflusst bzw. welches Verhalten dadurch erwartet wird.

Die Schnittstellenbeschreibung S^2I^2 enthält darüber hinaus vier Beschreibungstypen. Diese geben neben idealen Sollwerten (*Ausdruck*), auch zulässige Toleranzen (*reguläre Variabilität*) sowie unzulässige aber mögliche Toleranzen (*Irregularitäten*) und deren Ursachen (*Ursachen für Irregularitäten*) von diesen Sollwerten an. S^2I^2 dient der Vermeidung von Irrtümern in der Spezifikation sowie als Informationsquelle für die Ableitung von Modultests, die potentielle Fehlerzustände aufdecken. Die Beschreibung der Modulschnittstellen bieten dabei den Vorteil unabhängig von der Implementierung eines Moduls sein zu können, sofern die Beschreibung nicht zu spezifisch ist.

Die Demonstration von S^2I^2 erfolgt zuerst an einem Anwendungsbeispiel aus dem Projekt UNICAR*agil*. Hierbei zeigt sich, dass die Beschreibung der einzelnen Attribute in natürlicher Sprache für eine Schnittstelle aus der Informationstechnik grundsätzlich möglich ist. Es zeigt sich allerdings, dass die Menge an Informationen unübersichtlich wird und einen hohen Erstellungsaufwand erfordert. Darüber hinaus ist das Beispiel zur Beschreibung einer Trajektorie aus Sicht der Komplexität zwar repräsentativ für einen Großteil der Schnittstellen eines Fahrzeugs, allerdings weniger umfangreich und eindeutiger beschreibbar als bspw. Schnittstellen der Perzeption. Es existieren hierfür jedoch bereits Arbeiten und erste Normen, die Schnittstellen von Sensoren für hochautomatisierte Fahrzeuge beschreiben.

Die weitere Demonstration der Nutzung von S^2I^2 erfolgt anhand von Modul- und Integrationstests in Simulation und am Prototypenfahrzeug des Projekts UNICAR*agil*. Anhand einer retrospektiven Betrachtung der identifizierten Fehlerzustände und Irrtümer wird gezeigt, dass die Beschreibung der Modulschnittstellen mit S^2I^2 Irrtümer direkt vermeidet oder als Informationsquelle zur Generierung von Modultests Fehlerzustände aufdeckt. Die Demonstration des Nutzens von S^2I^2 basiert damit auf einem real entwickelten hochautomatisierten Fahrzeug, wird allerdings auf einer gegenüber einer Serienentwicklung stark reduzierten Datenbasis durchgeführt. Der Anwendungsfall und Testumfang des Prototypenfahrzeugs beschränkt sich jedoch auf deren Nutzung für wenige Szenarien und eine Einsatzzeit von wenigen Wochen (inkl. der Erprobungszeit). S^2I^2 wird dabei nur retrospektiv evaluiert. Eine

Validierung der Beschreibung ist erst mit deren Anwendung in einem vollständigen Entwicklungsprozess und deren Bewertung z. B. basierend auf aufgedeckten Fehlerzuständen und Irrtümern im Systemtest trotz modularer Absicherung möglich. Daher steht die Validierung von S^2I^2 weiterhin aus. Im Rahmen des Forschungsprojekts wurden auch keine Änderungen an Modulen des Prototypenfahrzeugs durchgeführt. Daher ist bisher offen, in welchem Umfang notwendige Änderungen an Modulen auch Änderungen der Schnittstellenbeschreibung erfordern, sodass weiterhin alle Informationen zur Vermeidung und Aufdeckung von Fehlerzuständen und Irrtümern durch S^2I^2 bereitgestellt werden. S^2I^2 dient daher insbesondere der strukturierten Dokumentation und dazu das mentale Modell des betrachteten Systems aller beteiligten Entwickler und Tester zu stärken.

Allgemein beinhalten alle Argumentationspfade die Systemspezifikation inklusive der Modularchitektur als Grundlage zur Spezifikation und zum Test von Modulen. Während das Ziel der modularen Absicherung der Verzicht auf Integrations- und Systemtests ist, wird die Spezifikation des Systems bzw. der höchsten sicherheitsrelevanten Hierarchieebene als unerlässliche Informationsquelle identifiziert. Ohne diese lässt sich das Systemverhalten nicht ableiten, sodass eine Sicherheitsbewertung nicht möglich ist. Schlussendlich geht mit jeder neuen Technologie und jeder neuen Vorgehensweise ein inhärentes Risiko einher. Dies gilt für die Einführung hochautomatisierter Fahrzeuge ebenso wie für die Einführung einer modularen Absicherung. Im Falle einer modularen Absicherung lässt sich dieses Risiko jedoch durch eine stufenweise Reduktion von Integrationstests in Abhängigkeit von einem Maß zur Bewertung der Prozessreife eines modularen Absicherungskonzepts stark reduzieren. Der Ansatz einer evolutionären Entwicklung unter Einbezug einer detaillierten semantischen Beschreibung der Modulschnittstellen unterstützt maßgeblich die Konstanz dieser Prozessreife. Hierdurch scheint eine modulare Absicherung unter Einbezug der in dieser Arbeit entwickelten Argumentationskette innerhalb weniger Generationen möglich.

Anhand der vorliegenden Arbeit wird deutlich, dass die Absicherung auf Modulebene ähnliche Herausforderungen besitzt wie die Absicherung hochautomatisierter Fahrzeuge auf Systemebene. Die in der vorliegenden Arbeit aufgestellte Argumentationskette erläutert Ziele, die erreicht werden müssen, um diese Herausforderungen in einem Entwicklungsprozess zu adressieren. Hierbei zeigt sich, dass Methoden für die Systemebene zur Absicherung hochautomatisierter Fahrzeuge teils auf die Modulebene übertragbar sind, um die Ziele der Argumentationskette zu erreichen. Insbesondere Ungewissheiten im Dekompositionsprozess stellen sich als bisher unvollständig adressierte Herausforderung dar, die bei einer modularen Absicherung nicht durch Integrationstests auflösbar sind. Mit der vorliegenden Arbeit werden dabei potentiell auftretende Irrtümer systematisch aufgedeckt und daraus Ziele vorgestellt, die zur Vermeidung dieser Irrtümer beitragen. Anhand der Ziele wird gezeigt, dass diese durch Beschreibung des Modulverhaltens an ihren Schnittstellen erreichbar und damit die Irrtümer im Dekompositionsprozess vermeidbar sind. Die daraus folgend entwickelte detaillierte semantische Schnittstellenbeschreibung S^2I^2 unterstützt eine modulare Absicherung sowohl bei der Spezifikation der Modularchitektur und der einzelnen Module als auch bei der Entwicklung von Modultests. Gegenüber dem bisherigen Stand der Technik

und Wissenschaft werden hierdurch Abhängigkeiten zwischen den Modulen im Detail dokumentiert, ohne dabei an spezifische Eigenschaften der Modulimplementierung gebunden zu sein. Dies ermöglicht eine potentielle evolutionäre Entwicklung der Module und ihrer Schnittstellenbeschreibungen, sodass noch verbleibende bisher unbekannte Irrtümer in zukünftigen Generationen weiterhin relevant sind und entsprechende Vermeidungsmaßnahmen die Ungewissheiten einer modularen Absicherung kontinuierlich verringern.

7.2 Ausblick

Die initiale Motivation der vorliegenden Arbeit, den Absicherungsaufwand hochautomatisierter Fahrzeuge zu reduzieren, erfordert weiterhin enorme Forschungsarbeiten zur Entwicklung neuer Methoden für die Sicherheitsargumentation. Eine modulare Absicherung leistet hierzu einen wesentlichen Beitrag durch die Reduktion des Absicherungsaufwands nach Änderungen oder bei Einsatz gleicher Module in verschiedenen Fahrzeugmodellen. Die dazu aufgestellte Argumentationskette stellt allerdings keine Anleitung für eine modulare Absicherung dar. Zukünftige Forschungsarbeiten sollten diese auf Basis neuer Entwicklungen im Bereich der Absicherung automatisierter Fahrzeuge ergänzen und daraus einen Homologationsprozess für Module ableiten. Hierzu kann die Argumentationskette für eine modulare Absicherung als ein Teil parallel geschalteter Argumentationsketten, die das Forschungsprojekt VVM für verschiedene Stufen des Entwicklungsprozesses vorschlägt, genutzt werden.

Die vorliegende Arbeit legt anhand der Argumentationskette dar, dass Module wie ein System abgesichert werden können, sofern die dabei abgeleiteten Ziele erreicht werden. Zur Verifikation, inwieweit ein Ziel erreicht ist, fehlen insbesondere für die Gestaltung der Modularchitektur Metriken oder mindestens Schwellwerte, um z. B. zu bewerten, wie viele Abhängigkeiten in welcher Stärke zulässig für eine modulare Absicherung sind. Dies kann nur mit der vollständigen Durchführung einer modularen Absicherung und anschließender Validierung in Integrationstests erfolgen. Ein mögliches Maß zur retrospektiven Bewertung sind die verbleibenden aufgedeckten Fehlerzustände und Irrtümer in Integrationstests. Ähnlich den verbleibenden „Überraschungen“ pro Testkilometer eines hochautomatisierten Fahrzeugs kann für die Überraschungen pro Test in den Integrationstests ein Schwellenwert identifiziert und hinsichtlich Konvergenz untersucht werden. Dieser Schwellenwert kann z.B. die Anzahl der Überraschungen pro Test sein.

Die aufgebaute Argumentationskette beinhaltet weiterhin die Spezifikation des Systems inklusive der Modularchitektur als Teil der Informationsquellen zur Ableitung der Modulspezifikation bzw. der S^2I^2 Attributbeschreibungen. Zur Erreichung einer stärkeren Unabhängigkeit von einer individuellen Systemspezifikation ist die Definition einer standardisierten Spezifikation eines hochautomatisierten Fahrzeugs z. B. in Form einer Item Definition nach

dem Vorbild von Reschka[412] denkbar. Anhand der Schnittstellenbeschreibung mit S^2I^2 lässt sich auch retrospektiv auf die Systemebene zurückführen, welche Informationen notwendig sind und auf welche Informationen für eine Generalisierung der Systemspezifikation bzw. der verwendeten Modelle zur Darstellung des Systems verzichtet werden kann.

Die Attribute von S^2I^2 sind wie bestehende Schnittstellenbeschreibungen aus wissenschaftlichen Arbeiten und Normen wissensbasiert erstellt. Ebenso wird eine wissensbasierte Ermittlung der Attributbeschreibungen für eine konkrete Schnittstelle demonstriert. Hierbei existiert immer eine nicht quantifizierbare Ungewissheit über die Vollständigkeit der Attribute selbst sowie deren Beschreibung. Hierzu wird bereits die evolutionäre Entwicklung unter Verwendung von S^2I^2 als Optimierungsmaßnahme aufgezeigt. Zur Erhöhung der Vollständigkeit bietet sich außerdem die Möglichkeit mit dem in dieser Arbeit abgeleiteten Rahmen von S^2I^2 und der bestehenden Modularchitektur von UNICAR*agil* einen datenbasierten Ansatz zu verfolgen. Dieser hätte das Ziel die konkreten Beschreibungen der Modulschnittstellen mit S^2I^2, aber auch die notwendigen Attribute selbst zu ergänzen. Das Folgeprojekt von UNICAR*agil*, AUTOtech.*agil* widmet sich diesem Ansatz, indem die Modulschnittstellen in umfangreichen Integrationstests überwacht werden.[413] Dabei sollte insbesondere analysiert werden, welche Änderungen an der Implementierung eines Moduls zulässig sind, ohne dass sich die Attribute von S^2I^2 ändern. Hieraus können sich neue Anforderungen an die Schnittstellenbeschreibung, die Modularchitektur oder die Module selbst ergeben, da Änderungen der S^2I^2 Attribute die Absicherung der Module nicht mehr sicherstellen.

Die Übersichtlichkeit der Schnittstellenbeschreibung aufgrund des hohen Umfangs an Informationen kann durch eine Formalisierung der Beschreibung verbessert werden. Am Fachgebiet Fahrzeugtechnik der TU Darmstadt wurde u. a. von Kohls[414] demonstriert, wie die Beschreibung der Abhängigkeiten zwischen Modulen formalisierbar ist. Sie erweitert hierzu die Elemente der Unified Modeling Language (UML), um verschiedene Kategorien an Abhängigkeiten darzustellen. Sharvia und Papadopoulos[415] stellen darüber hinaus einen Ansatz vor, um das Verhalten eines Systems zu modellieren und dabei Kompositionen potentieller Fehlerzustände und Irrtümer zu berücksichtigen. S^2I^2 kann in beide Ansätze integriert werden, indem die modellierten Abhängigkeiten als Einflussfaktoren bzw. Auswirkungen betrachtet werden, welche die Semantik einer Modulschnittstelle beeinflussen. S^2I^2 kann außerdem an bestehende Schnittstellenbeschreibungen für Komponenten hochautomatisierter

[412] Reschka, A.: Dissertation, Fertigkeiten- und Fähigkeitengraphen von automatisierten Fahrzeugen (2017), S. 101–105.

[413] van Kempen, R. et al.: AUTOtech.agil: Architecture for Automotive Agility (2023), S. 36–37.

[414] Kohls, S. K.: Master Thesis, Abhängigkeiten zwischen Modulen (2022).

[415] Sharvia, S.; Papadopoulos, Y.: Compositional and Behavioural Safety Analysis (2011).

Fahrzeuge, wie bspw. die Definition von Sensorschnittstellen[416] anknüpfen, sodass diese unabhängig voneinander beschrieben werden. Hierbei ist insbesondere zur Genehmigung von Softwareupdates unter der Regulierung UNECE R156 die zukünftige Beschreibung eines Prozesses notwendig, der die Einflüsse der Updates analysiert. Der Abgleich der Attribute von S^2I^2 als Teil eines solchen Prozesses kann einen wesentlichen Beitrag leisten, um den dazu notwendigen Analyseaufwand möglichst gering zu halten.

Der Trend zur Zentralisierung von Steuergeräten wirkt der in der vorliegenden Arbeit geforderten semantischen Äquivalenz zwischen EE- und Dienstarchitektur entgegen. Für eine modulare Absicherung ist daher weiterer Forschungsbedarf bestehender Methoden zur konsequenten Kapselung sicherheitsrelevanter Module, die sich Rechenressourcen mit anderen Modulen teilen, erforderlich. Entwicklungen zur Nutzung von Software Containern oder der Partitionierung von Steuergeräten als virtuelle Steuergeräte erscheinen hierfür vielversprechend. Die damit versprochene Unabhängigkeit zwischen Softwarekomponenten erfordert allerdings ohne dessen Überprüfung in Systemtests weitergehende Untersuchungen.

Das Ziel eines Unternehmens sollte die Minimierung der Ungewissheit über die Qualität der generierten Entwicklungsergebnisse sein. Eine Möglichkeit hierzu ist die Vorgabe von Prozessen und Methoden. Die detaillierte semantische Schnittstellenbeschreibung stellt einen neuen Rahmen für einen Entwicklungsprozess dar. Der Prozess zur Definition von S^2I^2 wurde in dieser Arbeit demonstriert, sollte jedoch an die jeweiligen Prozesse eines Unternehmens adaptiert werden. Gleiches gilt für die Methode zur Identifikation von Irrtümern im Dekompositionsprozess. Zur Bewertung und potentiellen Optimierung dieser Prozesse haben sich in der Industrie Reifegradmodelle bewährt. A-SPICE oder TestSPICE stellen in der Automobilindustrie standardisierte Modelle dar. Die in der vorliegenden Arbeit entwickelten Methoden würden von einer Integration in diese Reifegradmodelle profitieren. Dies gilt insbesondere für die evolutionäre Entwicklung von Modulschnittstellen. Dies wird als Prozessinnovation auch im höchsten A-SPICE und TestSPICE Level 5 als Teil der Prozessbewertung berücksichtigt.[417, 418] Bestehende Prozessbewertungen versuchen allerdings grundsätzlich aus einer positiven Perspektive die Einhaltung von Prozessen und damit auch die Qualität und Sicherheit zu argumentieren. Aus Sicht des Autors der vorliegenden Arbeit sollte daher eine möglichst offene Risikokommunikation verfolgt werden. Diese bietet das Potential Schwachstellen im Prozess zu identifizieren, indem diese klar dokumentiert und bewertet werden. Hieraus lassen sich neue Lösungen zur Reduktion der benannten Ungewissheiten entwickeln. Die Ergänzungen von Ungewissheiten in einer GSN Argumentationskette der vorliegenden Arbeit haben gezeigt, dass sich hieraus neue Lösungen zur Reduktion der identifizierten Ungewissheiten entwickeln lassen.

---

[416] ISO: ISO 23150 - Data communication between sensors (2021).

[417] VDA QMC: Automotive SPICE (2017), S. 86–90.

[418] TestSPICE SIG: TestSPICE Process Assessment Model (2018), S. 152-154.

Eine offene Risikokommunikation ist nicht nur nach innen, sondern auch nach außen, d.h. gegenüber der Gesellschaft und den Regulierungsbehörden ein entscheidender Faktor, um eine zukünftige Einführung hochautomatisierter Fahrzeuge nachhaltig zu ermöglichen. Die Risiken müssen gegenüber dem Nutzen klar benannt und diskutiert werden, um einen Vertrauensbruch und damit einhergehende Restriktionen nach einem von einem HAF verschuldeten Unfall zu vermeiden.

In Bezug auf mögliche ökonomische Potentiale sollten folgende Arbeiten untersuchen, inwieweit die Einsparungen einer modularen Absicherung die Mehraufwände aufwiegen. Hierbei sollten die Einsparungen für die Wiederverwendung von Modulen in verschiedenen Modellvarianten und über mehrere Modellgenerationen beachtet werden. Gleichzeitig sollten mögliche Einschränkungen bei der weiteren Entwicklung aufgrund der Modularität bzw. der eingeschränkt änderbaren Schnittstellen analysiert werden. Es besteht darüber hinaus die Möglichkeit, dass durch individuell abgesicherte Module die Rolle eines Automobilherstellers (OEM) reduziert wird, da der Aufwand für die Integration und damit verbundene Absicherung sinkt. Nach Einschätzung des Autors dieser Arbeit ist die Entwicklung hinreichend unabhängiger Module durch einzelne Zulieferer jedoch begrenzt und mit hohen Investitionen verbunden. Darüber hinaus wird für die Spezifikation von S^2I^2 bereits aufgezeigt, dass weiterhin Wissen über die Abläufe im gesamten System erforderlich ist. Die Fähigkeiten zur Koordination einer solchen Spezifikation in Verbindung mit der Standardisierung von Modulschnittstellen befindet sich in der bestehenden Marktstruktur bei OEMs. Diese besitzen darüber hinaus die Ressourcen und den entsprechenden Einfluss auf eine Großzahl an Zulieferern. Ein OEM wird jedoch kein Interesse daran zeigen, ein Großteil der Wertschöpfung und damit der wirtschaftlichen Stärke zu verringern. Daher erwartet der Autor der vorliegenden Arbeit mit der Einführung einer modularen Absicherung keine grundlegende Umstrukturierung der Marktstruktur. Während OEMs voraussichtlich weiterhin die Koordination und Integration übernehmen, fällt bei modularer Absicherung ein größerer Teil der Homologation den Zulieferern zu. Dadurch bleibt die Verteilung der Verantwortlichkeit für das unvermeidbare inhärente Risiko bei Einführung hochautomatisierter Fahrzeuge mit einer modularen Absicherung bislang offen.

A Ungelöste Ziele der Argumentationskette für eine modulare Absicherung

Tabelle A-1 zeigt Ziele der Argumentationskette nach Kapitel 4, die auf Basis des Stands der Technik ungelöst sind. Ziele, die mit dem Inhalt von Schnittstellenbeschreibungen nach dem Stand der Technik bereits erreicht werden, bzw. Informationen der Notwendigkeit für die Absicherung bereits bekannt sind, werden an dieser Stelle nicht aufgelistet. Von den noch offenen Zielen wird erwartet, dass sie von einer detaillierten semantischen Schnittstellenbeschreibung unterstützt werden. Anhang B.1 beschreibt die Attribute von S^2I^2 und nennt dabei die mit dem jeweiligen Attribut adressierten Ziele der folgenden Tabelle.

Tabelle A-1: Ziele der detaillierten semantischen Schnittstellenbeschreibung auf Basis der Argumentationskette nach Kapitel 4.

ID	Ungelöste Ziele der GSN Argumentationskette
G4	Eigenschaften und Verhalten bleibt bei Austausch oder Änderungen von Modulen konstant, sodass emergente Systemeigenschaften bestehen bleiben.
G8	Die Spezifikation ist für einen erweiterten Gültigkeitsbereich für zukünftige Änderungen bzw. Variationen definiert.
G11	Die Stärke der Abhängigkeiten ist möglichst gering.
G12	Auswirkungen bei Änderungen einer Abhängigkeit sind analysiert und beschrieben.
G13	Der Typ einer Abhängigkeit ist beschrieben.
G15	Interdependenzen zwischen Modulen sind aufgedeckt und beschrieben.
G16	Das Verhalten der Module ist beschrieben.
G17	Potentielle Änderungen des Verhaltens sind beschrieben.
G25	Das Modul ist äquivalent zur Item Definition eines hochautomatisierten Fahrzeugs beschrieben.
G25a	Beschreibung des Anwendungsfalls.
G25b	Modulverhalten bei beschriebenen Anwendungsfällen.
G25c	Fehlerverhalten und Ursachen sind beschrieben.
G25d	Degradationsmodi und Ursachen sowie das dabei erwartete Modulverhalten sind beschrieben.
G25e	Das Modulverhalten ist in äquivalente Cluster unterteilt.
G25f	Verhalten unter Eingriff von Sicherheitsmechanismen ist beschrieben.
G25g	Die „Leistungsfähigkeit“ des Moduls ist beschrieben.
G25h	Die Auswirkungen des Modulverhaltens auf das Systemverhalten sind beschrieben.
G27	Informationen zur Modulspezifikation sind ermittelt.
G27a	Das System ist mit ausreichend äquivalenten Schnittstellen bzw. Modulen über mehrere Generationen evolutionär entwickelbar.
G27b	Entwicklungs- und Entscheidungsprozess sowie Fehlerzustände und Irrtümer sind dokumentiert.
G27c	Modulfunktion und deren Einbindung ins System sind beschrieben.

ID	Ungelöste Ziele der GSN Argumentationskette
G27d	Die Modulumgebung und deren Einflüsse auf das Modul sind möglichst allgemein beschrieben.
G27e	Die Einflüsse auf das Modul sind mit Systemszenarien verknüpft.
G27f	Der Anwendungsfall des Moduls ist möglichst allgemein beschrieben.
G28a	Abläufe, Informationsflüsse und bekannte Schwachstellen/Fehlerzustände aus ggf. vorherigen Entwicklungen eines Moduls sind beschrieben.
G28b	Auslöser und Bedingungen für Fehlerzustände sind beschrieben.
G28c	Sicherheitsmechanismen sind beschrieben.
G28d	Die Leistungsfähigkeit ist mit einem zuvor definierten Leistungsmaß beschrieben.
G28e	Kriterien für die notwendige Leistungsfähigkeit eines Moduls sind beschrieben.
G28f	Übertragungsverhalten des Modulverhaltens auf Systemebene ist beschrieben.
G28g	Abweichungen (falscher Zeitpunkt/falsche Dauer) vom Soll-Zeitverhalten und deren Ursachen sind beschrieben.
G29a	Parameterwerte und mögliche Kombination an den Modulschnittstellen sind beschrieben.
G29b	Einflussfaktoren auf Parameterwerte und die Sensitivität des Moduls darauf sind beschrieben.
G29c	Äquivalenz- bzw. Leistungsklassen von Parameterwerten der Eingaben und Ausgaben bzw. von Funktionen sind beschrieben.
G29d	Grenzwerte bzw. minimale und maximale Leistungsfähigkeit eines Moduls.
G29e	Verbindung der Äquivalenzklassen und Grenzwerte auf Modulebene mit Szenarien auf Systemebene ist beschrieben.
G29f	Verbindung der Äquivalenzklassen und Grenzwerte auf Systemebene mit Parameterkombinationen auf Modulebene ist beschrieben.
G29g	Äquivalente Parameterwerte für die Ausgaben eines Moduls sind bestimmt.
G34	Zulässige Werte, Zeitverhalten, mögliche Verläufe einschließlich eines Fehlermodells sind beschrieben.
G35	Auswirkung auf Schnittstellenverhalten bei unterschiedlichen Zuständen der Module bzw. des Systems sind beschrieben.
G37	Schnittstellenbeschreibung beinhaltet Informationen zur Unterstützung einer validen Darstellung in einer Simulationsumgebung.
G38	Schnittstellenbeschreibung beinhaltet Informationen zur Bewertung von Testergebnissen.

B Detaillierte semantische Schnittstellenbeschreibung – S^2I^2

Im folgenden Anhang wird die Umsetzung S^2I^2 durch detaillierte Beschreibung aller Attribute und anhand zweier Beispiele gezeigt. Die Tabellen B-2 bis B-5 geben die Beschreibung der Attribute an. Die Tabellen B-6 bis B-9 zeigen das Beispiel aller Attribute einer Trajektorie als Schnittstelle zwischen der Verhaltens- und Trajektorien-planung und dem Fahrdynamik- und Trajektorienregler. Abbildung B-1 demonstriert zusätz-lich eine beispielhafte Modellierung von S^2I^2. Die Einflüsse zwischen Attributen sind dabei durch Pfeile dargestellt. Die Diagramme zur Darstellung der Beschleunigung einer Trajek-torie sind der Arbeit von Blödel[419] entnommen. Die restlichen Inhalte des Modells basieren auf den vorherigen Tabellen. Die Tabellen B-10 bis B-13 demonstrieren zusätzlich die Anwendung von S^2I^2 auf die mechanische Schnittstelle zwischen Motor und Getriebe zur Übertragung eines Drehmoments.

B.1 Beschreibung und Referenzen der S^2I^2 Attribute

In den folgenden Tabellen sind alle Attribute von S^2I^2 aufgelistet und beschrieben. Zusätzlich sind in der rechten Spalte die verwendeten Referenzen aus der Literatur genannt, die zur Herleitung des jeweiligen Attributs verwendet wurden. Hierzu zählen die Interface Definition Language (IDL)[420], die Hardware Software Interface Beschreibung (HSI)[421], die Schnittstellenbeschreibungen nach Bachmann et al.[422] (B.) sowie die Kategorien für Informationsqualität (IQ) nach Batini und Scannapieco[423] sowie Wang und Strong[424]. Hinter der jeweiligen Abkürzung ist in Klammern die Bezeichnung der entsprechenden Quelle angegeben. Darüber hinaus sind den Attributen die offenen Ziele der GSN Argumentationskette zugeordnet, für die sie Informationen liefern, um die Argumentation zu unterstützen, dass eine modulare Absicherung möglich ist. Für den Fall, dass ein Attribut rein argumentativ oder auf Basis anderer Attribute abgeleitet ist, ist dies in der Spalte kurz beschrieben.

---

[419] Blödel, A.: Master Thesis, Verfahren zur Generierung von Trajektorien (2021).

[420] Object Management Group: Interface Definition Language (2018).

[421] ISO: ISO 26262 - Road vehicles – Functional safety (2018), part 4, p. 12, p. 31-32.

[422] Bachmann, F. et al.: Documenting Software Architecture: Documenting Interfaces (2002).

[423] Batini, C.; Scannapieco, M.: Data and information quality (2016).

[424] Wang, R. Y.; Strong, D. M.: What Data Quality Means to Data Consumers (1996).

Tabelle B-2: Attribute der Kategorie Syntax.

Attribut	Beschreibung	Referenzen
Name	Name, um die Schnittstelle kurz zu beschreiben.	IDL (name); B. (interface identity)
Identifikation	String oder Wert zum Aufrufen der Schnittstelle.	IDL (id)
Versionsnummer	Nummer zur Identifikation der Version der Schnittstelle. Die Nummer sollte sich bei jeder signifikanten Änderung ändern, falls diese die Schnittstellenbeschreibung der Schnittstelle beeinflusst.	B. (interface identity); Err.: Demand for traceability of the information.
Datentyp	Beschreibung des genutzten Datentyps.	IDL (data type)
Signallänge	Signallänge in Bits.	Abgeleitet vom Attribut *Datentyp*.
Standardwert	Standardwerte, um bspw. zu kommunizieren, dass sich ein Modul im Standby-Modus befindet oder dass das Modul noch keine nützlichen Informationen liefert, aber ordnungsgemäß funktioniert.	IDL (default)
Quelle	Name des Moduls oder Dienstes, das die Informationen über die Schnittstelle veröffentlicht.	Es kann von Vorteil sein, um zu verstehen, woher die Daten kommen, und es kann notwendig sein, die Schnittstelle von der richtigen Quelle aus aufzurufen oder zu abonnieren.
Einheit	Die Maßeinheit, die zur Beschreibung der Werte verwendet wird.	IDL (unit)
Referenzsystem	Beschreibt ergänzend zur Einheit die Referenz, auf die sich die Werte beziehen.	G35, G38 und abgeleitet vom Attribut *Einheit.*
Eltern / Struct	Name der übergeordneten Schnittstelle.	Abgeleitet vom Attribut *Position*.
Position	Position auf der übergeordneten Schnittstelle.	IDL (position)
Größe / Anzahl	Größe der in einem Zyklus gesendeten Informationen.	IDL (bit bound)
Verfügbarer Wertebereich	Verfügbarer Wertebereich an der Schnittstelle.	IDL (range, min, max), G34
Diskretisierung	Diskretisierung der Werte.	Abgeleitet von den Attributen *verfügbaren Wertebereich* und *Größe / Anzahl*.
Klassifikation des Informationsflusses	Klassifizierung des zeitlichen Verhaltens der Schnittstelle in Bezug auf ihren Informationsfluss. Zum Beispiel können Signale kontinuierlich oder diskontinuierlich, periodisch oder nicht-periodisch, deterministisch oder stochastisch sein, wie von Woyczyński[425] beschrieben.	HSI (Access mechanism to hardware resources)
Frequenz	Frequenz des bereitgestellten Signals.	Abgeleitet vom Attribut *Zykluszeit*.
Zykluszeit	Zeit, die erforderlich ist, um ein neues Signal zu liefern. Je nach Bezugspunkt können unterschiedliche Zykluszeiten gelten. Die reguläre Variabilität wird als „Jitter“ bezeichnet.	HSI (Timing constraints)

[425] Woyczyński, W. A.: Statistics for signal analysis (2006), S. 1–3.

Tabelle B-3: Attribute der Kategorie Semantik.

Attribute	Description	Referenz
Zweck / Bedeutung / Anwendungsfall	Beschreibung, wofür die über diese Schnittstelle übertragenen Informationen aus Sicht des Moduls und des Systems verwendet werden.	G4, G16, G17, G25a, G27f
Gültiger Wertebereich	Beschreibt, welche Werte für den entsprechenden Zweck oder Anwendungsfall gültig sind.	G34, G29a und abgeleitet vom Attribut *verfügbarer Wertebereich*.
Werte- oder Informationsverlauf	Mögliche Werte- oder Informationsverläufe an der Schnittstelle, ähnlich der Beschreibung von Szenarien auf Systemebene.	G25b, G25c, G28a, G28d, G28f, G29a, G37
Trigger	Auslöser, die die zuvor beschriebenen Verläufe verursachen.	G27d, G29b
Bedingungen / Konsistenzkriterien	Beschreibung von Bedingungen, Beschränkungen oder Konsistenzkriterien zu anderen Werten im System / in der Systemumgebung.	G15, G27c
Interdependenzen	Beschreibung von Abhängigkeiten, Beschränkungen oder Konsistenzkriterien zu anderen Elementen innerhalb einer Struktur einer Schnittstelle.	HSI (Independence between elements), G15
Redundanzen	Beschreibt mögliche redundante Informationen, die für die Funktionalität nicht notwendig sind. Zusätzlich wird der Zweck der Redundanz (z. B. aus Sicherheitsgründen) beschrieben.	IDL (optional), G28c
Latenzen	Angabe von Zeiten, die für verschiedene Ereignisse auf Modul- oder Systemebene vergehen bis Informationen bereitgestellt werden.	HSI (Timing consistency), G28g
Aktualität	Spezifische Latenz, die die Zeit beschreibt, die seit dem Auftreten des ursprünglichen Ereignisses (meist auf Systemebene) vergangen ist, das zu den bereitgestellten Informationen führt.	IQ, G28d, G28e, G28g, G29d und abgeleitet vom Attribut *Latenz*.
Verfügbarkeit	Bedingungen, die für die Veröffentlichung von Informationen erforderlich sind.	G35
Vor- / Nachbedingungen	Bestimmte Bedingungen bevor und nachdem Informationen auf der Schnittstelle veröffentlicht werden können.	HSI (Initialization)
Unterbrechungsmechanismen	Bedingungen und deren Auswirkungen, die die Veröffentlichung der Informationen stoppen. Hierzu zählen auch erkannte Versagensfälle und entsprechende Sicherheitsmechanismen.	HSI (Interrupts), G25d G28b, G28c, G28d, G28f
Genauigkeit	Genauigkeit der Informationen.	IQ, G25g, G28d, G28e, G29d
Präzision	Präzision der Informationen.	IQ, G25g, G28d, G28e, G29d
Korrektheit	Korrektheit der Informationen.	IQ; HSI (data integrity), G25g, G28d, G28e, G29d
Informationsvolatilität	Beschreibung, wie die Qualität der Informationen während der Laufzeit schwankt sowie mögliche Gründe dafür.	IQ, G25g, G28d, G28e, G29d
Vollständigkeit	Vollständigkeit der Informationen, falls möglich, dass diese unvollständig sein können.	IQ, G28d, G28e, G29d

Tabelle B-4: Attribute der Kategorie Einflussfaktoren.

Attribute	Description	Reference
Initialisierung	Beschreibung, wie Attribute der Semantik durch die Initialisierung beeinflusst werden.	HSI (Initialization)
Abschaltung	Beschreibung, wie Attribute der Semantik bei Abschaltung beeinflusst werden.	Abgeleitet vom Attribut *Initialisierung*.
Modi	Beschreibung der Unterschiede in der Semantik für verschiedene Betriebs- und Degradationsmodi. Dazu gehört auch eine Beschreibung der Unterschiede bei Verwendung und Nichtverwendung der Informationen der Schnittstelle.	HSI (Operating mode), G25d, G35
Moduswechsel	Einfluss von Moduswechseln auf Semantik.	Abgeleitet vom Attribut *Modi*.
Modulkonfigurationen	Möglich Modulkonfigurationen und wie diese die Semantik der Schnittstelle beeinflussen.	HSI (Configuration parameters), G35
Konfigurationen des Systems / anderer Module	Mögliche Konfigurationen anderer Module oder des Systems und wie diese die Semantik der Schnittstelle beeinflussen.	Abgeleitet vom Attribut *Modulkonfigurationen*.
Systemszenarien	Beschreibung und Gruppierung von Systemszenarien, die zu signifikant unterschiedlichen Ausgaben an der Schnittstelle führen. Die Ausgaben der Schnittstelle sind dazu in Äquivalenzklassen unterteilt.	G37, G25b, G25e, G27e, G27f, G29b, G29c, G29f, G29g.
Verhalten anderer Module	Beschreibung und Gruppierung des Verhaltens anderer Module, die zu signifikant unterschiedlichen Ausgaben an der Schnittstelle führen. Die Ausgaben der Schnittstelle sind dazu in Äquivalenzklassen unterteilt.	G25e, G27c, G27d, G29b, G29c, G29g, G37
Vorverarbeitungs- / Transferkanal	Beschreibung des Einflusses von zusätzlichen Verarbeitungs- und Übertragungsschritten auf die Attribute der *Semantik*.	HSI (Message transfer), G27c, G28d, G37

Tabelle B-5: Attribute der Kategorie Auswirkungen.

Attribute	Description	Reference
Äquivalenzklassen	Beschreibung der Äquivalenzklassen des Parameterraums. Von jedem Wert in einer Äquivalenzklasse wird erwartet, dass dieser das gleiche Verhalten bei anderen Modulen oder dem Gesamtsystem hervorruft.	ISO 26262[426] nutzt Äquivalenzklassen für das schnittstellenbasierte Testen, G29c
Erwartetes Systemverhalten	Beschreibung des Verhaltens, das auf Systemebene auf Basis der Attribute der *Semantik* erwartet wird.	Exp.: When changing driving direction in standstill, wheels are only directed into the driving direction when velocity is not zero. G25h, G28f, G29e, G37
Ausgaben anderer Module	Beschreibung der Ausgaben, die von anderen Modulen auf Basis der Attribute der *Semantik* erwartet werden.	Abgeleitet vom Attribut *Erwartetes Systemverhalten*, G28f, G29e, G37
Folgende Verarbeitungsschritte	Beschreibung der folgenden Verarbeitungsschritte der bereitgestellten Informationen, um das grundlegende Verständnis der Architektur zu unterstützen.	G27c, G29e, G37
Nutzende Komponenten	Auflistung bekannter Komponenten bzw. Module, die die Schnittstelle abonniert haben, um das grundlegende Verständnis der Architektur zu unterstützen.	Abgeleitet von der Notwendigkeit des Attributs *Folgende Verarbeitungsschritte*, G27c
Relevanzmetriken	Metrik zur Bewertung der Ausgabe der Schnittstelle auf Grundlage der von anderen Modulen benötigten Informationen.	G28d, G28e, G38 Außerdem beschreiben Amersbach and Winner[427], dass nicht alle Parameter für alle funktionalen Dekompositionsebenen relevant sind. Dies lässt sich auf Module übertragen, die möglicherweise nur einige Informationen benötigen, während andere für die Erfüllung ihrer Anforderungen irrelevant sind.

[426] ISO: ISO 26262 - Road vehicles – Functional safety (2018), S. 17–18.

[427] Amersbach, C.; Winner, H.: Overcome the parameter space explosion (2019).

B.2 Anwendung von S^2I^2 für eine Trajektorie als Schnittstelle

Tabelle B-6: Beschreibung der Kategorie Syntax für eine Trajektorie beispielhaft für die Schnittstelle Trajektorie als Struct und dem Beschleunigungsbetrag als Zelle innerhalb des Structs. (Abkürzungen: n/a = nicht anwendbar, n/k = nicht bekannt)

Attribut	Ausdruck Struct	Ausdruck Cell	Reguläre Variabilität	Irregularitäten	Ursachen für Irregularität
Name	Trajektorie	Soll-Beschleunigungsbetrag	n/a	n/a	n/a
Identifikation	trajectory	.acc_enu_ mag_m_s2	n/a	n/a	n/a
Versionsnummer	0.1	n/a	Änderungen der Dezimalstelle bedeuten, dass S^2I^2 von einer Änderung nicht betroffen ist.	n/a	n/a
Datentyp	Struct	Sequence double	n/a	n/k	n/k
Signallänge	n/a	64 bit	n/a	n/k	n/k
Standardwert	n/a	0	n/a	n/a	n/a
Quelle	Verhaltens- und Trajektorienplaner (VTP)	n/a	Die Quelle kann sich bei Modusänderungen oder nach Updates ändern.	Eingabe aus zwei verschiedenen Quellen gleichzeitig.	Falsche Verknüpfung von Diensten durch ASOA.
Einheit	n/a	m/s^2	n/a	n/a	n/a
Referenzsystem	n/a	Polarkoordinatensystem, Ursprung in der Mitte zwischen Verbindungslinie der Räder.	n/a	n/a	n/a
Eltern / Struct	n/a	trajectory	n/a	n/a	n/a
Position	n/a	8	n/a	n/a	n/a
Größe / Anzahl	9 verschiedene Parameter	Vektor mit 51 Elementen	n/a	n/a	n/a
Verfügbarer Wertebereich	n/a	[0; 100] m/s^2	n/a	n/b	n/b
Diskretisierung	n/a	0.01 m/s^2	Keine.	n/b	n/b
Klassifikation des Informationsflusses	Periodisch, diskontinuierlich	n/a	n/a	n/a	n/a

Attribut	Ausdruck Struct	Ausdruck Cell	Reguläre Variabilität	Irregularitäten	Ursachen für Irregularität
Frequenz	10 Hz	n/a	+/- 1 Hz	n/a	n/a
Zykluszeit	100 ms	n/a	+/- 10 ms	Bis zu 510 ms	Berechnung konvergiert nicht, sodass nach 5 Zyklen die Berechnung abgebrochen wird.

Tabelle B-7: Beschreibung der Kategorie Semantik für eine Trajektorie beispielhaft für die Schnittstelle Trajektorie als Struct und dem Beschleunigungsbetrag als Zelle innerhalb des Structs. (Abkürzungen: n/a = nicht anwendbar, n/k = nicht bekannt)

Attribut	Ausdruck Struct	Ausdruck Cell	Reguläre Variabilität	Irregularitäten	Ursachen für Irregularität
Zweck / Bedeutung	Die Trajektorie stellt die geplanten Posen des Fahrzeugs über die Zeit dar.	Beschleunigungsbetrag in Polarkoordinaten für jedes Trajektorienelement.	n/b	n/b	n/b
Gültiger Wertebereich	n/a	[0; 15] m/s^2	Einzelner Wert springt auf 100 m/s^2, falls Ableitung der Geschwindigkeit mit $\Delta t = 0$.	Mehr als ein Wert über dem gültigen Bereich innerhalb eines Zeitraums von 5 s.	Irrtum bei Initialisierung der Startposition führt zu Sprüngen der Zielgeschwindigkeit.
Werte- oder Informationsverlauf	(Beschreibung von Trajektorien, die auf repräsentativen Manövern basieren.)	Abhängig von Trajektorie. Sprunghafte Änderungen >5 m/s^2 bei Notmanövern.	Beschleunigungssprung bei Anfahren und Anhalten bis zu 2 m/s^2.	Sprunghafte Änderungen >5 m/s^2 obwohl kein Notmanövern.	Falsch-Positive Detektion von dynamischen Objekten.
Trigger	Situationsänderungen; Entscheidungsänderungen	n/a	n/b	n/b	Falsche Verknüpfung von Diensten durch ASOA.
Bedingungen / Konsistenzkriterien	Vorgaben müssen die Grenzen der Fahrdynamik berücksichtigen und gleichzeitig den aktuellen dynamischen Zustand des Fahrzeugs berücksichtigen.	Die von den Dynamikmodulen während der Laufzeit veröffentlichten Beschleunigungsgrenzen müssen berücksichtigt werden.	Das Beschleunigungslimit kann bei Vorgabe einer Notbremsung überschritten werden.	Geänderte Beschleunigungsgrenzen werden in der nächsten Trajektorie nicht beachtet.	Die Berechnung der Trajektorie konvergiert nicht schnell genug.
Interdependenzen	Position, Geschwindigkeit, Beschleunigung.	n/a	n/b	n/b	n/b
Redundanzen	Position, Geschwindigkeit, Beschleunigung.	Ableitung der Geschwindigkeit über die Zeit.	n/b	n/b	n/b

Attribut	Ausdruck Struct	Ausdruck Cell	Reguläre Variabilität	Irregularitäten	Ursachen für Irregularität
Latenzen	~10 ms, gemessen in HiL	n/a	+/- 5 ms	Up to 115 ms	Netzwerk durch anderen Verkehr blockiert. Erzwungene Priorität nach 100 ms.
Aktualität	500 ms, nachdem sich eine Situation auf Systemebene geändert hat, die zu einer neuen Trajektorie führt.	n/a	+/- 150 ms	Bis zu 1500 ms	Objekt wird nach Änderung der Situation vergleichsweise spät erkannt.
Verfügbarkeit	Verfügbar 5 Sekunden nach Einschalten der Automation.	n/a	Die Startprozedur kann sich ändern, so dass die Zeitdauer kürzer oder länger sein kann.	Verfügbar erst nach mehr als 5 Sekunden.	Internet- oder Satellitenverbindung nicht verfügbar.
Vorbedingungen	Route empfangen; Ausreichend genaue Lokalisierungsdaten empfangen; Berechnung der Trajektorie ist konvergiert; Die letzte Trajektorie wurde gesendet.	n/a	n/b	n/b	n/b
Nachbedingungen	Keine.	n/a	n/a	n/a	n/a
Unterbrechungsmechanismen	Die Werte werden auf die Standardwerte gesetzt, wenn: - Startvorgang nicht abgeschlossen ist - Keine gültige Trajektorie berechnet ist Das Signal kann verloren gehen aufgrund von: - Begrenzter Rechenressourcen (Zykluszeit wird nicht eingehalten) - Speicherlecks - Hardware- oder Kommunikationsausfälle	Der Wert wird auf den Standardwert gesetzt, wenn: Irrtümer/Versagensfälle bei der Berechnung auftreten.	n/k	n/k	n/k

Attribut	Ausdruck Struct	Ausdruck Cell	Reguläre Variabilität	Irregularitäten	Ursachen für Irregularität
Genauigkeit	Genauigkeit gegenüber einer Referenztrajektorie. Hängt von der Genauigkeit der Lokalisierung, der Umfelderkennung und der Berechnung der Trajektorie ab. Auf dieser Grundlage können die einzelnen Parameter quantifiziert werden.	0.1 m/s^2	0.1 m/s^2	0.5 m/s^2	Irrtum in der Berechnung aufgrund einer plötzlichen Änderung der Trajektorie.
Präzision	Geschätzte Standardabweichung der Parameter zu einer optimalen Referenztrajektorie (mit niedrigster Kritikalität und Optimum für Komfort und Dynamik)	0.05 m/s^2	0.05 m/s^2	0.2 m/s^2	n/b
Korrektheit	Die Korrektheit der Trajektorie hängt von der Korrektheit des Umweltmodells und der Entscheidung des Verhaltensplaners ab.	n/a	Ungewissheit über Umfeld steigt für spätere Zeitpunkte, sodass Korrektheit späterer Trajektorienposen ebenfalls geringer ist.	n/b	n/b
Informationsvolatilität	Die Trajektorie ändert sich, wenn sich das Umgebungsmodell oder die Verhaltensentscheidung ändert.	n/a	Trajektorie ändert sich kontinuierlich mit Änderungen des Umfeldmodells und Änderungen des Verhaltenplaners.	Trajektorie ändert sich plötzlich mit hoher Abweichung der Posen zur vorigen Trajektorie.	Umfeldmodell oder Verhaltensentscheidung ändert sich plötzlich in hohem Umfang.
Vollständigkeit	Letzte 10 Elemente der Trajektorie nicht erforderlich.	n/a	0 bis 10 der letzten Elemente können fehlen.	Es fehlen mehr als 10 Elemente.	Berechnung der neuen Trajektorie zu langsam.

Tabelle B-8: Beschreibung der Kategorie Einflussfaktoren für eine Trajektorie beispielhaft für die Schnittstelle Trajektorie als Struct. In dieser Kategorie ist keines der Attribute auf den Beschleunigungsbetrag anwendbar (Abkürzungen: n/a=nicht anwendbar, n/b=nicht bekannt)

Attribut	Ausdruck Struct	Reguläre Variabilität	Irregularitäten	Ursachen für Irregularität
Initialisierung	Nach dem Start findet eine Initialisierung statt, bis das Umgebungsmodell aufgebaut und die erste Trajektorie berechnet ist.	Die Initialisierung dauert 10 bis 30 Sekunden.	Mehr als 30 Sekunden für die Initialisierung erforderlich bei erhöhter Ungewissheit im Umweltmodell z. B. aufgrund von Regen.	Das Fahrzeug befindet sich nicht in der spezifizierten ODD.
Abschaltung	Vor Abschaltung muss das Fahrzeug sicher zum Stillstand kommen und die Trajektorie eine Parkposition ausgeben.	n/b	Abschaltung der Automation, obwohl Trajektorien ohne Parkposition ausgegeben werden.	Ein Operator steuert das Fahrzeug via Teleoperation aufgrund eines Ausfalls.
Modi	Automatisierung/Sicheres Anhalten: Keine Unterschiede zu beschriebenen Eigenschaften. Andere Modi: Trajektorie nicht verfügbar.	Zusätzliche Modi können hinzugefügt werden, wenn Trajektorien die gleichen Eigenschaften aufweisen.	n/a	n/a
Moduswechsel	Automation→ Sicheres Anhalten: Neue Trajektorie, basierend auf gewünschter Notbahn.	Ruckartige Bewegung aufgrund inkonsistenter Bewegungsdaten ggü. der vorherigen Trajektorie.	n/b	n/b
Modulkonfigurationen	n/b	n/b	n/b	n/b
Konfigurationen des Systems / anderer Module	Die Perzeption ist so konfiguriert, dass sie auch kleine Objekte wie Dosen einbezieht, was zu stärker schwankenden Trajektorien führt.	Die Perzeption wird aktualisiert, um noch kleinere Objekte zu berücksichtigen.	n/a	n/a
Systemszenarien	Fußgänger betritt Fußgängerüberweg → Trajektorie mit Verzögerung vor Fußgängerüberweg.	Regen verringert Reibkoeffizient. → Verzögerungsanforderung übersteigt verfügbare Bremskraft.	Fußgänger betritt plötzlich die Straße. → Trajektorie beinhaltet sprungartig Ausweichmanöver.	Fußgänger versucht den Bus zu erwischen.
Verhalten anderer Module	Durch FTR und Dynamikmodul verursachte Regelabweichung; Verlauf der Lokalisierungsgenauigkeit (z. B. führen Sprünge in der Genauigkeit zu Sprüngen im Lokalisierungssignal)	n/b	n/b	n/b
Vorverarbeitungs- / Transferkanal	Transformation von Sollpositionen in ein lokales Koordinatensystem durch Dienst zur Vorverarbeitung.	Vorverarbeitung nicht erforderlich, da andere Dienste das gleiche Referenzsystem nutzen.	Ungenauigkeiten der Trajektorien-Posen durch Transformation.	Verwendung der falschen Bezugspunkte für die Transformation.

Tabelle B-9: Beschreibung der Kategorie Auswirkungen für eine Trajektorie beispielhaft für die Schnittstelle Trajektorie als Struct und dem Beschleunigungsbetrag als Zelle innerhalb des Structs. (Abkürzungen: n/a = nicht anwendbar, n/k = nicht bekannt)

Attribut	Ausdruck Struct	Ausdruck Cell	Reguläre Variabilität	Irregularitäten	Ursachen für Irregularität
Äquivalenzklassen	Pfade, die sich durch verschiedene Manöver unterscheiden.	[-10; -8; -5; -2; -0.5; -0.25; 0; 0.25; 0.5; 1; 2] m/s^2	n/b	n/b	n/b
Erwartetes Systemverhalten	Fahrzeuge sollen sich entsprechend der Trajektorie fortbewegen.	Die Beschleunigung wird durch Dynamikmodule umgesetzt.	Negative Abweichung erlaubt, um das Positionsziel zu erreichen. Positive Abweichung bis zu 0.2 m/s^2, wenn geforderte Geschwindigkeit nicht um mehr als 0,25 m/s überschritten wird.	Verzögerung nicht wie gewünscht durchgeführt.	Die Bremsen sind zu heiß, um ein ausreichendes Bremsmoment zu erzeugen.
Ausgaben anderer Module	Der FTR berechnet die Stellgrößen für die Aktoren, um der Trajektorie zu folgen.	Der FTR berechnet ein Brems- oder Motordrehmoment auf Basis der Beschleunigungsanforderung der Trajektorie.	Berechnetes Drehmoment ist kleiner als das gewünschte, da das verfügbare Drehmoment zu klein ist.	n/b	n/b
Folgende Verarbeitungsschritte	(In diesem Fall besteht kein Unterschied zum Attribut *Ausgabe anderer Module*.)	(In diesem Fall besteht kein Unterschied zum Attribut *Ausgabe anderer Module*.)	n/a	n/a	n/a
Nutzende Komponenten	Fahrdynamik- und Trajektorienregler (FTR)	n/a	Nutzung durch beliebige andere Module, z. B. zur Prädiktion des Verhaltens in der Cloud.	n/a	n/a
Relevanzmetriken	Metrik zur Bewertung, ob die Trajektorie von der FTR ohne Stabilitätsprobleme ausgeführt werden kann.	n/a	n/a	n/a	n/a

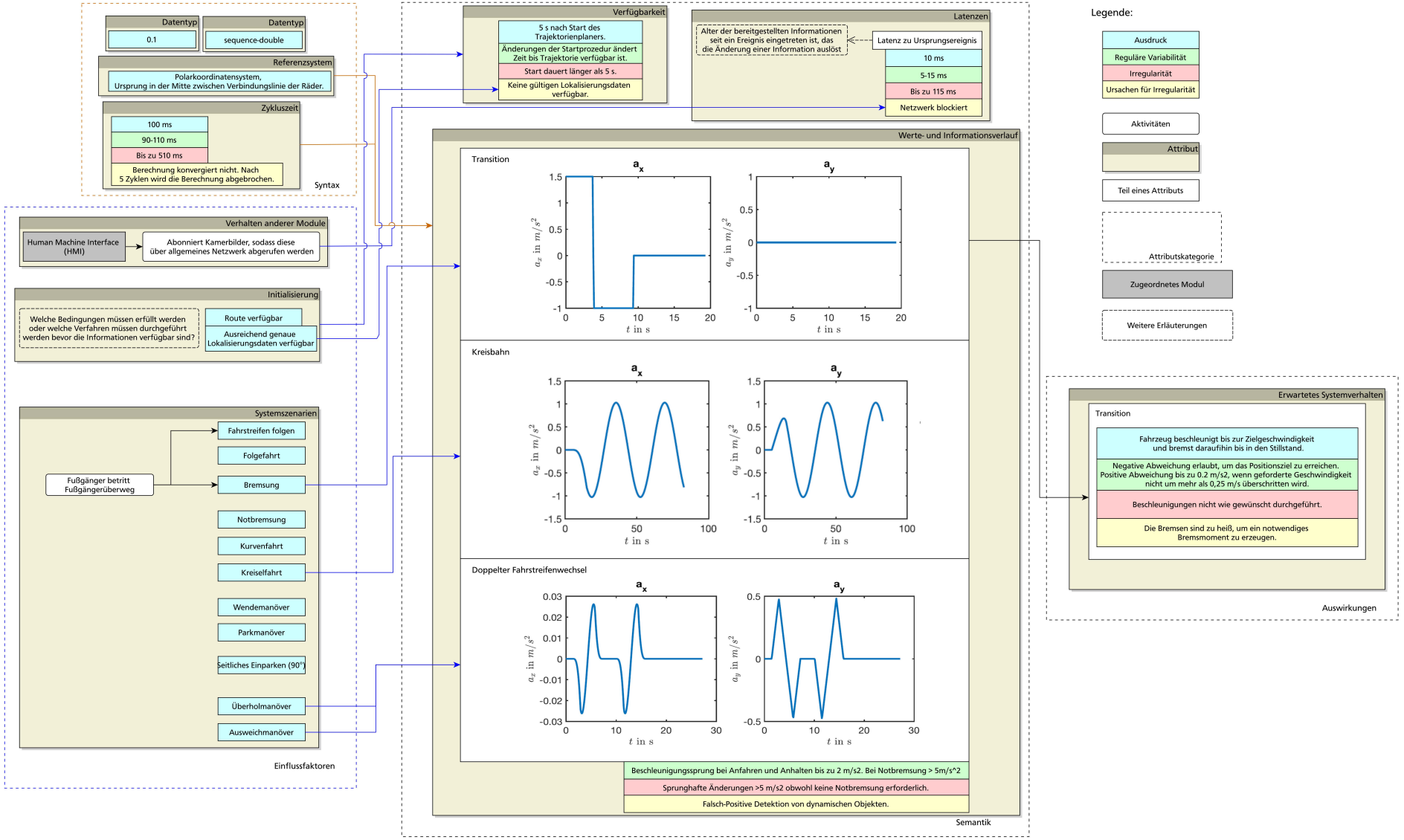


Abbildung B-1: Beispielhafte Modellierung von S^2I^2 für das Beispiel einer Trajektorie und der darin enthaltenen Beschleunigungsvorgabe.

B.3 Anwendung von S^2I^2 für eine Motorwelle als Schnittstelle

Tabelle B-10: Beschreibung der Kategorie Syntax für eine Motorwelle beispielhaft zur Übertragung des Drehmoments zwischen Motorwelle und Getriebe. (Abkürzungen: n/a = nicht anwendbar, n/k = nicht bekannt)

Attribut	Ausdruck	Reguläre Variabilität	Irregularitäten	Ursachen für Irregularität
Name	Drehmoment Motorwelle	n/a	n/a	n/a
Identifikation	Motor_shaft_torque	n/a	n/b	n/b
Versionsnummer	0.1	Änderungen der Dezimalstelle bedeuten, dass S^2I^2 von einer Änderung nicht betroffen ist.	n/k	n/k
Datentyp	n/a	n/a	n/a	n/a
Signallänge	n/a	n/a	n/a	n/a
Standardwert	0 Nm (Zielwert)	-5 Nm aufgrund von Widerstandsmomenten (z. B. durch Reibung)	-50 Nm	Kurzschluss zwischen den Stromanschlüssen des Elektromotors.
Quelle	Elektromotor	Andere Elektromotoren ankoppelbar.	n/b	n/b
Einheit	Nm	n/a	n/b	n/b
Referenzsystem	Motorachse, Motorflansch und Fahrtrichtung.	n/a	Der Getriebeausgang oder das Rad wird als Referenz genommen.	Referenzsystem nicht kommuniziert.
Eltern / Struct	Kopplung zwischen Motor und Getriebe.	Für die Übertragung können andere Kopplungen verwendet werden.	n/a	n/a
Position	n/a	n/a	n/a	n/a
Größe / Anzahl	Durch einen Wert beschreibbar.	n/a	n/a	n/a
Verfügbarer Wertebereich	[-250; 450]	[0; 100] falls Motor Hitzeschutz aktiv.	n/k	n/k
Diskretisierung	0 (analog)	n/a	n/a	n/a
Klassifikation des Informationsflusses	Kontinuierlich, nicht periodisch	n/k	n/k	n/k
Frequenz	n/a	n/a	n/a	n/a
Zykluszeit	n/a	n/a	n/a	n/a

Tabelle B-11: Beschreibung der Kategorie Semantik für eine Motorwelle beispielhaft für die Schnittstelle zur Übertragung des Drehmoments zwischen Motorwelle und Getriebe. (Abkürzungen: n/a = nicht anwendbar, n/k = nicht bekannt)

Attribut	Ausdruck	Reguläre Variabilität	Irregularitäten	Ursachen für Irregularität
Zweck / Bedeutung	Drehmoment wird über die Motorwelle zwischen Motor und Getriebe übertragen, um Fahrzeug zu beschleunigen.	n/b	n/b	n/b
Gültiger Wertebereich	[-250; 450]	[0; 100] falls Motor Hitzeschutz aktiv.	n/b	n/b
Werte- oder Informationsverlauf	Mögliche Verläufe des Drehmoments über die Drehzahl und die Zeit, die durch charakteristische Kurven in einem Lastkollektiv beschrieben werden.	n/b	n/b	n/b
Trigger	Beschleunigungsvorgabe.	Hitzeschutz führt zu einer Verringerung des verfügbaren Drehmoments.	n/b	n/b
Bedingungen / Konsistenzkriterien	Das maximale und minimale Drehmoment hängt von der Drehzahl sowie der Temperatur von Motor und Batterie ab.	n/b	n/b	n/b
Interdependenzen	n/a	n/a	n/a	n/a
Redundanzen	n/a	n/a	n/a	n/a
Latenzen	~20 ms, von der ersten Drehmomentanforderung bis zum Anliegen des vollen geforderten Drehmoments am Getriebe.	Bis zu 80 ms Aufbauzeit, wenn das Drehmoment vom Minimum zum Maximum wechselt.	n/b	n/b
Aktualität	~200 ms, vom Ereignis auf Systemebene, das die Drehmomentanforderung auslöst, bis zur Anwendung auf das Getriebe.	n/b	n/b	n/b
Verfügbarkeit	Drehmoment verfügbar 3 Sekunden nach Start.	n/b	n/b	n/b
Vorbedingungen	Elektrischer Strom angelegt.	n/b	n/b	n/b
Nachbedingungen	Abkühlung nach 10-maliger maximaler Beschleunigung erforderlich.	n/b	n/b	n/b
Unterbrechungsmechanismen	Eingriffe der elektronischen Stabilitätskontrolle.	n/b	n/b	n/b
Genauigkeit	+- 5 Nm zum gewünschten Drehmoment	Eingriffe der Schlupfregelung bzw. falls die Räder das geforderte Drehmoment nicht übertragen können.	n/b	n/b

Attribut	Ausdruck	Reguläre Variabilität	Irregularitäten	Ursachen für Irregularität
Präzision	Die maximale Steigung des Drehmomentanstiegs/-abfalls von 20 Nm/ms kann zu Abweichungen von der geforderten Drehmomentsteigung führen.	n/b	n/b	n/b
Korrektheit	n/a	n/a	n/a	n/a
Informationsvolatilität	n/a	n/a	n/a	n/a
Vollständigkeit	n/a	n/a	n/a	n/a

Tabelle B-12: Beschreibung der Kategorie Einflussfaktoren für eine Motorwelle beispielhaft für die Schnittstelle zur Übertragung des Drehmoments zwischen Motorwelle und Getriebe. (Abkürzungen: n/a = nicht anwendbar, n/k = nicht bekannt)

Attribut	Ausdruck	Reguläre Variabilität	Irregularitäten	Ursachen für Irregularität
Initialisierung	Drehmoment muss langsam erhöht werden, um Rücke zu vermeiden. Motorlager müssen Betriebstemperatur erreichen bevor maximales Drehmoment gestellt werden darf.	„Kickdown“ erlaubt maximales Drehmoment auch unterhalb der Betriebstemperatur der Motorlager.	Regelmäßiges Stellen des max. Drehmoments unterhalb der Betriebstemperatur.	Fahrer nicht über erhöhten Verschleiß bei regelmäßigem „Kickdown“ aufgeklärt.
Abschaltung	Keine Relevanz.	n/b	n/b	n/b
Modi	Jeder Fahrmodus: Keine Unterschiede zu den beschriebenen Eigenschaften.	n/b	n/b	n/b
Moduswechsel	Automation→Sicheres Anhalten: Plötzliche starke Reduktion von hohem positiven in hohes negatives Drehmoment möglich.	n/b	n/b	n/b
Modulkonfigurationen	Elektromotor mit Drehmoment bis zu 500 Nm.	n/b	n/b	n/b
Konfigurationen des Systems / anderer Module	Ein kritisches Szenario kann eine Plötzliche starke Reduktion von hohem positiven in hohes negatives Drehmoment erfordern.	n/b	n/b	n/b
Systemszenarien	Vibrationen der Karosserie übertragen sich auf Motorwelle.	n/b	n/b	n/b
Verhalten anderer Module	n/a	n/a	n/a	n/a

Attribut	Ausdruck	Reguläre Variabilität	Irregularitäten	Ursachen für Irregularität
Vorverarbeitungs- / Transferkanal	Drehmoment wird über Getriebe an die Antriebswelle und an die Räder übertragen.	Verwendung unterschiedlicher Räder- und Reifenkombinationen mit unterschiedlichen Massen, Massenverteilung und Durchmessern.	Nutzung nicht zulässiger Räder- und Reifenkombinationen.	Zulässigen Räder- und Reifenkombinationen nicht kommuniziert.

Tabelle B-13: Beschreibung der Kategorie Auswirkungen für eine Motorwelle beispielhaft für die Schnittstelle zur Übertragung des Drehmoments zwischen Motorwelle und Getriebe. (Abkürzungen: n/a = nicht anwendbar, n/k = nicht bekannt)

Attribut	Ausdruck	Reguläre Variabilität	Irregularitäten	Ursachen für Irregularität
Äquivalenzklassen	[-250; -150; -50; -5; 5; 50; 150; 250; 350; 450]	n/a	n/a	n/a
Erwartetes Systemverhalten	Beschleunigung des Fahrzeugs durch das an den Rädern anliegende Drehmoment.	n/b	n/b	n/b
Ausgaben anderer Module	Das Getriebe gibt das Drehmoment an die Räder ab.	n/b	n/b	n/b
Folgende Verarbeitungsschritte	Drehmoment wird durch Getriebe erhöht.	n/b	n/b	n/b
Nutzende Komponenten	Getriebe.	n/b	n/b	n/b
Relevanzmetriken	n/b	n/b	n/b	n/b

C Integrationstests

Die folgenden Integrationstests aus dem Forschungsprojekt UNICAR*agil* für Testumgebungen nach Tabelle 6-3 werden im Rahmen dieser Arbeit durchgeführt, um Unzulänglichkeiten bei der Spezifikation, der Implementierung oder dem Testen von Modulen aufzudecken. Tabelle C-14 stellt Testfälle dar, die die Module zur manuellen Steuerung der Fahrzeuge prüfen. Die Testfälle nach Tabelle C-15 wurden zunächst nicht bestanden und decken entsprechend Unzulänglichkeiten zuvor durchgeführter Modultests bzw. des Entwicklungsprozesses auf. Die Testfälle basieren auf den Arbeiten von Krause[428], Blödel[429], Kohls[430] und Smits[431]. Die ursprüngliche Definition der Testfälle wurde teils angepasst oder ergänzt, um in den jeweiligen Testumgebungen ausführbar zu sein. Die Testfälle stellen keinen hinreichenden Umfang für eine modulare Absicherung dar. Sie dienen vielmehr dazu eine erste Überprüfung der gewünschten Funktionen des Prototyps durchzuführen.

Die Erkenntnisse aus diesen Tests stellen außerdem Fehlerzustände und Irrtümer dar, die in einem Forschungsprojekt bei der Entwicklung eines neuen Prototyps gemacht werden. Die Ergebnisse entsprechen daher möglicherweise nicht Fehlerzuständen und Irrtümern, die in etablierteren, kontrollierten Entwicklungs- und Testprozessen für ein Serienprodukt auftreten.

---

[428] Krause, J.: Bachelor Thesis, Tests für die Freigabe des Dynamikmoduls (2021).

[429] Blödel, A.: Master Thesis, Verfahren zur Generierung von Trajektorien (2021).

[430] Kohls, S. K.: Master Thesis, Abhängigkeiten zwischen Modulen (2022).

[431] Smits, L.: Master Thesis, Testfälle für die Verhaltens- und Trajektorienplanung (2022).

C.1 Integrationstests der Dynamikmodule für manuelles Fahren

Tabelle C-14: Integrationstests der Dynamikmodule im Modus für manuelles Fahren (MD) des Projekts UNICAR*agil*.

Test Nr.	**Testname**	**Testziel**	**Testbeschreibung**	**Testschritte**	**Sollverhalten / Bestehenskriterien**	**Beobachtung**	**Testergebnis**
MD1-1	Prüfung des Startprozesses	Decke auf, dass der Fahrmodus aktiviert wird, obwohl sicherheitskritische Informationen nicht verfügbar sind.	Alle notwendigen Informationen und Verbindungen werden geprüft, bevor eine Fahrt möglich ist.	1. Trennung jeweils einer der aufgelisteten Hardware-Verbindungen, während das System ausgeschaltet ist: a. Flexray - Sidestick b. Stromversorgung - Sidestick c. CAN Bus - Bremspedal d. Stromversorgung - Bremspedal e. Ethernet – Stidestick (nur für Schritt 2b.)	Keine beobachtbare Reaktion, da das System abgeschaltet ist.	-Sollverhalten-	Pass
MD1-2				2. Startvorgang durchführen a. Manuelles Fahren b. Automatisiertes Fahren	Fahrmodus wird nicht aktiviert. Parkbremsen und Lenkbremsen bleiben geschlossen. Der Fehlermodus wird durch rote Leuchten am Sidestick signalisiert (sofern der Sidestick an die Stromversorgung angeschlossen ist).	-Sollverhalten -	Pass
MD2-1	Unterbrechung der Signalübertragung während des manuellen Fahrens.	Decke auf, dass ein unterbrochenes Signal oder Stromversorgung während manueller Fahrt zu einem	Definierte Verbindungen werden während einer simulierten Fahrt auf der Hebe-	1. Startvorgang durchführen und Räder beschleunigung während gleichzeitig kontinuierlich von links nach rechts gelenkt wird.	Räder drehen und lenken wie durch Sidestick und Bremspedal vorgegeben.	-Sollverhalten-	Pass

Test Nr.	Testname	Testziel	Testbeschreibung	Testschritte	Sollverhalten / Bestehenskriterien	Beobachtung	Testergebnis
		unsicheren Verhalten des Systems führt.	bühne unterbrochen, um eine mögliche Unterbrechung der Verbindung oder ein fehlendes Signal darzustellen.				
MD2-2				2. Trennung jeweils einer der aufgelisteten Hardwareverbindungen während der Fahrt: a. Flexray - Sidestick b. Stromversorgung - Sidestick c. CAN Bus - Bremspedal d. Stromversorgung - Bremspedal e. Ethernet – Sidestick	Bei getrennten Sidestick Verbindungen: Hydraulisches bremsen weiterhin möglich, Lenkwinkel durch Lenkwinkelbremsen fixiert, keine weitere Beschleunigung durch Motoren möglich. Bei getrennten Bremspedal Verbindungen: Sidestick leuchtet rot, hydraulisches bremsen nicht möglich, rekuperatives Bremsen mit Sidestick möglich, Lenkung voll funktionsfähig.	-Sollverhalten-	Pass
MD3-1	Unterbrechung der Signalübertragung im automatisierten Fahrmodus.	Decke auf, dass ein unterbrochenes Signal oder Stromversorgung während automatisierter Fahrt zu einem unsicheren Verhalten des Systems führt.	Definierte Verbindungen werden während einer simulierten Fahrt auf der Hebebühne unterbrochen, um eine mögliche Unterbrechung der Verbindung oder ein	1. Starte Wiedergabe zuvor generierter Stellgrößen für die Dynamikmodule im automatisierten Fahrmodus.	Die Räder bewegen sich entsprechend den generierten Stellgrößen.	-Sollverhalten-	Pass

Test Nr.	Testname	Testziel	Testbeschreibung	Testschritte	Sollverhalten / Bestehenskriterien	Beobachtung	Testergebnis
			fehlendes Signal darzustellen.				
MD3-2				2a: Trennung jeweils einer der aufgelisteten Hardwareverbindungen während der Wiedergabe der Stellgrößen: a. Flexray - Sidestick b. Stromversorgung - Sidestick c. Ethernetverbindung – Sidestick	Automatische Bremsung mit 3-5 m/s^2, Lenkwinkel durch Lenkwinkelbremsen fixiert, keine weitere Beschleunigung durch Motoren möglich, Fehlermodus am Sidestick signalisiert, hydraulisches bremsen mit Bremspedal weiterhin möglich.	Fehlermodus löst keine automatische Bremsung aus.	Fail
MD3-3				2b: Trennung jeweils einer der aufgelisteten Hardwareverbindungen während der Wiedergabe der Stellgrößen: a. CAN Bus - Bremspedal b. Stromversorgung - Bremspedal	Automatisierte Bremsung mit 3-5 m/s^2, Lenkwinkel durch Lenkwinkelbremsen fixiert, keine weitere Beschleunigung durch Motoren möglich, Fehlermodus am Sidestick signalisiert, hydraulisches bremsen mit Bremspedal nicht möglich.	Kein Wechsel in Fehlermodus, Stellgrößen werden weiter ausgeführt, Bremsung mit Bremspedal nicht mehr möglich, Verlassen des Automationsmodus mit Bremspedal nicht mehr möglich.	Fail
MD4-1	Prüfung des Bremsmoments	Decke auf, dass die Reibbremse nicht in der Lage ist, das Fahrzeug bei maximalem Motordrehmoment zu stoppen.	Reibbremse wird unter verschiedenen Bedingungen betätigt.	Beschleunigung auf eingestellte Höchstgeschwindigkeit von 5 km/h, wobei weiterhin eine maximale Beschleunigung vorgegeben wird. Währenddessen wird die Reibbremse mit maximaler Kraft betätigt.	Das Fahrzeug muss eine durchschnittliche Verzögerung von mindestens 5 m/s^2 erfahren und bis in den Stillstand kommen und darin verbleiben.	-Sollverhalten-	Pass

Test Nr.	Testname	Testziel	Testbeschreibung	Testschritte	Sollverhalten / Bestehenskriterien	Beobachtung	Testergebnis
MD4-2				Beschleunigung auf eingestellte Höchstgeschwindigkeit von 15 km/h, wobei weiterhin eine maximale Beschleunigung vorgegeben wird. Währenddessen wird die Reibbremse mit maximaler Kraft betätigt.	Das Fahrzeug muss eine durchschnittliche Verzögerung von mindestens 5 m/s^2 erfahren und bis in den Stillstand kommen und darin verbleiben.	Die Räder kommen zum Stillstand, beschleunigen aber unmittelbar danach wieder. Dieses Verhalten hält solange an, bis die Bremse oder das Gaspedal nicht mehr betätigt wird.	Fail
MD5-1	Parkbremse als Rückfallebene für die Primärbremse	Decke auf, dass das Bremsen über die Feststellbremse unter bestimmten Bedingungen nicht möglich ist.	Parkbremse wird unter verschiedenen Bedingungen betätigt.	Beschleunigung auf eingestellte Höchstgeschwindigkeit von, wobei weiterhin eine maximale Beschleunigung vorgegeben wird. Währenddessen wird die Reibbremse mit maximaler Kraft betätigt.	Das Fahrzeug muss eine durchschnittliche Verzögerung von mindestens 3 m/s^2 erfahren und bis in den Stillstand kommen und darin verbleiben.	Fahrzeug kann nicht durch einmaliges Ziehen des Schalters der Feststellbremse angehalten werden.	Fail
MD6-1	Rekuperation durch Motoren als Rückfallebene für die Primärbremse	Decke auf, dass das Bremsen über die Rekuperation unter bestimmten Bedingungen nicht möglich ist.	Rekuperationsbremse wird unter verschiedenen Bedingungen betätigt.	Maximale Beschleunigung und darauffolgend maximale Rekuperationsbremsung.	Das Fahrzeug muss eine durchschnittliche Verzögerung von mindestens 3 m/s^2 erfahren und bis in den Stillstand kommen und darin verbleiben.	Oberhalb der eigenstellten Höchstgeschwindigkeit erfolgt keine Rekuperationsbremsung. Nachdem die Geschwindigkeitsgrenze unterschritten wurde muss Trigger erneut betätigt werden, um Rekuperationsbremsung auszulösen.	Fail
MD7-1	Einhaltung der Höchstgeschwindigkeit	Decke auf, dass Höchstgeschwindigkeit nicht eingehalten wird.	Fahrzeug wird unter verschiedenen Bedingungen beschleunigt, um	Beschleunigung des Fahrzeugs auf Höchstgeschwindigkeit und	Höchstgeschwindigkeit wird um maximal 1 m/s überschritten.	Höchstgeschwindigkeit wird um etwa 3 m/s überschritten.	Fail

Test Nr.	Testname	Testziel	Testbeschreibung	Testschritte	Sollverhalten / Bestehenskriterien	Beobachtung	Testergebnis
			Höchstgeschwindigkeit zu überschreiten.	Halten der Beschleunigungsvorgabe für weitere 10 Sekunden nach Erreichen bzw. Überschreiten der Höchstgeschwindigkeit.			
MD8-1	Dokumentation sicherheitsrelevanter Ereignisse in der Integrationsphase der Fahrzeuge	Decke Fehlerzustände auf bei Integration verschiedener Modulkombinationen unter verschiedenen Bedingungen.	-	-	-	Primäre Hydraulikbremse fällt aus, erkennbar an Geräuschen der sekundären Hydraulikbremse.	-

C.2 Integrationstests der automatisierten Fahrfunktion

Tabelle C-15: Nicht bestandene Integrationstests der automatisierten Fahrfunktion des Projekts UNICAR*agil*. Verwendete Testumgebung nach Tabelle 6-3.

Test Nr.	Testname	Module under Test	Testbeschreibung	Testschritte	Sollverhalten / Bestehenskriterien	Beobachtung
G1	Positionsanfahrt mit Kurswinkel	Fahrdynamik- und Trajektorienregler (FTR)	Unzureichende Schnittstellenimplementierung für bestimmte Manöver identifizieren. Instabiles Verhalten aufgrund von Kurswinkeländerungen aufdecken.	Fahrt mit einem Kurswinkel von 0°; Bremsung in den Stillstand; Beschleunigung unter einem konstanten Kurswinkel größer/kleiner 0°; Bremsung in den Stillstand.	Fahrzeug erreicht Zwischen- und Endposition; Beschleunigungs-, Geschwindigkeits- und Positionsziele werden innerhalb der Toleranzen eingehalten; keine Positionsänderung während des Anhaltens; kein beobachtbares instabiles Verhalten; kein unerwartetes Verhalten.	Unerwartetes Verhalten: Radlenkwinkel verbleiben im Stillstand in der vorherigen (Null-)Position, obwohl in der Trajektorie bereits eine neue Richtung vorgegeben ist. Räder werden erst bei wieder anfahren ruckartig eingeschlagen, wodurch eine erhöhte Positionsabweichung entsteht.

Test Nr.	Testname	Module under Test	Testbeschreibung	Testschritte	Sollverhalten / Bestehenskriterien	Beobachtung
G2	Fahren in Längsrichtung	FTR	Decke instabiles Verhalten auf.	Fahrt einer longitudinalen Trajektorie.	Beschleunigungs-, Geschwindigkeits- und Positionsvorgaben innerhalb der Toleranzen; keine Positionsänderung während des Stopps; kein beobachtbares instabiles Verhalten; kein unerwartetes Verhalten.	Unerwartetes Verhalten: Die Simulation stoppt plötzlich.
G3	Abklingender Slalom	FTR	Decke instabiles Verhalten oder Überschreitung der Regelabweichungen durch wechselnde laterale Beschleunigung auf.	Fahrt eines sinusförmigen Slaloms mit bis auf 0 m abklinende Amplitude. Drive a slalom with decreasing lateral amplitude until lateral movement is 0 m; Bremsung in den Stillstand.	Beschleunigungs-, Geschwindigkeits- und Positionsvorgaben innerhalb der Toleranzen; keine Positionsänderung während des Stopps; kein beobachtbares instabiles Verhalten; kein unerwartetes Verhalten.	Unerwartetes Verhalten: Am Ende des abklingenden Slaloms beginnen die Radlenkwinkel zu ruckeln, während der resultierende seitliche Versatz klein gehalten wird. Auch im Stillstand alternieren die Radlenkwinkel weiter unter geringen Winkeländerungen.
E1	Annäherung eines von hinten kommenden Fahrzeugs	Sicheres Anhalten	Aufdeckung unsicherer Reaktionen auf nicht relevante dynamische Objekte.	Sicheres Anhalten wird aktiviert, um am Straßenrand sicher zum Stehen zu kommen; Objektfahrzeug nähert sich von hinten; Objektfahrzeug folgt dem Ego-Fahrzeug mit gleicher Geschwindigkeit; Ego-Fahrzeug fährt in die Endposition; verlangsamt bis zum endgültigen Halt.	Beschleunigungs-, Geschwindigkeits- und Positionsvorgaben innerhalb der Toleranzen; keine Positionsänderung während des Stopps; kein beobachtbares instabiles Verhalten; kein unerwartetes Verhalten; Sicherer Abstand zu anderen Objekten wird eingehalten.	Reaktion auf sich von hinten näherndes Objektfahrzeug mit erhöhter Verzögerung bis in den Stillstand, daher: Beschleunigungs- und Geschwindigkeitsziele nicht eingehalten; Endposition nicht erreicht; Sicherheitsabstand zu anderen Objekten nicht eingehalten.
E2	Durchscherer	Sicheres Anhalten	Aufdeckung unsicherer Reaktionen auf relevante dynamische Objekte.	Sicheres Anhalten wird aktiviert, um am Straßenrand sicher zum Stehen zu kommen; Objektfahrzeug überholt das	Beschleunigungs-, Geschwindigkeits- und Positionsvorgaben innerhalb der Toleranzen; keine Positionsänderung wäh-	Nach dem Fahrstreifenwechsel des Objektfahrzeugs beschleunigt das Ego-Fahrzeug stark und

Test Nr.	Testname	Module under Test	Testbeschreibung	Testschritte	Sollverhalten / Bestehenskriterien	Beobachtung
				Ego-Fahrzeug; Objektfahrzeug führt ein Durchschermanöver vor dem Ego-Fahrzeug durch, indem es zwei Fahrstreifen nach rechts wechselt; Objektfahrzeug hält außerhalb der Notbahn des Ego-Fahrzeugs.	rend des Stopps; kein beobachtbares instabiles Verhalten; kein unerwartetes Verhalten; Sicherer Abstand zu anderen Objekten wird eingehalten.	überschreitet die vorgesehene Höchstgeschwindigkeit, daher: Beschleunigungs- und Geschwindigkeitsziele nicht eingehalten; Endposition nicht erreicht; Sicherheitsabstand zu anderen Objekten nicht eingehalten.
D1	Fahren in Längsrichtung	FTR, Verhaltens- und Trajektorienplaner (VTP)	Unzureichende Schnittstellenimplementierung für bestimmte Manöver identifizieren. Decke instabiles Verhalten durch Trajektorienvorgaben auf.	Fahre entlang eines geraden Fahrstreifens.	Beschleunigungs-, Geschwindigkeits- und Positionsvorgaben innerhalb der Toleranzen; keine Positionsänderung während des Stopps; kein beobachtbares instabiles Verhalten; kein unerwartetes Verhalten.	Egofahrzeug fährt in die falsche Richtung.
D2	Variierte Fahrt über Teststrecke	FTR, VTP	Unzureichende Schnittstellenimplementierung für bestimmte Manöver identifizieren. Decke instabiles Verhalten durch Trajektorienvorgaben auf.	Fahrt einer Runde um die Teststrecke entlang des rechten Fahrstreifens.	Beschleunigungs-, Geschwindigkeits- und Positionsvorgaben innerhalb der Toleranzen; keine Positionsänderung während des Stopps; kein beobachtbares instabiles Verhalten; kein unerwartetes Verhalten; kein Überschreiten durchgezogener Fahrstreifenmarkierungen.	Permanente laterale Abweichung vom gewünschten Fahrstreifen um ca. 2 m.
D3	Überholen eines geparkten Fahrzeugs am Fahrbahnrand.	VTP	Decke unzureichende Beachtung der Verkehrsregeln oder unzureichende Reaktion auf andere Objekte auf.	Folgefahrt hinter Objektfahrzeug; Objektfahrzeug hält am Straßenrand an.	Das Egofahrzeug überholt das geparkte Fahrzeug mit einem sicheren Abstand und beachtet dabei die Verkehrsregeln.	Zu geringer seitlicher Abstand beim Überholen eines Objekts, was unter bestimmten Parametervariationen sogar zu Kollisionen führt.

Test Nr.	Testname	Module under Test	Testbeschreibung	Testschritte	Sollverhalten / Bestehenskriterien	Beobachtung
C1	Variierte Fahrt über Teststrecke	FTR, VTP	Unzureichende Schnittstellenimplementierung für bestimmte Manöver identifizieren. Decke instabiles Verhalten durch Trajektorienvorgaben auf.	Fahrt einer Runde um die Teststrecke entlang des rechten Fahrstreifens.	Beschleunigungs-, Geschwindigkeits- und Positionsvorgaben innerhalb der Toleranzen; keine Positionsänderung während des Stopps; kein beobachtbares instabiles Verhalten; kein unerwartetes Verhalten; kein Überschreiten durchgezogener Fahrstreifenmarkierungen.	Permanente Abweichung vom Soll-Gierwinkel nach Durchfahrt einer 180° Kurve, sichtbar durch permanenten Schwimmwinkel größer als 0°.
A1	Variierte Fahrt über Teststrecke	FTR, VTP	Unzureichende Schnittstellenimplementierung für bestimmte Manöver identifizieren. Decke instabiles Verhalten durch Trajektorienvorgaben auf.	Fahrt einer Runde um die Teststrecke entlang des rechten Fahrstreifens.	Beschleunigungs-, Geschwindigkeits- und Positionsvorgaben innerhalb der Toleranzen; keine Positionsänderung während des Stopps; kein beobachtbares instabiles Verhalten; kein unerwartetes Verhalten; kein Überschreiten durchgezogener Fahrstreifenmarkierungen.	Ruckartige Änderung des Kurswinkels nach Durchfahrt einer 180° Kurve, daher: Giereschleunigungsziel nicht eingehalten.

Literaturverzeichnis

Abrecht, S. et al.: Deep Learning Safety Concerns in Automated Driving Perception (2023)
Abrecht, Stephanie; Hirsch, Alexander; Raafatnia, Shervin; Woehrle, Matthias: Deep Learning Safety Concerns in Automated Driving Perception; https://arxiv.org/pdf/2309.03774.pdf, 2023, Zugriff: 25.10.2023

Amersbach, C.: Dissertation, Functional Decomposition Approach (2020)
Amersbach, Christian: Functional Decomposition Approach - Reducing the Safety Validation Effort for Highly Automated Driving, Dissertation Technische Universität Darmstadt, 2020

Amersbach, C.; Winner, H.: Functional Decomposition to Reduce Approval Effort (2017)
Amersbach, Christian; Winner, Hermann: Functional Decomposition: An Approach to Reduce the Approval Effort for Highly Automated Driving, in: 8. Tagung Fahrerassistenz, München, 2017

Amersbach, C.; Winner, H.: Test Coverage for Scenario-Based Validation (2019)
Amersbach, Christian; Winner, Hermann: Defining Required and Feasible Test Coverage for Scenario-Based Validation of Highly Automated Vehicles*, in: IEEE (Ed.): 2019 IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, New Zealand, IEEE, Piscataway, NJ, USA, 2019

Amersbach, C.; Winner, H.: Overcome the parameter space explosion (2019)
Amersbach, Christian; Winner, Hermann: Functional decomposition-A contribution to overcome the parameter space explosion during validation of highly automated driving, in: Traffic injury prevention 20:sup1, p. 52–57, 2019

ASAM e.V.: ASAM OpenDRIVE (2021)
ASAM e.V.: 1.7.0: ASAM OpenDRIVE, 2021

ASAM e.V.: ASAM Test Specification Study Group Report 2022 (2022)
ASAM e.V.: ASAM Test Specification Study Group Report 2022, 2022

ASAM e.V.: ASAM OpenSCENARIO® 2.0.0 (2022)
ASAM e.V.: 2.0.0: ASAM OpenSCENARIO® 2.0.0, 2022

ASAM e.V.: OpenODD: Concept Paper (2023)
ASAM e.V.: OpenODD: Concept Paper; https://www.asam.net/index.php?eID=dumpFile&t=f&f=4544&token=1260ce1c4f0afdbe18261f7137c689b1d9c27576, 2023, Zugriff: 08.11.2023

AUTOSAR: Overview of Functional Safety Measures in AUTOSAR (2019)
AUTOSAR: R19-11: Overview of Functional Safety Measures in AUTOSAR, 2019

AUTOSAR: Specification of Sensor Interfaces (2020)
AUTOSAR: R20-11: Specification of Sensor Interfaces, 2020

AUTOSAR: Requirements on Operating System (2022)
AUTOSAR: R22-11: Requirements on Operating System, 2022

Avizienis, A. et al.: Taxonomy of dependable and secure computing (2004)
Avizienis, A.; Laprie, J.-C.; Randell, B.; Landwehr, C.: Basic concepts and taxonomy of dependable and secure computing, in: IEEE Transactions on Dependable and Secure Computing (1), Issues1, pp. 1–37, 2004

Aydin, M.; Akbas, M. I.: Identification of Test Scenarios Using Accident Data (2021)
Aydin, Mert; Akbas, Mustafa I.: Identification of Test Scenarios for Autonomous Vehicles Using Fatal Accident Data, in: SAE International Journal of Connected and Automated Vehicles (1), Issues 4, 2021

Bachmann, F. et al.: Documenting Software Architecture: Documenting Behavior (2002)
Bachmann, Felix; Bass, Len; Clements, Paul; Garlan, David; Ivers, James; Little, Reed; Nord, Robert; Stafford, Judith: Documenting Software Architecture: Documenting Behavior, Pittsburgh, PA, 2002

Bachmann, F. et al.: Documenting Software Architecture: Documenting Interfaces (2002)
Bachmann, Felix; Bass, Len; Clements, Paul; Garlan, David; Ivers, James; Little, Reed; Nord, Robert; Stafford, Judith: Documenting Software Architecture: Documenting Interfaces, Pittsburgh, PA, 2002

Bagschik, G. et al.: Architecture Framework for Safe Automated Vehicles (2018)
Bagschik, Gerrit; Nolte, Marcus; Ernst, Susanne; Maurer, Markus: A System's Perspective Towards an Architecture Framework for Safe Automated Vehicles, in: 2018 21st International Conference on Intelligent Transportation Systems (ITSC), 2018

Bagschik, G. et al.: Wissensbasierte Szenariengenerierung (2018)
Bagschik, Gerrit; Menzel, Till; Körner, C.; Maurer, Markus: Wissensbasierte Szenariengenerierung für Betriebsszenarien auf deutschen Autobahnen, in: Uni-DAS e.V. (Ed.): 12. Workshop Fahrerassistenzsysteme und automatisiertes Fahren, Walting im Altmühltal, 2018

Bagschik, G.: Dissertation, Systematischer Einsatz von Szenarien (2022)
Bagschik, Gerrit: Systematischer Einsatz von Szenarien für die Absicherung automatisierter Fahrzeuge am Beispiel deutscher Autobahnen, Dissertation Technische Universität Braunschweig, 2022

Baldwin, C. Y.; Clark, K. B.: Managing in an age of modularity (2008)
Baldwin, Carliss Y.; Clark, Kim B.: Managing in an age of modularity, in: Harvard business review on manufacturing excellence at Toyota, Harvard Business School Press, Boston, Mass., 2008

Bandur, V. et al.: Making the Case for Centralized Automotive E/E Architectures (2021)
Bandur, Victor; Selim, Gehan; Pantelic, Vera; Lawford, Mark: Making the Case for Centralized Automotive E/E Architectures, in: IEEE Transactions on Vehicular Technology (2), Issues 70, pp. 1230–1245, 2021

Bass, L. et al.: Software architecture in practice (2013)
Bass, Len; Clements, Paul; Kazman, Rick: Software architecture in practice, SEI series in software engineering, 3. Edition, Addison-Wesley, Upper Saddle River, N.J., 2013

Bass, L. et al.: Software Architecture in Practice, 4th Edition (2021)
Bass, Len; Clements, Paul; Kazman, Rick: Software Architecture in Practice, 4th Edition, 1. Edition, Addison-Wesley Professional; Safari, Boston, MA, 2021

Batini, C.; Scannapieco, M.: Data and information quality (2016)
Batini, Carlo; Scannapieco, Monica: Data and information quality, Data-centric systems and applications, Springer, Switzerland, Cham (ZG), 2016

Baun, C.: Computernetze kompakt (2020)
Baun, Christian: Computernetze kompakt, IT kompakt, 5. Edition, Springer Vieweg, Berlin, Heidelberg, 2020

Baun, C.: Operating Systems (2020)
Baun, Christian: Operating Systems / Betriebssysteme, Lehrbuch, Springer, Wiesbaden, Heidelberg, 2020

Beck, H. N. et al.: Phänomen-Signal-Modell: Formalismus, Graph und Anwendung (31.07.21)
Beck, Hans N.; Salem, Nayel F.; Haber, Veronica; Rauschenbach, Matthias; Reich, Jan: Phänomen-Signal-Modell: Formalismus, Graph und Anwendung, 31.07.21

Bender, B.; Gericke, K.: Pahl/Beitz Konstruktionslehre (2021)
Bender, Beate; Gericke, Kilian: Pahl/Beitz Konstruktionslehre, 9. Edition, Springer Berlin Heidelberg, Berlin, Heidelberg, 2021

Bennaceur, A. et al.: Requirements Engineering (2019)
Bennaceur, Amel; Tun, Thein T.; Yu, Yijun; Nuseibeh, Bashar: Requirements Engineering, in: Cha, Sungdeok; Taylor, Richard N.; Kang, Kyochul (Eds.): Handbook of software engineering, Springer, Cham, 2019

Bernhart, W.; Alexander, M.: Computer on wheels (2020)
Bernhart, Wolfgang; Alexander, Michael: Computer on wheels; https://www.rolandberger.com/publications/publication_pdf/roland_berger_computer_on_wheels.pdf, 2020, Zugriff: 07.11.2023

Bhange, A. et al.: Kernelemente einer SW-Architektur für Cross-Domänen-Rechner (2023)
Bhange, Anand; Börner, Elmar; Jentges, Marco; Tasky, Tom: Kernelemente einer SW-Architektur für Cross-Domänen-Rechner, in: ATZelektronik (5), Issues 18, pp. 16–23, 2023

Bickel, J.: Masterthesis, Identifikation und Zuordnung von Einflussparametern (2020)
Bickel, Julia: Entwicklung einer Methode zur systematischen Identifikation, Diskretisierung und Zuordnung von Einflussparametern für die Validierung hochautomatisierter Fahrfunktionen, Masterthesis
TU Darmstadt, Darmstadt, 2020

Blödel, A.: Master Thesis, Verfahren zur Generierung von Trajektorien (2021)
Blödel, Alexander: Entwicklung und Implementierung eines Verfahrens zur Generierung von Trajektorien für das modulare Testen automatisierter Fahrfunktionen, Master Thesis
TU Darmstadt, Darmstadt, 2021

Börcsök, J.: Funktionale Sicherheit (2011)
Börcsök, Josef: Funktionale Sicherheit, Praxis, 3. Edition, VDE-Verl., Berlin, Offenbach, 2011

Borner, L.: Dissertation, Integrationstest (2010)
Borner, Lars: Integrationstest, Dissertation
Universität Heidelberg, Heidelberg, 2010

Broy, M.; Kuhrmann, M.: Einführung in die Softwaretechnik (2021)
Broy, Manfred; Kuhrmann, Marco: Einführung in die Softwaretechnik, Xpert.press, Springer Vieweg, Berlin, Heidelberg, 2021

BSI: PAS 1883:2020 (2020)
British Standards Institution: PAS 1883:2020, 2020

Buchholz, M. et al.: Automation of the UNICARagil vehicles (2020)
Buchholz, Michael; Gies, Fabian; Danzer, Andreas; Henning, Matti; Hermann, Charlotte; Herzog, Manuel; Horn, Markus; Schön, Markus; Rexin, Nils; Dietmayer, Klaus; Fernandez, Carlos; Janosovits, Johannes; Kamran, Danial; Kinzig, Christian; Lauer, Martin; Molinos, Eduardo; Stiller, Christoph; Ackermann, Stefan; Homolla, Tobias; Winner, Hermann; Gottschalg, Grischa; Leinen, Stefan; Becker, Matthias; Feiler, Johannes; Hoffmann, Simon; Diermeyer, Frank; Lampe, Bastian; Beemelmanns, Till; van Kempen, Raphael; Woopen, Timo; Eckstein, Lutz; Voget, Nicolai; Moormann, Dieter; Jatzkowski, Inga; Stolte, Torben; Maurer, Markus; Graf, Jürgen; Hinüber, Edgar L. von; Siepenkötter, Norbert: Automation of the UNICARagil vehicles, in: 29th Aachen Colloquium Sustainable Mobility 2020, Digital, 2020

Bundesrepublik Deutschland: Straßenverkehrsgesetz (StVG) (2023)
Bundesrepublik Deutschland Straßenverkehrsgesetz (StVG), 29.11.2023

Burcicki, D.: The E/E architecture and the future of automotive innovation (2022)
Burcicki, Doug: The E/E architecture and the future of automotive innovation; https://blogs.sw.siemens.com/ee-systems/2020/06/09/the-ee-architecture-and-the-future-of-automotive-innovation/, 2022, Zugriff: 07.11.2023

Burkacky, O. et al.: Rethinking car software and electronics architecture (2018)
Burkacky, Ondrej; Deichmann, Johannes; Doll, Georg; Knochenhauer, Christian: Rethinking car software and electronics architecture; https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/rethinking-car-software-and-electronics-architecture#/, 2018, Zugriff: 28.09.2023

Burton, S.: Entwicklungsprozess für funktional sichere Steuergeräte (2015)
Burton, Simon: Entwicklungsprozess für funktional sichere Steuergeräte; https://www.elektroniknet.de/automotive/software-tools/rundherum-sicher.121095.html, 2015, Zugriff: 22.11.2023

California Department of Motor Vehicles: Driver handbook (2020)
California Department of Motor Vehicles: Driver handbook, 2020

Carney, D.: New Mercedes self-driving system (2022)
Carney, Dan: This new Mercedes self-driving system lets you take your eyes off the road, in: Popular Science, 2022

Christian Rösener et al.: A Comprehensive Evaluation Approach for Highly Automated Driving (2017)
Christian Rösener; Jan Sauerbier; A. Zlocki; Felix Fahrenkrog; Lei Wang; A. Várhelyi; E. D. Gelder; Sandra Breunig; P. Mejuto; F. Tango: A Comprehensive Evaluation Approach for Highly Automated Driving, in: 25th Enhanced Safety of Vehicles 2017, Detroit, Michigan, U.S.A., 2017

Conrad, B.: Marktstart für Honda Legend Hybrid EX (2020)
Conrad, Bernd: Marktstart für Honda Legend Hybrid EX, in: Auto Motor und Sport, 2020

Czarnecki, K.: Operational Design Domain for Automated Driving Systems (2018)
Czarnecki, Krzysztof: Operational Design Domain for Automated Driving Systems - Taxonomy of Basic Terms, Waterloo Intelligent Systems Engineering (WISE) Lab, University of Waterloo, 2018

Dajsuren, Y. et al.: Formalizing correspondence rules for automotive architecture views (2014)
Dajsuren, Yanja; Gerpheide, Christine M.; Serebrenik, Alexander; Wijs, Anton; Vasilescu, Bogdan; van den Brand, Mark G.: Formalizing correspondence rules for automotive architecture views, in: Seinturier, Lionel; Bures, Tomas; McGregor, John D. (Eds.): Proceedings of the 10th international ACM Sigsoft conference on Quality of software architectures - QoSA '14, Marcq-en-Bareul, France, ACM Press, New York, New York, USA, 2014

Dajsuren, Y.: Defining Architecture Framework for Automotive Systems (2019)
Dajsuren, Yanja: Defining Architecture Framework for Automotive Systems, in: Dajsuren, Yanja; van den Brand, Mark (Eds.): Automotive systems and software engineering, Springer, Cham, 2019

Danquah, B. et al.: Statistical Validation Framework for Automotive Vehicle Simulations (2021)
Danquah, Benedikt; Riedmaier, Stefan; Meral, Yasin; Lienkamp, Markus: Statistical Validation Framework for Automotive Vehicle Simulations Using Uncertainty Learning, in: Applied Sciences (5), Issues 11, p. 1983, 2021

Despotou, G.; Kelly, T.: Argument modularity for safety assurance (2008)
Despotou, G.; Kelly, T.: Investigating the use of argument modularity to optimise through-life system safety assurance, in: The 3rd International Conference on System Safety, Birmingham, UK, IEEE, Stevanage, 2008

DIN: DIN VDE 31000-2 - Begriffe der Sicherheitstechnik (1987)
Deutsches Institut für Normen: 1987-12: DIN VDE 31000-2 - Begriffe der Sicherheitstechnik, 1987

Doran, D.: Supply chain implications of modularization (2003)
Doran, Desmond: Supply chain implications of modularization, in: International Journal of Operations & Production Management (3), Issues 23, pp. 316–326, 2003

Driessen, T.: Dissertation, Modularity by Design (2019)
Driessen, Thomas: Modularity by Design for Safety-Critical Software Systems, Dissertation
University of Augsburg, Augsburg, 2019

Ebert, C.: Systematisches Requirements Engineering (2014)
Ebert, Christof: Systematisches Requirements Engineering, 5. Edition, dpunkt.verlag, Heidelberg, 2014

Eckstein, L. et al.: UNICARagil news 4. Ausgabe (2021)
Eckstein, Lutz; Woopen, Timo; Kretschmer, Celine: UNICARagil news 4. Ausgabe; https://www.unicaragil.de/images/medien/news/Newsletter4_02-2021_DE.pdf, 2021, Zugriff: 06.09.2023

Europäische Kommission: Durchführungsverordnung (EU) 2022/1426 (2022)
Europäische Kommission Durchführungsverordnung (EU) 2022/1426, 05.08.2022

Fairbanks, G.: Just enough software architecture (2010)
Fairbanks, George: Just enough software architecture, Marshall & Brainerd, Boulder, CO, 2010

Farcas, C. et al.: Addressing the Integration Challenge (2010)
Farcas, C.; Farcas, E.; Krueger, I. H.; Menarini, M.: Addressing the Integration Challenge for Avionics and Automotive Systems—From Components to Rich Services, in: Proceedings of the IEEE (4), Issues 98, pp. 562–583, 2010

Faria, J. M.: Non-determinism and Failure Modes in Machine Learning (2017)
Faria, Jose M.: Non-determinism and Failure Modes in Machine Learning, in: 2017 IEEE 28th International Symposium on Software Reliability Engineering workshops, Toulouse, IEEE, Piscataway, NJ, 2017

Fowler, M.: Refactoring (2019)
Fowler, Martin: Refactoring, A Martin Fowler signature book, Addison-Wesley, Boston, Columbus, New York, San Francisco, Amsterdam, Cape Town, Dubai, London, Munich, 2019

Galbas, R.: VV-Methods comprehensive framework for AD safety assurance (2022)
Galbas, Roland: VV-Methods Towards a comprehensive framework for AD safety assurance, 2022 TRB Automated Road Transportation Symposium (ARTS), 2022

Gesmann-Nuissl, D.; Tacke, I.: Funktionale Sicherheit KI-basierter Systeme im Automobilsektor (2022)
Gesmann-Nuissl, Dagmar; Tacke, Ines: Funktionale Sicherheit KI-basierter Systeme im Automobilsektor, in: Uni-DAS e.V. (Ed.): 14. Workshop Fahrerassistenz und automatisiertes Fahren, Berkheim, 2022

Glatzki, F. et al.: Behavioral Attributes for a Behavior-Semantic Scenery Description (2021)
Glatzki, Felix; Lippert, Moritz; Winner, Hermann: Behavioral Attributes for a Behavior-Semantic Scenery Description (BSSD) for the Development of Automated Driving Functions, in: 2021 IEEE International Intelligent Transportation Systems Conference (ITSC), Indianapolis, IN, USA, IEEE, Piscataway, NJ, 2021

Göpfert, J.: Modulare Produktentwicklung (2009)
Göpfert, Jan: Modulare Produktentwicklung, Books on Demand, Norderstedt, 2009

Graubohm, R. et al.: Functional Safety Concept of Automated Driving (2019)
Graubohm, Robert; Stolte, Torben; Bagschik, Gerrit; Steimle, Markus; Maurer, Markus: Functional Safety Concept Generation within the Process of Preliminary Design of Automated Driving Functions at the Example of an Unmanned Protective Vehicle, in: Proceedings of the Design Society: International Conference on Engineering Design (1), Issues1, pp. 2863–2872, 2019

Greimel, H.: Honda's Level 3 system for automated driving has limits (2022)
Greimel, Hans: Honda's Level 3 system for automated driving has limits, in: Automotive News, 2022

Greis, F.: Level 3 im Stau: BMW erhält Zulassung für hochautomatisiertes Fahren - Golem.de (2023)
Greis, Friedhelm: Level 3 im Stau: BMW erhält Zulassung für hochautomatisiertes Fahren - Golem.de, in: Golem.de, 2023

Grube Doiz, N.: Bachelor Thesis, Modularisierung in der Automobilindustrie (2021)
Grube Doiz, Nerea: Potenzialanalyse der Modularisierung in der Automobilindustrie für die Freigabe automatisierter Fahrzeuge, Bachelor Thesis
Technische Universität Darmstadt, Darmstadt, 2021

Hackelsperger, M. et al.: Master Controller für das softwaredefinierte Fahrzeug (2022)
Hackelsperger, Markus; Reindl, Thomas; Mader, Ralph; Winkler, Gerd: Master Controller für das softwaredefinierte Fahrzeug, in: ATZelektronik 7-8, Issues 17, pp. 44–47, 2022

Haigh, F. D.: IDD - Interface Design Description (2020)
Haigh, Fred D.: IDD - Interface Design Description;
https://swehb.nasa.gov/display/SWEHBVC/IDD+-+Interface+Design+Description, 2020, Zugriff: 16.06.2023

Hakuli, S.; Krug, M.: Virtual Integration in the Development Process (2016)
Hakuli, Stephan; Krug, Markus: Virtual Integration in the Development Process of ADAS, in: Winner, Hermann (Ed.): Handbook of driver assistance systems, Springer International Publishing, Cham, 2016

Harel, D.; Rumpe, B.: Modeling Languages: Syntax, Semantics and All That Stuff (2000)
Harel, David; Rumpe, Bernhard: Modeling Languages: Syntax, Semantics and All That Stuff, Part I: The Basic Stuff, Technical Report MCS00-16, 2000

Harris, M.: NTSB Investigation Into Deadly Uber Self-Driving Car Crash (2019)
Harris, Mark: NTSB Investigation Into Deadly Uber Self-Driving Car Crash Reveals Lax Attitude Toward Safety, in: IEEE Spectrum, 2019

Heinze, L.: Steckbrief Forschungsprojekt VIVID (2020)
Heinze, Lars: Deutsch-japanische Forschungskooperation im automatisierten und vernetzten Fahren: Virtuelle Validierung; https://www.hs-kempten.de/fileadmin/Bildpool/News/PSB_CADJapanGermany_VIVID_final.pdf, 2020, Zugriff: 24.10.2023

Helper, S. et al.: Modularization and Outsourcing
Helper, Susan; Pil, Frits; Sako, Mari; Takeishi, Akira; Warburton, Max; MacDuffie, John: Modularization and Outsourcing: Implications for the Future of Automotive Assembly "Management of the Extended Enterprise" Research Team, Project Report to International Motor Vehicle Program

Henzel, M.: Dissertation, Generalisierbarkeit von maschinell gelernten Algorithmen (2019)
Henzel, Maren: Analyse der Generalisierbarkeit von maschinell gelernten Algorithmen in Fahrerassistenzsystemen, Dissertation
Technische Universität Darmstadt, 2019

Hildebrand, K. et al.: Daten- und Informationsqualität (2021)
Hildebrand, Knut; Gebauer, Marcus; Mielke, Michael (Eds.) Daten- und Informationsqualität, Springer eBook Collection, 5. Edition, Springer Vieweg, Wiesbaden, 2021

Holder, M. et al.: Challenges in Radar Sensor Modeling (2018)
Holder, Martin; Rosenberger, Philipp; Winner, Hermann; Dhondt, Thomas; Makkapati, Vamsi P.; Maier, Michael; Schreiber, Helmut; Magosi, Zoltan; Slavik, Zora; Bringmann, Oliver; Rosenstiel, Wolfgang: Measurements revealing Challenges in Radar Sensor Modeling for Virtual Validation of Autonomous Driving, in: 2018 IEEE Intelligent Transportation Systems Conference (ITSC), Maui, HI, IEEE, Piscataway, NJ, 2018

Homolla, T. et al.: Verfahren zur Korrektur von inkonsistenten Lokalisierungsdaten (2020)
Homolla, Tobias; Gottschalg, Grischa; Winner, Hermann: Verfahren zur Korrektur von inkonsistenten Lokalisierungsdaten in modularen technischen Systemen, in: Uni-DAS e.V. (Ed.): 13. Workshop Fahrerassistenz und automatisiertes Fahren, Digital, 2020

Homolla, T.: Dissertation, Gekapselte Trajektorienfolgeregelung für autonomes Fahren (2023)
Homolla, Tobias: Gekapselte Trajektorienfolgeregelung für autonomes Fahren, Dissertation
TU Darmstadt, 2023

Homolla, T.; Winner, H.: Encapsulated trajectory tracking control for autonomous vehicles (2022)
Homolla, Tobias; Winner, Hermann: Encapsulated trajectory tracking control for autonomous vehicles, in: Automotive and Engine Technology, pp. 1–12, 2022

Hood, C. et al.: Requirements Management (2008)
Hood, Colin; Fichtinger, Stefan; Pautz, Urte; Wiedemann, Simon: Requirements Management, Springer-Verlag Berlin Heidelberg, Berlin, Heidelberg, 2008

Horstmann, M.: Dissertation, Verflechtung von Test und Entwurf (2005)
Horstmann, Marc: Verflechtung von Test und Entwurf für eine verlässliche Entwicklung eingebetteter Systeme im Automobilbereich, Dissertation
Technische Universität Braunschweig, 2005

IEC: IEC 61508 (1997)
International Electrotechnical Commission: Version 4.0: IEC 61508, 1997

IEC: International Electrotechnical Vocabulary (2015)
International Electronical Commission: International Electrotechnical Vocabulary, 2015

IEEE: IEEE Std 1012 - Verification and Validation (2017)
IEEE Computer Society: IEEE Std 1012-2016: IEEE Standard for System, Software, and Hardware Verification and Validation, IEEE, Piscataway, NJ, USA, 2017

IEEE Computer Society: IEEE Std 1044 - Software Anomalies (2010)
IEEE Computer Society: IEEE Std 1044-2009: IEEE Std 1044 - Classification for Software Anomalies, IEEE, Piscataway, NJ, USA, 2010

INCOSE: Guide for Writing Requirements (2017)
International Council on Systems Engineering: 2.1: Guide for Writing Requirements, 2017

INCOSE: Systems Engineering Definition (2022)
International Council on Systems Engineering: Systems Engineering Definition; https://www.incose.org/about-systems-engineering/system-and-se-definition/systems-engineering-definition, 2022, Zugriff: 01.11.2022

ISO: ISO/IEC 7498 - Basic reference model (1994)
International Organization for Standardization: ISO/IEC 7498 - Information technology - Open Systems Interconnection - Basic reference model, 1994

ISO: ISO/IEC 25012 - Data quality model (2008)
International Organization for Standardization: ISO/IEC 25012 - Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Data quality model, 2008

ISO: ISO 26262 - Road vehicles – Functional safety (2018)
International Organization for Standardization: ISO 26262 - Road vehicles – Functional safety, 2018

ISO: ISO/PAS 21448: Safety of the intended functionality (2019)
International Organization for Standardization: ISO/PAS 21448: Road vehicles - Safety of the intended functionality, 2019

ISO: ISO/TR 4804 - Safety and cybersecurity for automated driving systems (2020)
International Organization for Standardization: ISO/TR 4804 - Road vehicles - Safety and cybersecurity for automated driving systems - Design, verification and validation, 2020

ISO: ISO 23150 - Data communication between sensors (2021)
International Organization for Standardization: ISO 23150 - Road Vehicles - Data communication between sensors and data fusion unit for automated driving functions, 2021

ISO: ISO/DIS 21448: Safety of the intended functionality (2022)
International Organization for Standardization: ISO/DIS 21448: Road vehicles - Safety of the intended functionality, 2022

ISO: ISO/FDIS 34503 - Specification for operational design domain (2023)
International Organization for Standardization: ISO/FDIS 34503 - Road Vehicles - Test scenarios for automated driving systems - Specification for operational design domain, 2023

Jatzkowski, I. et al.: Vehicle Operating Mode Management (2021)
Jatzkowski, Inga; Stolte, Torben; Graubohm, Robert; Maurer, Markus; Kampmann, Alexandru; Alrifaee, Bassam; Kowalewski, Stefan; Buchholz, Michael; Dietmayer, Klaus: Integration of a Vehicle Operating Mode Management into UNICARagil's Automotive Service-oriented Software Architecture, in: 30th Aachen Colloquium Sustainable Mobility 2021, Aachen, 2021

Jimeno Altelarrea, S. et al.: STPA enabled safety assessment of complex systems (2022)
Jimeno Altelarrea, Sergio; Riaz, Atif; Guenov, Marin D.: STPA enabled safety assessment in the architecting of complex systems, in: Safety and Reliability (4), Issues 41, pp. 197–224, 2022

Jungmayr, S.: Testbarkeitsanforderungen an die Software (2008)
Jungmayr, Stefan: Testbarkeitsanforderungen an die Software, in: Softwaretechnik-Trends (3), Issues 28, 2008

Junietz, P. et al.: Approaches to Adress Safety Validation of Automated Driving (2018)
Junietz, Philipp; Wachenfeld, Walther; Klonecki, Kamil; Winner, Hermann: Evaluation of Different Approaches to Address Safety Validation of Automated Driving, in: 2018 21st International Conference on Intelligent Transportation Systems (ITSC), pp. 491–496, 2018

Junietz, P. M.: Dissertation, Microscopic and Macroscopic Risk Metrics (2019)
Junietz, Philipp M.: Microscopic and Macroscopic Risk Metrics for the Safety Validation of Automated Driving, Dissertation
TU Darmstadt, Darmstadt, 2019

Kalra, N.; Paddock, S. M.: Driving to safety (2016)
Kalra, Nidhi; Paddock, Susan M.: Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?, in: Transportation Research Part A: Policy and Practice, Issues 94, pp. 182–193, 2016

Kampmann, A. et al.: Dynamic Service-Oriented Software Architecture (2019)
Kampmann, Alexandru; Alrifaee, Bassam; Kohout, Markus; Wustenberg, Andreas; Woopen, Timo; Nolte, Marcus; Eckstein, Lutz; Kowalewski, Stefan: A Dynamic Service-Oriented Software Architecture for Highly Automated Vehicles, in: IEEE (Ed.): 2019 IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, New Zealand, IEEE, Piscataway, NJ, USA, 2019

Kelly, T.; Weaver, R.: The Goal Structuring Notation (2004)
Kelly, Tim; Weaver, Rob: The Goal Structuring Notation – A Safety Argument Notation, in: Proceedings of the dependable systems and networks, 2004

Keuth, H.: Karl Poppers „Logik der Forschung“ (2019)
Keuth, Herbert: Karl Poppers „Logik der Forschung“, in: Franco, Giuseppe (Ed.): Handbuch Karl Popper, Springer Reference, Springer VS, Wiesbaden, Germany, 2019

Kiureghian, A. D.; Ditlevsen, O.: Aleatory or epistemic? (2009)
Kiureghian, Armen D.; Ditlevsen, Ove: Aleatory or epistemic? Does it matter?, in: Structural Safety (2), Issues 31, pp. 105–112, 2009

Klamann, B. et al.: Pass-/Fail-Criteria for Particular Tests (2019)
Klamann, Björn; Lippert, Moritz; Amersbach, Christian; Winner, Hermann: Defining Pass-/Fail-Criteria for Particular Tests of Automated Driving Functions, in: 2019 IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, NZ, 2019

Klamann, B.; Winner, H.: Comparing Different Levels of Technical Systems (2021)
Klamann, Björn; Winner, Hermann: Comparing Different Levels of Technical Systems for a Modular Safety Approval—Why the State of the Art Does Not Dispense with System Tests Yet, in: Energies (22), Issues 14, 2021

Klamann, B.; Winner, H.: Analyse von Dekompositionsprozessen zur modularen Absicherung (2022)
Klamann, Björn; Winner, Hermann: Analyse von Dekompositionsprozessen zur Umsetzung einer modularen Absicherung automatisierter Fahrzeuge, in: Uni-DAS e.V. (Ed.): 14. Workshop Fahrerassistenz und automatisiertes Fahren, Berkheim, 2022

Klamann, B.; Winner, H.: Introducing the detailed semantic interface description (2023)
Klamann, Björn; Winner, Hermann: Introducing the detailed semantic interface description to support a modular safety approval of automated vehicles – S^2I^2, in: Safety and Reliability, pp. 1–40, 2023

Kocerka, J. et al.: Analysing Quality of Textual Requirements (2018)
Kocerka, Jerzy; Krzeslak, Michal; Galuszka, Adam: Analysing Quality of Textual Requirements Using Natural Language Processing: A Literature Review, in: 2018 23rd International Conference on Methods & Models in Automation & Robotics (MMAR), Miedzyzdroje, IEEE, Piscataway, New Jersey, 2018

Kohls, S. K.: Master Thesis, Abhängigkeiten zwischen Modulen (2022)
Kohls, Svenja K.: Entwicklung eines Verfahrens zur Modellierung von Abhängigkeiten zwischen Modulen für die modulare Absicherung automatisierter Fahrfunktionen, Master Thesis
TU Darmstadt, Darmstadt, 2022

Koopman, P. et al.: A Safety Standard Approach for Fully Autonomous Vehicles (2019)
Koopman, Philip; Ferrell, Uma; Fratrik, Frank; Wagner, Michael: A Safety Standard Approach for Fully Autonomous Vehicles, in: Romanovsky, Alexander et al. (Eds.): Computer Safety, Reliability, and Security, Springer eBooks Computer Science Nr. 11699, 1. Edition, Springer, Cham, 2019

Koopman, P.: How safe is safe enough? (2022)
Koopman, Philip: How safe is safe enough?, Carnegie Mellon University, Pittsburgh, Pennsylvania, 2022

Krause, J.: Bachelor Thesis, Tests für die Freigabe des Dynamikmoduls (2021)
Krause, Johannes: Entwicklung von Tests für die Freigabe des Dynamikmoduls als Teil der Absicherung eines automatisierten Fahrzeugs, Bachelor Thesis
TU Darmstadt, Darmstadt, 2021

Kröger, W.; Ayoub, A.: Towards «type approval» of automated vehicles (2022)
Kröger, Wolfgang; Ayoub, Ali: Towards «type approval» of automated vehicles: Means of safety validation and simulation-based methods, in particular, in: Uni-DAS e.V. (Ed.): 14. Workshop Fahrerassistenz und automatisiertes Fahren, Berkheim, 2022

Kröger, W.; Nan, C.: Interdependencies of Complex Technical Networks (2014)
Kröger, Wolfgang; Nan, Cen: Addressing Interdependencies of Complex Technical Networks, in: D'Agostino, Gregorio; Scala, Antonio (Eds.): Networks of Networks: The Last Frontier of Complexity, Understanding Complex Systems, Springer International Publishing, Cham, 2014

Kröger, W.; Nan, C.: Dealing with complexity (2019)
Kröger, Wolfgang; Nan, Cen: Dealing with complexity, in: Büscher, Christian; Schippl, Jens; Sumpf, Patrick (Eds.): Energy as a sociotechnical problem, Earthscan from Routledge, Routledge, Abingdon, Oxon, New York, NY, 2019

La Rocco, N.: BMW 5er: Kraftfahrtbundesamt erteilt Genehmigungen für Level 2+ (2023)
La Rocco, Nicolas: BMW 5er: Kraftfahrtbundesamt erteilt Genehmigungen für Level 2+, in: ComputerBase, 2023

La Rocco, N.: BlueCruise: Ford bringt freihändiges Level 2+ nach Deutschland (2023)
La Rocco, Nicolas: BlueCruise: Ford bringt freihändiges Level 2+ nach Deutschland, in: ComputerBase, 2023

Lackes, R.; Siepermann, M.: Gabler Wirtschaftslexikon (2018)
Lackes, Richard; Siepermann, Markus: Gabler Wirtschaftslexikon; https://wirtschaftslexikon.gabler.de/definition/modul-40077/version-263472, 2018, Zugriff: 20.11.2022

Larses, O.: Factors influencing dependable modular architectures (2005)
Larses, Ola: Factors influencing dependable modular architectures for automotive applications, Stockholm, 2005

Lazard; Roland Berger: Global Automotive Supplier Study 2018 (2017)
Lazard; Roland Berger: Global Automotive Supplier Study 2018; https://www.rolandberger.com/publications/publication_pdf/roland_berger_global_automotive_supplier_study_2018.pdf, 2017, Zugriff: 28.09.2023

Leitner, A.: ENABLE-S3: Project Introduction (2020)
Leitner, Andrea: ENABLE-S3: Project Introduction, in: Leitner, Andrea; Watzenig, Daniel; Ibanez-Guzman, Javier (Eds.): Validation and Verification of Automated Systems: Results of the ENABLE-S3 Project, Springer International Publishing, Cham, 2020

Leveson, N.: Engineering a safer world (2012)
Leveson, Nancy: Engineering a safer world, Engineering systems, MIT Press, Cambridge, Mass, 2012

Leveson, N. G.; Thomas, J. P.: STPA Handbook (2018)
Leveson, Nancy G.; Thomas, John P.: STPA Handbook; http://psas.scripts.mit.edu/home/get_file.php?name=STPA_handbook.pdf, 2018, Zugriff: 16.06.2019

Li, H. et al.: Simulation with Vehicle-in-the-Loop Testbed (2023)
Li, Hexuan; Makkapati, Vamsi P.; Wan, Li; Tomasch, Ernst; Hoschopf, Heinz; Eichberger, Arno: Validation of Automated Driving Function Based on the Apollo Platform: A Milestone for Simulation with Vehicle-in-the-Loop Testbed, in: Vehicles (2), Issues 5, pp. 718–731, 2023

Lippert, M. et al.: Behavior-Semantic Scenery Description (2024)
Lippert, Moritz; Glatzki, Felix; Winner, Hermann: Behavior-Semantic Scenery Description (BSSD) of Road Networks for Automated Driving, in: IEEE Access, Issues 12, pp. 43039–43052, 2024

Lipsmeier, A. et al.: Mechatronic Modularization of Intelligent Technical Systems (2018)
Lipsmeier, A.; Westermann, T.; Anacker, H.; Dumitrescu, R.: Mechatronic Modularization of Intelligent Technical Systems, in: Maier, Anja et al. (Eds.): Product, services and systems design, DS 87, 3, Curran Associates Inc, Red Hook, NY, 2018

Lotz, F. G.: Dissertation, Referenzarchitektur für automatisierte Fahrzeugführung (2017)
Lotz, Felix G. O.: Eine Referenzarchitektur für die assistierte und automatisierte Fahrzeugführung mit Fahrereinbindung, Dissertation
Technische Universität Darmstadt, Darmstadt, 2017

Luckcuck, M. et al.: Formal Specification and Verification of Autonomous Robotic Systems (2020)
Luckcuck, Matt; Farrell, Marie; Dennis, Louise A.; Dixon, Clare; Fisher, Michael: Formal Specification and Verification of Autonomous Robotic Systems, in: ACM Computing Surveys (5), Issues 52, pp. 1–41, 2020

Martens, T. et al.: UNICARagil Dynamics Module (2020)
Martens, Timm; Pouansi Majiade, Lionel B.; Henkel, Niclas; Eckstein, Lutz; Wielgos, Sebastian; Schlupek, Martin: UNICARagil Dynamics Module, in: 29th Aachen Colloquium Sustainable Mobility 2020, Digital, 2020

Maurer, M.: Complexity management in engineering design (2017)
Maurer, Maik: Complexity management in engineering design - a primer, Springer Berlin Heidelberg, Berlin, Heidelberg, s.l., 2017

Maurer, M.: Das inhärente Risiko autonomer Straßenfahrzeuge (2018)
Maurer, Markus: Das inhärente Risiko autonomer Straßenfahrzeuge, in: 1. Regelungstechnisches Kolloquim, Braunschweig, Deutschland, 2018

Mazzega, J. et al.: Pegasus Method (2019)
Mazzega, Jens; Lipinski, Daniel; Eberle, Ulrich; Schittenhelm, Helmut; Wachenfeld, Walther: Pegasus Method, 2019

McGregor, J. D.; Sykes, D. A.: Testing object-oriented software (2001)
McGregor, John D.; Sykes, David A.: A practical guide to testing object-oriented software, Addison-Wesley object technology series, Addison-Wesley, Boston, Ma, 2001

Mercedes-Benz Group AG: Introducing DRIVE PILOT (2023)
Mercedes-Benz Group AG: Introducing DRIVE PILOT: An Automated Driving System for the Highway, 2023

Mokhtarian, A. et al.: Dynamic Service-oriented Software Architecture (2020)
Mokhtarian, Armin; Kampmann, Alexandru; Alrifaee, Bassam; Kowalewski, Stefan: The Dynamic Service-oriented Software Architecture for the UNICARagil Project, in: 29th Aachen Colloquium Sustainable Mobility 2020, Digital, 2020

Mondragon, C. C. et al.: Managing technology for complex systems (2009)
Mondragon, Christian C.; Mondragon, Adrian C.; Miller, Roger; Mondragon, Etienne C.: Managing technology for highly complex critical modular systems: The case of automotive by-wire systems, in: International Journal of Production Economics (2), Issues 118, pp. 473–485, 2009

Mori, K. T. et al.: The Inadequacy of Discrete Scenarios (2022)
Mori, Ken T.; Liang, Xu; Elster, Lukas; Peters, Steven: The Inadequacy of Discrete Scenarios in Assessing Deep Neural Networks, in: IEEE Access, Issues 10, pp. 118236–118242, 2022

Nalic, D. et al.: Stress Testing Method for Scenario-Based Testing of Automated Driving Systems (2020)
Nalic, Demin; Li, Hexuan; Eichberger, Arno; Wellershaus, Christoph; Pandurevic, Aleksa; Rogic, Branko: Stress Testing Method for Scenario-Based Testing of Automated Driving Systems, in: IEEE Access, Issues 8, pp. 224974–224984, 2020

Nalic, D. et al.: Software Framework for Testing of ADS (2021)
Nalic, Demin; Pandurevic, Aleksa; Eichberger, Arno; Fellendorf, Martin; Rogic, Branko: Software Framework for Testing of Automated Driving Systems in the Traffic Environment of Vissim, in: Energies (11), Issues 14, p. 3135, 2021

Nalic, D. et al.: Criticality Assessment Method for Automated Driving Systems (2023)
Nalic, Demin; Mihalj, Tomsilac; Orucevic, Faris; Schabauer, Martin; Lex, Cornelia; Sinz, Wolfgang Eichberger, Arno: Criticality Assessment Method for Automated Driving Systems by Introducing Fictive Vehicles and Variable Criticality Thresholds, in: Huang, Chao et al. (Eds.): Advanced Sensing and Safety Control for Connected and Automated Vehicles, MDPI - Multidisciplinary Digital Publishing Institute, Basel, 2023

NASA: Expanded Guidance for NASA Systems Engineering (2016)
National Aeronautics and Space Administration: Expanded Guidance for NASA Systems Engineering, 1. Edition, Washington, D.C., 2016

Nielsen, C. B. et al.: Systems of Systems Engineering (2015)
Nielsen, Claus B.; Larsen, Peter G.; Fitzgerald, John; Woodcock, Jim; Peleska, Jan: Systems of Systems Engineering, in: ACM Computing Surveys (2), Issues 48, pp. 1–41, 2015

Nolte, M. et al.: Skill- And Ability-Based Development Process (2017)
Nolte, Marcus; Bagschik, Gerrit; Jatzkowski, Inga; Stolte, Torben; Reschka, Andreas; Maurer, Markus: Towards a skill- and ability-based development process for self-aware automated road vehicles, in: 2017 IEEE Intelligent Transportation Systems Conference (ITSC), Yokohama, IEEE, Piscataway, NJ, 2017

Nolte, M. et al.: Summary and Taxonomy of Self-Representation Concepts (2020)
Nolte, Marcus; Jatzkowski, Inga; Ernst, Susanne; Maurer, Markus: Supporting Safe Decision Making Through Holistic System-Level Representations & Monitoring - A Summary and Taxonomy of Self-Representation Concepts for Automated Vehicles; http://arxiv.org/pdf/2007.13807v2, 2020, Zugriff: 02.05.2024

Object Management Group: Interface Definition Language (2018)
Object Management Group: Interface Definition Language, 2018

Patriarca, R. et al.: Past and present of System-Theoretic Accident Model And Processes (2022)
Patriarca, Riccardo; Chatzimichailidou, Mikela; Karanikas, Nektarios; Di Gravio, Giulio: The past and present of System-Theoretic Accident Model And Processes (STAMP) and its associated techniques: A scoping review, in: Safety Science, Issues 146, p. 105566, 2022

PEGASUS Projekt: PEGASUS Sicherheitsargumentation (2018)
PEGASUS Projekt: PEGASUS Sicherheitsargumentation, 2018

PEGASUS Projekt: PEGASUS Abschlussbericht (2020)
PEGASUS Projekt: PEGASUS Abschlussbericht, 2020

Pek, C. et al.: Using online verification to prevent autonomous vehicles from causing accidents (2020)
Pek, Christian; Manzinger, Stefanie; Koschi, Markus; Althoff, Matthias: Using online verification to prevent autonomous vehicles from causing accidents, in: Nature Machine Intelligence (9), Issues 2, pp. 518–528, 2020

Philion, J. et al.: Evaluate Perception Models Using Planner-Centric Metrics (2020)
Philion, Jonah; Kar, Amlan; Fidler, Sanja: Learning to Evaluate Perception Models Using Planner-Centric Metrics, in: Mortensen, Eric; Masson-Forsythe, Margaux (Eds.): 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, IEEE, Piscataway, NJ, 2020

Pimmler, T. U.; Eppinger, S. D.: Integration Analysis of Product Decompositions (1994)
Pimmler, Thomas U.; Eppinger, Steven D.: Integration Analysis of Product Decompositions, in: Hight, T. K. (Ed.): Design theory and methodology, DTM '94, Minneapolis, Minnesota, USA, DE Nr. 68, American Soc. of Mechanical Engineers, New York, NY, 1994

Popovic, V. M.; Vasic, B. M.: Review of hazard analysis methods (2008)
Popovic, Vladimir M.; Vasic, Branko M.: Review of hazard analysis methods and their basic characteristics, in: FME Transactions (4), Issues 36, pp. 181–187, 2008

Rasmussen, J.: Distinctions in Human Performance Models (1983)
Rasmussen, Jens: Skills, Rules, and Knowledge; Signals, Signs, and Symbols, and Other Distinctions in Human Performance Models, in: IEEE Transactions on Systems, Man, and Cybernetics (3)SMC-13, pp. 257–266, 1983

Reich, J.: Assurance Argumentation Framework (2023)
Reich, Jan: Assurance Argumentation Framework, VVM Final Event Project Results Day, Stuttgart, 2023

Reich, J.; Nolte, M.: VVM Assurance Argumentation (2022)
Reich, Jan; Nolte, Marcus: VVM Assurance Argumentation, Mid-Term Presentation VVM, 2022

Reschka, A. et al.: Ability and skill graphs for system modeling (2015)
Reschka, Andreas; Bagschik, Gerrit; Ulbrich, Simon; Nolte, Marcus; Maurer, Markus: Ability and skill graphs for system modeling, online monitoring, and decision support for vehicle guidance systems, in: 2015 IEEE Intelligent Vehicles Symposium (IV 2015), Seoul, South Korea, IEEE, Piscataway, NJ, 2015

Reschka, A.: Dissertation, Fertigkeiten- und Fähigkeitengraphen von automatisierten Fahrzeugen (2017)
Reschka, Andreas: Fertigkeiten- und Fähigkeitengraphen als Grundlage des sicheren Betriebs von automatisierten Fahrzeugen im öffentlichen Straßenverkehr in städtischer Umgebung, Dissertation
Technische Universität Braunschweig, Braunschweig, 2017

Rinaldi, S. M. et al.: Identifying critical infrastructure interdependencies (2001)
Rinaldi, S. M.; Peerenboom, J. P.; Kelly, T. K.: Identifying, understanding, and analyzing critical infrastructure interdependencies, in: IEEE Control Systems Magazine (6), Issues 21, pp. 11–25, 2001

Rosenberger, P. et al.: Functional Decomposition of Lidar Sensor Systems (2020)
Rosenberger, Philipp; Holder, Martin; Zofka, Marc R.; Fleck, Tobias; D'hondt, Thomas; Wassermann, Benjamin; Prstek, Juraj: Functional Decomposition of Lidar Sensor Systems for Model Development, in: Leitner, Andrea; Watzenig, Daniel; Ibanez-Guzman, Javier (Eds.): Validation and Verification of Automated Systems: Results of the ENABLE-S3 Project, Springer International Publishing, Cham, 2020

Rumez, M. et al.: Automotive Service-Oriented Architectures (2020)
Rumez, Marcel; Grimm, Daniel; Kriesten, Reiner; Sax, Eric: An Overview of Automotive Service-Oriented Architectures and Implications for Security Countermeasures, in: IEEE Access, Issues 8, pp. 221852–221870, 2020

Rushby, J.; Miner, P. S.: Modular Certification (2002)
Rushby, John; Miner, Paul S.: Modular Certification, in: NASA Contractor Report CR-2002-212130, 2002

SAE International: Best Practice for Describing an ODD (2020)
SAE International: AVSC Best Practice for Describing an Operational Design Domain: Conceptual Framework and Lexicon; http://go.sae.org/rs/525-RCG-129/images/AVSC00002202004.pdf, 2020, Zugriff: 02.05.2024

SAE International: SAE J3016 (2021)
SAE International: SAE J3016, SAE International, Warrendale, Pa., 2021

Sanchez, R.: Integrating technology strategy and marketing strategy
Sanchez, Ron: Integrating technology strategy and marketing strategy, in: Thomas, H.; O'Neal, D. (Eds.): Strategic Integration, John Wiley & Sons, Ltd, Chichester, West Sussex

Sanchez, R.: Building real modularity competence (2013)
Sanchez, Ron: Building real modularity competence in automotive design, development, production, and after-service, in: International Journal of Automotive Technology and Management (3), Issues 13, pp. 204–236, 2013

Schachner, M. et al.: Observation-Based Pedestrian Scenario Extraction (2023)
Schachner, Martin; Kirillova, Nadezda; Weißenbacher, Fabian; Schneider, Bernd; Possegger, Horst; Eichberger, Arno; Ferenc Magosi, Zoltan; Dobberstein, Jan; Lich, Thomas; Kirchengast, Martin; Hennecke, Marcus; Klug, Corina: Observation-Based Pedestrian Scenario Extraction For Virtual Testing, in: 27th International Technical Conference on the Enhanced Safety of Vehicles (ESV), Yokohama, Japan, 03.-06.04.2023

Schäuffele, J. et al.: Automotive software engineering (2016)
Schäuffele, Jörg; Zurawka, Thomas; Carey, Roger: Automotive software engineering, SAE International Automotive, 2. Edition, SAE International, Warrendale, PA, 2016

Schäuffele, J.; Zurawka, T.: Automotive Software Engineering (2016)
Schäuffele, Jörg; Zurawka, Thomas: Automotive Software Engineering, ATZ/MTZ-Fachbuch Ser, 6. Edition, Springer Fachmedien Wiesbaden GmbH, Wiesbaden, 2016

Schlatow, J. et al.: Self-awareness in autonomous automotive systems (2017)
Schlatow, Johannes; Moostl, Mischa; Ernst, Rolf; Nolte, Marcus; Jatzkowski, Inga; Maurer, Markus; Herber, Christian; Herkersdorf, Andreas: Self-awareness in autonomous automotive systems, in: Proceedings of the 2017 Design, Automation & Test in Europe (DATE), Lausanne, Switzerland, IEEE, Piscataway, NJ, 2017

Schoeneberg, K.-P.: Komplexitätsmanagement in Unternehmen (2014)
Schoeneberg, Klaus-Peter (Ed.) Komplexitätsmanagement in Unternehmen, Springer Gabler, Wiesbaden, 2014

Scholtes, M. et al.: 6-Layer Model for a Description of Urban Environment (2021)
Scholtes, Maike; Westhofen, Lukas; Turner, Lara R.; Lotto, Katrin; Schuldes, Michael; Weber, Hendrik; Wagener, Nicolas; Neurohr, Christian; Bollmann, Martin H.; Kortke, Franziska; Hiller, Johannes; Hoss, Michael; Bock, Julian; Eckstein, Lutz: 6-Layer Model for a Structured Description and Categorization of Urban Traffic and Environment, in: IEEE Access, Issues 9, pp. 59131–59147, 2021

Schöner, H.-P.; Antona-Makoshi, J.: Testing for Tactical Safety of Autonomous Vehicles (2021)
Schöner, Hans-Peter; Antona-Makoshi, Jacobo: Testing for Tactical Safety of Autonomous Vehicles, in: 30th Aachen Colloquium Sustainable Mobility 2021, Aachen, 2021

Schuh, G.; Riesener, M.: Produktkomplexität managen (2018)
Schuh, Günther; Riesener, Michael: Produktkomplexität managen, Hanser eLibrary, 3. Edition, Hanser, München, 2018

Schuldt, F. et al.: Systematische Testgenerierung für Fahrerassistenzsysteme (2013)
Schuldt, Fabian; Saust, Falko; Lichte, Bernd; Maurer, Markus; Scholz, Stephan: Effiziente systematische Testgenerierung für Fahrerassistenzsysteme in virtuellen Umgebungen, in: AAET 2013 - Automatisierungssysteme, Assistenzsysteme und eingebettete Systeme für Transportmittel, 2013

Schuldt, F. et al.: Systematic Test Case Generation for ADAS (2018)
Schuldt, Fabian; Reschka, Andreas; Maurer, Markus: A Method for an Efficient, Systematic Test Case Generation for Advanced Driver Assistance Systems in Virtual Environments, in: Winner, Hermann; Prokop, Günther; Maurer, Markus (Eds.): Automotive Systems Engineering II, Springer International Publishing, Cham, 2018

SCSC: Goal Structuring Notation Community Standard (2021)
Safety-Critical Systems Club: Version 3: Goal Structuring Notation (GSN) Community Standard, 2021

SET Level Projekt: Das Projekt hinter SET Level (2022)
SET Level Projekt: Das Projekt hinter SET Level; https://setlevel.de/projekt, 2022, Zugriff: 15.08.2023

Shalev-Shwartz, S. et al.: On a Formal Model of Safe and Scalable Self-driving Cars (21.08.17)
Shalev-Shwartz, Shai; Shammah, Shaked; Shashua, Amnon: On a Formal Model of Safe and Scalable Self-driving Cars, 21.08.17

Sharvia, S.: Dissertation, Integrated Application of Compositional and Behavioural Safety Analysis (2011)
Sharvia, Septavera: Integrated Application of Compositional and Behavioural Safety Analysis, Dissertation
The University of Hull, 2011

Sharvia, S.; Papadopoulos, Y.: Compositional and Behavioural Safety Analysis (2011)
Sharvia, Septavera; Papadopoulos, Yiannis: Integrated Application of Compositional and Behavioural Safety Analysis, in: Zamojski, Wojciech (Ed.): Dependable computer systems, Advances in Intelligent and Soft Computing Nr. 97, Springer, Berlin, Heidelberg, 2011

Sheard, S. A.: Three Types of Systems Engineering Implementation (2000)
Sheard, Sarah A.: Three Types of Systems Engineering Implementation, in: Tenth International Symposium of the International Council on Systems Engineering, Minneapolis, 16-20.07.2000

Shevchenko, N.: Introduction to Model-Based Systems Engineering (2020)
Shevchenko, Nataliya: An Introduction to Model-Based Systems Engineering (MBSE); http://insights.sei.cmu.edu/blog/introduction-model-based-systems-engineering-mbse/, 2020, Zugriff: 22.03.2022

Smits, L.: Master Thesis, Testfälle für die Verhaltens- und Trajektorienplanung (2022)
Smits, Leonard: Entwicklung einer Methode zur Generierung und Durchführung von Testfällen für die Verhaltens- und Trajektorienplanung automatisierter Fahrzeuge, Master Thesis
Technische Universität Darmstadt, Darmstadt, 2022

Spillner, A.; Linz, T.: Basiswissen Softwaretest (2019)
Spillner, Andreas; Linz, Tilo: Basiswissen Softwaretest, 6. Edition, dpunkt.verlag, Heidelberg, 2019

SRA: Society for Risk Analysis Glossary (2018)
Society for Risk Analysis: Society for Risk Analysis Glossary, 2018

Stahl, T. N.: Dissertation, Safeguarding via online verification (2022)
Stahl, Tim N.: Safeguarding complex and learning-based automated driving functions via online verification, Dissertation
Technische Universität München, München, 2022

Steimle, M. et al.: Terminology for Scenario-Based Development and Test
Steimle, Markus; Menzel, Till; Maurer, Markus: Toward a Consistent Taxonomy for Scenario-Based Development and Test Approaches for Automated Vehicles: A Proposal for a Structuring Framework, a Basic Vocabulary, and Its Application, in: IEEE Access, Issues 9, pp. 147828–147854

Stolte, T. et al.: Hazard analysis and risk assessment for automated vehicles (2017)
Stolte, Torben; Bagschik, Gerrit; Reschka, Andreas; Maurer, Markus: Hazard analysis and risk assessment for an automated unmanned protective vehicle, in: 2017 IEEE Intelligent Vehicles Symposium (IV), 2017

Stolte, T. et al.: Taxonomy to Unify Fault Tolerance Regimes (2021)
Stolte, Torben; Ackermann, Stefan; Graubohm, Robert; Jatzkowski, Inga; Klamann, Björn; Winner, Hermann; Maurer, Markus: A Taxonomy to Unify Fault Tolerance Regimes for Automotive Systems: Defining Fail-Operational, Fail-Degraded, and Fail-Safe, in: IEEE Transactions on Intelligent Vehicles, p. 1, 2021

Tamanaka, G. T. et al.: Fault-tolerant architecture using containers (2022)
Tamanaka, Gustavo T. B.; Aroca, Rafael V.; Paula Caurin, Glauco A. de: Fault-tolerant architecture and implementation of a distributed control system using containers, in: 2022 Latin American Robotics Symposium, 2022 Brazilian Symposium on Robotics, and 2022 Workshop on Robotics in Education, São Bernardo do Campo, Brazil, IEEE, Piscataway, NJ, 2022

Těsanović, A. et al.: Modular Verification of Reconfigurable Components (2005)
Těsanović, Aleksandra; Nadjm-Tehrani, Simin; Hansson, Jörgen: Modular Verification of Reconfigurable Components, in: Hutchison, David et al. (Eds.): Component-Based Software Development for Embedded Systems: An Overview of Current Research Trends Nr. 3778, Scholars Portal, Berlin, Heidelberg, 2005

TestSPICE SIG: TestSPICE Process Assessment Model (2018)
TestSPICE SIG: TestSPICE Process Assessment Model; https://intacs.info/index.php/old-spice-center/spice-models/testspice-1, 2018, Zugriff: 19.07.2022

The Open Group: Service-Oriented Architecture (SOA) (2007)
The Open Group: Service-Oriented Architecture (SOA); https://pubs.opengroup.org/onlinepubs/7699929599/toc.pdf, 2007, Zugriff: 20.11.2022

The Open Group: Developing Architecture Views (2022)
The Open Group: Developing Architecture Views, in: The Open Group (Ed.): The Open Group Architecture Framework (TOGAF) 10th Edition, 25.04.2022

UK MoD: Defence Standard 00-56 - Safety Management Requirements (2017)
United Kingdom Ministry of Defense: issue 4: Defence Standard 00-56, 2017

UL: ANSI/UL 4600 - Voting Version
Underwriters' Laboratories: ANSI/UL 4600 - Voting Version

Ulbrich, S. et al.: Defining Scene, Situation, and Scenario (2015)
Ulbrich, Simon; Menzel, Till; Reschka, Andreas; Schuldt, Fabian; Maurer, Markus: Defining and Substantiating the Terms Scene, Situation, and Scenario for Automated Driving, in: IEEE (Ed.): 2015 IEEE 15th Conference on Intelligent Transportation Systems, Gran Canaria, Spain, 2015

Ulbrich, S. et al.: Definition Szene, Situation und Szenario (2015)
Ulbrich, Simon; Menzel, Till; Reschka, Andreas; Schuldt, Fabian; Maurer, Markus: Definition der Begriffe Szene, Situation und Szenario für das automatisierte Fahren, in: Uni-DAS e.V. (Ed.): 10. Workshop Fahrerassistenzsysteme und automatisiertes Fahren, Walting im Altmühltal, 2015

UNECE: UN-Regulation No. 152 - Approval of the Advanced Emergency Braking System (AEBS) (2020)
United Nations Economic Commission for Europe UN Regulation No. 152 - Uniform provisions concerning the approval of motor vehicles with regard to the Advanced Emergency Braking System (AEBS) for M1 and N1 vehicles; 3, 04.11.2020

UNECE: UN Regulation No. 156 - Provisions concerning software updates (2021)
United Nations Economic Commission for Europe UN Regulation No. 156 - Uniform provisions concerning the approval of vehicles with regards to software update and software updates management system, 22.01.2021

UNECE: UN-Regelung Nr. 157 - Genehmigung automatischer Spurhalteassistenzsysteme (2022)
United Nations Economic Commission for Europe UN-Regelung Nr. 157 — Einheitliche Bedingungen für die Genehmigung von Fahrzeugen hinsichtlich des automatischen Spurhalteassistenzsystems (ALKS) [2021/389], 30.05.2022

van der Aalst, W.: Six Levels of Autonomous Process Execution Management (APEM) (2022)
van der Aalst, Wil: Six Levels of Autonomous Process Execution Management (APEM); https://arxiv.org/pdf/2204.11328, 2022, Zugriff: 25.03.2024

van Kempen, R. et al.: AUTOtech.agil: Architecture for Automotive Agility (2023)
van Kempen, Raphael; Lampe, Bastian; Leuffen, Marc; Wirtz, Lena; Thomsen, Fabian; Bilkei-Gorzo, Gergely; Busch, Jean-Pierre; Feger, Ida; Geller, Christian; Kehl, Christian; Uszynski, Olaf; Wagner-Douglas, Lotte; Zanger, Lukas; Eckstein, Lutz; Klüner, David; Beerwerth, Julius; Alrifaee, Bassam; Kowalewski, Stefan; Konersmann, Marco; Steinfurth, Felix; Rumpe, Bernhard; Hartmann, Max; Siepenkötter, Norbert; Moormann, Dieter; Böhlen, Boris; Hannig, Claudia; Hekele, Esther; Gotzig, Heinrich; Rostocki, Paul-David; Gautam, Deepak-Kumar; Schubert, Richard; Braun, Niklas; Maurer, Markus; Gemlau, Kai-Björn; Abel, Sebastian; Ernst, Rolf; Lutwitzi, Melina; Bayerlein, Lorenz; Berghöfer, Moritz; Blödel, Alexander; Klamann, Björn; Kuznietsov, Anton; Peters, Steven; Leinen, Stefan; Bahle, Jakob; Ullrich, Lars; Graichen, Knut; Woopen, Timo; Spychalski, Dominik; Krauß, Christoph; Alayan, Mohamad; Giesler, Jens; Lilienthal, Martin; Schulik, Thomas; Lauer, Martin; Fernandez, Carlos; Molinos, Eduardo; Le Large, Nick; Rack, Nils; Steiner, Marlon; Wang, Kaiwen; Stiller, Christoph; Arndt, Gideon; Schulz, Benedikt; Furmans, Kai; Rauber, Stephan; Diermeyer, Frank; Brecht, David; Gehrke, Nils; Lienkamp, Markus; Zimmer, Walter; Creß, Christian; Zhou, Xingcheng; Knoll, Alois; Püllen, Dominik; Katzenbeisser, Stefan; Elmazi, Arlinda; Sailer, Andreas; Alfranseder, Martin; Mader, Ralph; Berkel, Felix; Specker, Thomas; Mayer, Philip; Hasseln, Hermann von; Jung, Lukas; Grandinetti, Marcel; Neidhart, Dominik; Greiner, Dan; Niedballa, Dennis; Zaheri, Dorsa; Maier, Jonas; Reuss, Hans-Christian; Afanasenko, Valentyna; Solomakha, Oleksandr; Roge, Swapnil S.; Kallfass, Ingmar; Buchholz, Michael; Dehler, Robin; Henning, Matti; Hermann, Charlotte; Schön, Markus; Dietmayer, Klaus: AUTOtech.agil: Architecture and Technologies for Orchestrating Automotive Agility, in: 32nd Aachen Colloquium Sustainable Mobility 2023, Aachen, Germany, 09.-11.10.2023

VDA QMC: Automotive SPICE (2017)
VDA QMC: Automotive SPICE Process Assessment / Reference Model; https://www.automotivespice.com/fileadmin/software-download/AutomotiveSPICE_PAM_31.pdf, 2017, Zugriff: 19.07.2022

Vesely, W. E.: Fault tree handbook (1981)
Vesely, William E.: Fault tree handbook, Washington, DC, 1981

Viehof, M.: Dissertation, Objektive Qualitätsbewertung durch statistische Validierung (2018)
Viehof, Michael: Objektive Qualitätsbewertung von Fahrdynamiksimulationen durch statistische Validierung, Dissertation
TU Darmstadt, Darmstadt, 2018

Viehof, M.; Winner, H.: Modellvalidierung im Anwendungsbereich der Fahrdynamiksimulation (2017)
Viehof, Michael; Winner, Hermann: Stand der Technik und der Wissenschaft: Modellvalidierung im Anwendungsbereich der Fahrdynamiksimulation, Darmstadt, 2017

Vogel, O.: Software-Architektur (2009)
Vogel, Oliver: Software-Architektur, 2. Edition, Springer, Dordrecht, 2009

Wachenfeld, W.: Dissertation, How Stochastic can Help to Introduce AD (2016)
Wachenfeld, Walther: How Stochastic can Help to Introduce Automated Driving, Dissertation
Technische Universität Darmstadt, Darmstadt, 2016

Wachenfeld, W.; Winner, H.: The Release of Autonomous Vehicles (2016)
Wachenfeld, Walther; Winner, Hermann: The Release of Autonomous Vehicles, in: Maurer, Markus et al. (Eds.): Autonomous Driving: Technical, Legal and Social Aspects, Springer, Berlin, Heidelberg, 2016

Wang, C.: Dissertation, Silent Testing for Safety Validation of Automated Driving (2021)
Wang, Cheng: Silent Testing for Safety Validation of Automated Driving in Field Operation, Dissertation
TU Darmstadt, Darmstadt, 2021

Wang, R. Y.; Strong, D. M.: What Data Quality Means to Data Consumers (1996)
Wang, Richard Y.; Strong, Diane M.: Beyond Accuracy: What Data Quality Means to Data Consumers, in: Journal of Management Information Systems (4), Issues 12, p. 5–33, 1996

WAYMO: Waymo Safety Report (2021)
WAYMO: Waymo Safety Report;
https://downloads.ctfassets.net/sv23gofxcuiz/4gZ7ZUxd4SRj1D1W6z3rpR/2ea16814cdb42f9e8eb34cae4f30b35d/2021-03-waymo-safety-report.pdf, 2021, Zugriff: 19.11.2022

Weber, H. et al.: A framework for definition of logical scenarios (2019)
Weber, Hendrik; Bock, Julian; Klimke, Jens; Roesener, Christian; Hiller, Johannes; Krajewski, Robert; Zlocki, Adrian; Eckstein, Lutz: A framework for definition of logical scenarios for safety assurance of automated driving, in: Traffic injury prevention sup1, Issues 20, S65-S70, 2019

Weber, N. et al.: Statistical approach for the derivation of scenarios (2020)
Weber, Nico; Frerichs, Dirk.; Eberle, Ulrich: A simulation-based, statistical approach for the derivation of concrete scenarios for the release of highly automated driving functions, in: AmE 2020 - Automotive meets Electronics; 11th GMM-Symposium, 2020

Weitzel, A. et al.: Absicherungsstrategien für Fahrerassistenzsysteme (2014)
Weitzel, Alexander; Winner, Hermann; Peng, Cao; Geyer, Sebastian; Lotz, Felix; Sefati, Mohsen: Absicherungsstrategien für Fahrerassistenzsysteme mit Umfeldwahrnehmung, Berichte der Bundesanstalt für Straßenwesen FahrzeugtechnikHeft 98, Fachverl. NW, Bremen, 2014

Westhofen, L. et al.: Criticality Metrics for Automated Driving (2022)
Westhofen, Lukas; Neurohr, Christian; Koopmann, Tjark; Butz, Martin; Schütt, Barbara; Utesch, Fabian; Kramer, Birte; Gutenkunst, Christian; Böde, Eckard: Criticality Metrics for Automated Driving: A Review and Suitability Analysis of the State of the Art, in: Archives of Computational Methods in Engineering (30), pp. 1–35, 2022

Woopen, T. et al.: UNICARagil - Disruptive Modular Architectures (2018)
Woopen, Timo; Lampe, Bastian; Böddeker, Torben; Eckstein, Lutz; Kampmann, Alexandru; Alrifaee, Bassam; Kowalewski, Stefan; Moormann, Dieter; Stolte, Torben; Jatzkowski, Inga; Maurer, Markus; Möstl, Mischa; Ernst, Rolf; Ackermann, Stefan; Amersbach, Christian; Leinen, Stefan; Winner, Hermann; Püllen, Dominik; Katzenbeisser, Stefan; Becker, Matthias; Stiller, Christoph; Furmans, Kai; Bengler, Klaus; Diermeyer, Frank; Lienkamp, Markus; Keilhoff, Dan; Reuss, Hans-Christian; Buchholz, Michael; Dietmayer, Klaus; Lategahn, Henning; Siepenkötter, Norbert; Elbs, Martin; Hinüber, Edgar v.; Dupuis, Marius; Hecker, Christian: UNICARagil - Disruptive Modular Architectures for Agile, Automated Vehicle Concepts, in: 27th Aachen Colloquium, Aachen, Germany, 2018

Woopen, T. et al.: UNICARagil news Juli 2020 (2020)
Woopen, Timo; Siepenkötter, Norbert; Stolte, Torben; Graubohm, Robert; Maurer, Markus; Jatzkowski, Inga; Ackermann, Stefan; Winner, Hermann; Gemlau, Kai-Björn; Lippert, Moritz; Klamann, Björn; Katzenbeisser, Stefan; Püllen, Dominik; Keilhoff, Dan; Niedballa, Dennis; Zaheri, Dorsa; Gehringer, Daniel; Goth, Marcus, Hisung, Matthias; Alrifaee, Bassam; Kampmann, Alexandru; Mokhtarian, Armin; Buchholz, Michael; Henning, Matti; Horn, Markus; Danzer, Andreas; Herzog, Manuel; Kinzig, Christian; Gies, Fabian; Schön, Markus; Kamran, Danial; Wang, Lingguang; Gottschalg, Grischa; Homolla, Tobias; Graf, Jürgen; Diermeyer, Frank; Feiler, Johannes; Hoffmann, Simon; Lampe, Bastian; Böddeker, Torben; Westerkamp, Philip; Schockenhoff, Ferdinand; Kipp, Manuel; Markert, Kai; Schräder, Tobias; Schwiebacher, Johannes; Koch, Alexander; König, Adrian; Struth, Michael: UNICARagil news Juli 2020;
https://www.unicaragil.de/images/medien/news/NewsletterHZE_07-2020_DE.pdf, 2020, Zugriff: 04.09.2023

Woyczyński, W. A.: Statistics for signal analysis (2006)
Woyczyński, Wojbor A.: A first course in statistics for signal analysis, Birkhäuser, Boston, Basel, Berlin, 2006

Wright, H.: Hyrum's Law (2022)
Wright, Hyrum: Hyrum's Law; https://www.hyrumslaw.com/, 2022, Zugriff: 04.06.2023

Yarlagadda, R. K.: Analog and digital signals and systems (2010)
Yarlagadda, R. K. R.: Analog and digital signals and systems, Electrical engineering, Springer, New York, Heidelberg, 2010

Zhao, D.: Dissertation, Accelerated Evaluation of Automated Vehicles (2016)
Zhao, Ding: Accelerated Evaluation of Automated Vehicles, Dissertation
University of Michigan, Ann Arbor, Michigan, USA, 2016

Eigene Veröffentlichungen

Junietz, Philipp; Bonakdar, Farid; Klamann, Björn; Winner, Hermann: Criticality Metric for the Safety Validation of Automated Driving using Model Predictive Trajectory Optimization, in: 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, USA, 2018

Klamann, Björn; Lippert, Moritz; Amersbach, Christian; Winner, Hermann: Defining Pass-/Fail-Criteria for Particular Tests of Automated Driving Functions, in: 2019 IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, NZ, 2019

Klamann, Björn; Winner, Hermann: Comparing Different Levels of Technical Systems for a Modular Safety Approval—Why the State of the Art Does Not Dispense with System Tests Yet, in: Energies (22), Issues 14, 2021

Lippert, Moritz; Klamann, Björn; Amersbach, Christian; Winner, Hermann: Definition von Bestehens-/Versagenskriterien für das partikuläre Testen von automatisierten Fahrfunktionen, in: TU München; TÜV Süd (Eds.): 9. Tagung Automatisiertes Fahren, München, 2019

Lippert, Moritz; Klamann, Björn; Winner, Hermann: Modulare Absicherung, Posterpräsentation; https://www.unicaragil.de/de/events/digitales-halbzeitevent.html#ModulareAbsicherung, 2020, Zugriff: 27.11.2022

Stolte, Torben; Graubohm, Robert; Jatzkowski, Inga; Maurer, Markus; Ackermann, Stefan; Klamann, Björn; Lippert, Moritz; Winner, Hermann: Towards Safety Concepts for Automated Vehicles by the Example of the Project UNICAR*agil*, in: 29th Aachen Colloquium Sustainable Mobility, Aachen, 2020

Stolte, Torben; Ackermann, Stefan; Graubohm, Robert; Jatzkowski, Inga; Klamann, Björn; Winner, Hermann; Maurer, Markus: A Taxonomy to Unify Fault Tolerance Regimes for Automotive Systems: Defining Fail-Operational, Fail-Degraded, and Fail-Safe, in: IEEE Transactions on Intelligent Vehicles, vol. 7, no. 2, 2021

Klamann, Björn; Winner, Hermann: Analyse von Dekompositionsprozessen zur Umsetzung einer modularen Absicherung automatisierter Fahrzeuge, in: Uni-DAS e.V. (Ed.): 14. Workshop Fahrerassistenz und automatisiertes Fahren, Berkheim, 2022

Klamann, Björn; Winner, Hermann: Introducing the Detailed Semantic Interface Description to Support a Modular Safety Approval of Automated Vehicles – S^2I^2, in: Safety and Reliability, S. 1-40, 2023

van Kempen, Raphael; Lampe, Bastian; Leuffen, Marc; Wirtz, Lena; Thomsen, Fabian; Bilkei-Gorzo, Gergely; Busch, Jean-Pierre; Feger, Ida; Geller, Christian; Kehl, Christian; Uszynski, Olaf; Wagner-Douglas, Lotte; Zanger, Lukas; Eckstein, Lutz; Klüner, David; Beerwerth, Julius; Alrifaee, Bassam; Kowalewski, Stefan; Konersmann, Marco; Steinfurth, Felix; Rumpe, Bernhard; Hartmann, Max; Siepenkötter, Norbert; Moormann, Dieter; Böhlen, Boris; Hannig, Claudia; Hekele, Esther; Gotzig, Heinrich; Rostocki, Paul-David; Gautam, Deepak-Kumar; Schubert, Richard; Braun, Niklas; Maurer, Markus; Gemlau, Kai-Björn; Abel, Sebastian; Ernst, Rolf; Lutwitzi, Melina; Bayerlein, Lorenz; Berghöfer, Moritz; Blödel, Alexander; Klamann, Björn; Kuznietsov, Anton; Peters, Steven; Leinen, Stefan; Bahle, Jakob; Ullrich, Lars; Graichen, Knut; Woopen, Timo; Spychalski, Dominik; Krauß, Christoph; Alayan, Mohamad; Giesler, Jens; Lilienthal, Martin; Schulik, Thomas; Lauer, Martin; Fernandez, Carlos; Molinos, Eduardo; Le Large, Nick; Rack, Nils; Steiner, Marlon; Wang, Kaiwen; Stiller, Christoph; Arndt, Gideon; Schulz, Benedikt; Furmans, Kai; Rauber, Stephan; Diermeyer, Frank; Brecht, David; Gehrke, Nils; Lienkamp, Markus; Zimmer, Walter; Creß, Christian; Zhou, Xingcheng; Knoll, Alois; Püllen, Dominik; Katzenbeisser, Stefan; Elmazi, Arlinda; Sailer, Andreas; Alfranseder, Martin; Mader, Ralph; Berkel, Felix; Specker, Thomas; Mayer, Philip; Hasseln, Hermann von; Jung, Lukas; Grandinetti, Marcel; Neidhart, Dominik; Greiner, Dan; Niedballa, Dennis; Zaheri, Dorsa; Maier, Jonas; Reuss, Hans-Christian; Afanasenko, Valentyna; Solomakha, Oleksandr; Roge, Swapnil S.; Kallfass, Ingmar; Buchholz, Michael; Dehler, Robin; Henning, Matti; Hermann, Charlotte; Schön, Markus; Dietmayer, Klaus: AUTOtech.agil: Architecture and Technologies for Orchestrating Automotive Agility, in: 32nd Aachen Colloquium Sustainable Mobility, Aachen, 2023

Betreute studentische Arbeiten

Abali, Emre: Definition von Anforderungen und Sicherheitszielen automatisierter Fahrzeuge und deren Ableitung auf Modulebene, Master Thesis Nr. 772/20, 2020

Blödel, Alexander: Entwicklung und Implementierung eines Verfahrens zur Generierung von Trajektorien für das modulare Testen automatisierter Fahrfunktionen, Master Thesis Nr. 803/20, 2021

Fröhlich, Kevin: Konzeptionierung eines Antriebsstrangs für den Formula Student Electric Rennwagen des TU Darmstadt Racing Team e.V., Bachelor Thesis Nr. 1344/19, 2019

Grube Doiz, Nerea: Potenzialanalyse der Modularisierung in der Automobilindustrie für die Freigabe automatisierter Fahrzeuge, Bachelor Thesis Nr. 1369/20, 2020

Kohls, Svenja: Entwicklung eines Verfahrens zur Modellierung von Abhängigkeiten zwischen Modulen für die modulare Absicherung automatisierter Fahrfunktionen, Master Thesis Nr. 835/21, 2022

Krause, Johannes: Entwicklung von Tests für die Freigabe des Dynamikmoduls als Teil der Absicherung eines automatisierten Fahrzeugs, Bachelor Thesis Nr. 1364/20, 2021

Lang, Enno: Identifizierung und Analyse relevanter Einflussgrößen zur Absicherung von Modulen autonomer Fahrzeuge, Master Thesis Nr. 765/19, 2020

Langtao, Liao: Entwicklung und Implementierung eines Verfahrens zur Generierung von Umfeldmodelldaten für das modulare Testen automatisierter Fahrfunktionen, Master Thesis Nr. 798/20, 2021

Müller, Laurenz: Integration und Bewertung von Methoden zur Sensitivitätsanalyse von Szenarien für das hochautomatisierte Fahren, Bachelor Thesis Nr. 1386/21, 2021

Muneeb, Syed Rafay: Entwicklung eines Inbetriebnahme- und Debugging-Prozesses für die diensteorientierte Architektur eines automatisierten Fahrzeugs, Master Thesis Nr. 806/20, 2021

Olbrich, Paul: Potentialanalyse eines Methodenprüfstands für Automatisiertes Fahren, Bachelor Thesis Nr. 1405/23, 2023

Prajapati, Nabin: Entwicklung und Umsetzung eines Absicherungskonzepts zur Sicherstellung der Applikationsgüte von OBD2 Diagnosen bei Ottomotoren, Master Thesis Nr. 783/20, 2021

Scholl, Karl: Konzeptionierung und Konstruktion eines Lenkaktors für ein automatisiertes Formula Student Fahrzeug, Bachelor Thesis Nr. 1343/19, 2019

Smits, Leonard: Entwicklung einer Methode zur Generierung und Durchführung von Testfällen für die Verhaltens- und Trajektorienplanung automatisierter Fahrzeuge, Master Thesis Nr. 836/21, 2022

Troll, Niklas: Analyse des Einsatzes der funktionalen Dekomposition automatisierter Fahrfunktionen auf den urbanen Raum, Master Thesis Nr. 800/20, 2021

Vierneisel, Florian: Entwicklung von Testfällen zur Absicherung von Modulen automatisierter Fahrzeuge, Bachelor Thesis Nr. 1340/19, 2019

Wang, Boyan: Risikoanalyse der Leitwarte als Teil eines teleoperierten Modus von automatisierten Straßenfahrzeugen, Master Thesis Nr. 798/20, 2021

Wang, Shicheng: Entwicklung eines Testkonzepts und Generierung von Testfällen für Teleoperationssysteme von automatisierten Fahrzeugen, Master Thesis Nr. 838/21, 2022

Weber, Nico: Reduzierung des Parameterraums für die Freigabe von hochautomatisierten Fahrfunktionen, Master Thesis Nr. 736/19, 2019

Wu, Di: Development of a Methodology for Deriving Parameter Constraints for Combinatorial Testing, Master Thesis Nr. 816/21, 2021