Methoden und Ansätze für die Entwicklung und den Test prädiktiver Fahrzeugregelungsfunktionen

Zur Erlangung des akademischen Grades eines

DOKTOR-INGENIEURS

von der Fakultät für Elektrotechnik und Informationstechnik
des Karlsruher Instituts für Technologie (KIT)
genehmigte

DISSERTATION

von

Dipl.-Ing. Johannes Bach
geb. in Mutlangen

Tag der mündlichen Prüfung: 20. Februar 2018
Hauptreferent: Prof. Dr.-Ing. Eric Sax
Korreferent: Prof. Dr. Frank Köster

Kurzfassung

In dieser Arbeit werden das aktuelle Vorgehen und die Prozesse in der automobilen Produktentwicklung sowie die etablierten Methoden für die Entwicklung, Verifikation und Validierung von Fahrzeugregelungsfunktionen analysiert. Dem wird eine Taxonomie und Analyse aktueller Serienanwendungen und Forschungskonzepte gegenüber gestellt. Ziel ist es, durch eine ganzheitliche Betrachtung die aktuellen Rahmenbedingungen und Herausforderungen bei der Entwicklung innovativer Funktionen für die Automatisierung der Fahraufgabe zu identifizieren. Auf dieser Grundlage wird ein neuartiges Konzept für die Entwicklung und den Test prädiktiver Fahrzeugregelungsfunktionen erarbeitet.

Das Kernstück des entwickelten Konzepts stellt die Reactive-Replay Methode dar. Sie ermöglicht eine enge Verzahnung von Erprobungsfahrten in der realen Welt mit der Ausführung der entwickelten Fahrzeugfunktion innerhalb einer Simulationsumgebung. Die adaptive Wiedergabe von während der Erprobung aufgezeichneten Daten des fahrzeuginternen Kommunikationsnetzes ermöglicht einen nahtlosen Übergang von der realen Welt im Fahrzeug in die Simulation im Büro. Auf diese Weise können in der Realität aufgetretene Situationen und Szenarien detailliert und unter Laborbedingungen untersucht und für Tests wiederverwendet werden. Darüber hinaus ermöglicht dieser Ansatz eine effiziente Generierung valider Testszenarien, die durch ihre Vielfältigkeit und Varianz zu einer verbesserten Testabdeckung beitragen.

Um die entwickelte Methode systematisch in den produktiven Alltag der Funktionsentwicklung zu integrieren, wird ein schlankes, iteratives Vorgehen zur prozessualen Integration der Reactive-Replay Methode vorgeschlagen. Die Verifikation in der Simulationsumgebung wird so mit der Validierung in der Fahrzeugerprobung gekoppelt. Dies unterstützt die frühzeitige und durchgängige Qualitätsbewertung der entwickelten Fahrzeugfunktion. Weiter wird eine Methode zur kontinuierlichen Überprüfung von Anforderungen während der Simulationsausführung untersucht. Ein Ansatz zur effizienten Auswahl von Testszenarien auf Basis der innerhalb eines Szenarios erreichten Parameterüberdeckung rundet die Arbeit ab.

Abstract

In this thesis the current procedure and processes in the automotive product development as well as the established methods for the development, verification and validation of automotive systems are analyzed. This is contrasted with a taxonomy and analysis of current series applications and research concepts. The aim is to identify the current requirements and challenges in the development of innovative functions for assisted and automated driving through a holistic view. On this basis a new concept for the development and test of predictive vehicle control functions is developed.

As the core feature of this new concept, the Reactive-Replay method allows close interlinking of real-world trials with the execution of the developed vehicle function within a simulation environment. The adaptive replay of vehicle-internal communication data recorded during t enables a seamless transition from the real world to the simulation. In this way, situations and scenarios occurring in reality can be investigated in detail and under laboratory conditions and re-used for tests. In addition, the approach allows efficient generation of valid test scenarios that contribute to improved test coverage through their versatility and variance.

In order to integrate the developed method into the daily productive development activities, the Reactive-Replay based verification is coupled with the real world trials in a lean and iterative procedure. Furthermore, a method for the continuous verification of requirements during the simulation execution is examined. An approach for the efficient selection of test scenarios on the basis of parameter value coverage completes the work.

Danksagung

Die vorliegende Dissertation entstand im Rahmen meiner Tätigkeit als wissenschaftlicher Mitarbeiter am FZI Forschungszentrum Informatik im Forschungsbereich Embedded Systems and Sensors Engineering (ESS) und in enger Kooperation mit der Dr. Ing. h.c. F. Porsche AG.
Herzlich danken möchte ich meinem Doktorvater Prof. Dr.-Ing. Eric Sax für die Möglichkeit der Promotion und das mir stets entgegengebrachte Vertrauen. Seine vielfältigen Anregungen förderten meine eigenverantwortliche und verbindliche Arbeitsweise auch über die Gestaltung meiner wissenschaftlichen Arbeit hinaus. Herrn Prof. Dr. Frank Köster danke ich für die Übernahme des Korreferats und die engagierte Diskussion der Arbeitsinhalte. Prof. Dr.-Ing. Klaus D. Müller-Glaser danke ich für die wohlwollende Unterstützung und große Inspiration in den Anfangsphasen meiner Arbeit und darüber hinaus.
Für die Unterstützung und Förderung meiner Arbeit sowie anregende und inspirierende Gespräche danke ich allen Kollegen des Sachgebiets energieeffiziente prädiktive Fahrerassistenzsysteme der Dr. Ing. h.c. F. Porsche AG, insbesondere Marc Holzäpfel, Kai-Lukas Bauer und Sebastian Fünfgeld.
Mein besonderer Dank gilt auch allen Mitarbeitern des Forschungsbereichs ESS für die motivierende und kreative Arbeitsatmosphäre. Für viele Impulse und gemeinsame Stunden danke ich Johannes Schneider, Jacob Langner, Timon Blöcher, Markus Schienle, Friedrich Gauger und Christoph Zimmermann sowie allen von mir betreuten Studenten für ihren wichtigen Beitrag zu dieser Arbeit. Ganz besonderer Dank gilt Stefan Otten für seine Freundschaft, stete Motivation und Unterstützung in zahlreichen Projekten und Veröffentlichungen.
Ganz herzlich danke ich meinem Bruder Mathias und meinen Eltern Doris und Franz für die fortwährende Unterstützung und Ermutigung zur Fertigstellung dieser Arbeit. Meiner Freundin Katrin Heen danke ich aus vollem Herzen für ihre unerschöpfliche Geduld, ihre Zuneigung und ihr Verständnis, die mich auch durch schwierige Zeiten dieser Arbeit getragen haben.

Karlsruhe, im April 2018 *Johannes Bach*

Inhaltsverzeichnis

1 Einleitung

Dieses Kapitel leitet das Themengebiet der vorliegenden Dissertation anhand der Digitalisierung und der Innovationskraft Software-getriebener Kundenfunktionen im Automobilbereich ein. Darauf folgt die wissenschaftliche Zielsetzung der Arbeit, deren Umfeld und Rahmenbedingungen sowie deren Aufbau.

1.1 Motivation

Die digitale Revolution [1] führt zu einem rasanten Wandel zentraler Bereiche unserer Gesellschaft und Arbeitswelt. Mit dem Anstieg der technischen Kapazitäten zur Informationsverarbeitung und -speicherung wächst auch die Menge digitaler Informationen. Weiter ermöglichen Breitbandtechnologien zur Datenübertragung die Vernetzung mobiler Systeme unterschiedlichster Bereiche. Insbesondere am Beispiel des Automobils kann die Digitalisierung und Vernetzung aufgezeigt werden.

Im Automobilsektor liegt ein wesentlicher Anteil der Innovationskraft neuer Fahrzeuggenerationen in Kundenfunktionen, welche zunehmend in Software realisiert werden. Berlinghaus [2] formuliert drastisch: „Elektronik und Software beherrschen die Innovationen im Auto“. Hierin liegt gleichzeitig ein hohes Differenzierungspotential der verschiedenen Automobilhersteller [3]. Dieses Differenzierungspotential verschärft den Konkurrenzdruck und öffnet den Markt für neue Akteure. Insbesondere Innovationen im Bereich der Unterhaltungselektronik, wie leistungsstarke Smartphones und Tablets, dringen über Anwendungen und Services in den Automobilsektor vor. Beispielsweise können Navigationsapps mit Online-Verkehrsinformationen auf Augenhöhe mit den fest in das Infotainment der Fahrzeuge integrierten Navigationslösungen konkurrieren [4].

Das automatisierte Fahren und die vernetzte Mobilität [5] tragen zusätzlich zum schnellen Wachstum der Software-getriebenen Innovationen bei. Dies steigert die

Produktkomplexität weiter [6]. Abbildung 1.1 skizziert die zunehmende Produktkomplexität an Beispielfunktionen über einen Zeitraum von 25 Jahren.

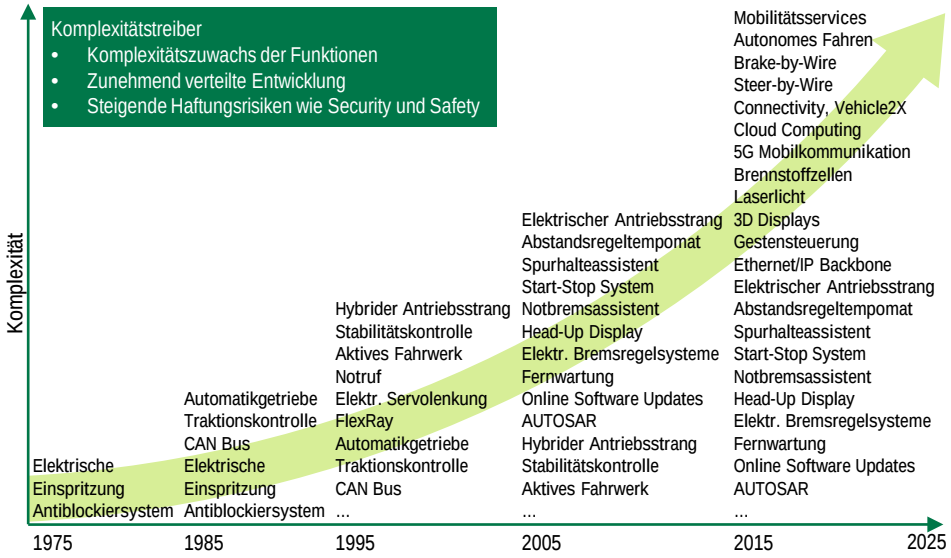


Abbildung 1.1: Das Wachstum Software-getriebener Innovationen trägt maßgeblich zu einer erhöhten Produktkomplexität bei [6].

Diese Komplexitätszunahme führt die gängigen Vorgehensweisen in der Entwicklung, Verifikation und Validierung von Fahrzeugsystemen an ihre Grenzen. Beispielsweise wurde in einer Studie für die Freigabe eines automatisierten Autobahnpiloten durch Erprobung in der realen Welt statistisch ein benötigter Testumfang von 6 Milliarden Testkilometer berechnet [7]. Dieser Testumfang ist mit Erprobungsträgern und Realfahrten nicht realisierbar. Zudem sind in dieser Abschätzung des Testumfangs sehr viele redundante und für die Absicherung nicht relevante Situationen und Abschnitte enthalten. Um den Testumfang bei gleicher Testabdeckung zu verringern, werden neuartige Ansätze benötigt, die zu einer Komprimierung des benötigten Testumfangs beitragen. Sie sollen es den Entwicklern ermöglichen, einen Fokus auf relevante Szenarien und Situationen zu legen, in denen das System derartig beansprucht wird, dass die eigentliche Systemqualität beurteilt werden kann [8]. Diese ausgewählten Szenarien müssen dann sowohl in der Simulation getestet, als auch in der realen Welt erprobt werden.

1.2 Zielsetzung der Arbeit

Um die im Zuge der Digitalisierung zunehmende Produktkomplexität auch zukünftig beherrschbar zu halten, liefert diese Dissertation einen Beitrag zur Weiterentwicklung der im Automotive Systems Engineering (ASE) eingesetzten Vorgehensweisen und Methoden. Der Schwerpunkt der Arbeit liegt auf der Erforschung innovativer Methoden und Ansätze für die Entwicklung und den Test von prädiktiven Fahrzeugregelungsfunktionen sowie deren Integration in die etablierten Prozesse und Vorgehensweisen im Automobilsektor. Basierend auf einer Analyse des Stands der Technik und der Anforderungen durch zukünftige Fahrzeugfunktionen stehen folgende Fragestellungen im Fokus:

- Welche besonderen Herausforderungen an die Prozesse und Methoden des Systems Engineering ergeben sich durch prädiktive Fahrzeugregelungsfunktionen?

- Wie können virtuelle Methoden für die Entwicklung und den Test von Fahrzeugfunktionen gestärkt werden und einen größeren Beitrag zur Produktentwicklung leisten?

- Kann eine Methode entwickelt werden, die virtuelle Ansätze mit den Erprobungen in der realen Welt verbindet und kann dies die Absicherungstiefe verbessern?

Neben dem methodischen Fokus entstehen aus der Entwicklungsperspektive weitere Fragen in Bezug auf die Vorgehensweise und Abläufe während der Produktentwicklung, die unter Berücksichtigung des aktuellen Wissensstands in der Forschung untersucht werden:

- Wie können die Methoden im produktiven Alltag effizient eingesetzt werden, um schlanke und agile Prozesse und Abläufe zu erreichen?

- Wie kann die Testabdeckung von automatisierten Fahrfunktionen bewertet werden, die in der realen Welt vielfältigen Szenarien und Situationen ausgesetzt sind?

1.3 Umfeld

Diese Dissertation entstand im Rahmen der Zusammenarbeit zwischen dem FZI Forschungszentrum Informatik und der Dr. Ing. h.c. F. Porsche AG, deren Projekte im Bereich prädiktive Assistenzsysteme die Ausrichtung der Forschungsarbeit prägten. Während der Promotionszeit wurde dort mit dem Prädiktiver Abstandsregeltempomat (Predictive Cruise Control, PCC) eine Kundenfunktion zur zeit-, energie- und komfortoptimierten Längsführung erforscht, entwickelt und in einem Serienfahrzeug in den Markt eingeführt. Daher stand für alle entwickelten Methoden und Ansätze deren Anwendbarkeit im produktiven Alltag eines Entwicklungsprojekts im Fokus.

1.4 Aufbau

Der Aufbau dieser Dissertation ist in Abbilung 1.2 dargestellt. Dem Stand der Technik zu Vorgehensmodellen und Prozessen in der Systementwicklung in Kapitel 2 und zu Methoden für die Entwicklung, Verifikation und Validierung in Kapitel 3 wird in Kapitel 4 eine Analyse aktueller Serienanwendungen und Forschungskonzepte gegenübergestellt.

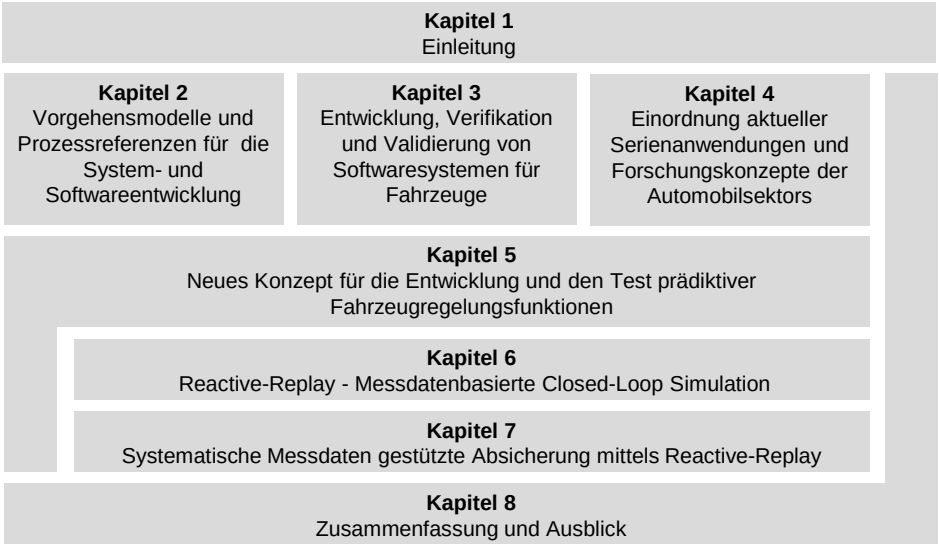


Abbildung 1.2: Aufbau der Dissertation.

In Kapitel 5 werden auf Basis der vorhergehenden Analysen aktuelle Herausforderungen identifiziert und gleichzeitig ein neues Konzept für die Entwicklung und den Test prädiktiver Fahrzeugregelungsfunktionen vorgestellt. In Kapitel 6 wird die Reactive-Replay Methode als innovativer Kern dieser Dissertation und als neuartiger Ansatz zur Verzahnung eingesetzter Simulationstechniken und Erprobungsfahrten in der realen Welt präsentiert. Die systematische Integration der Reactive-Replay Methode in das Entwicklungsvorgehen wird im darauf folgenden Kapitel 7 diskutiert. Kapitel 8 fasst die Ergebnisse dieser Arbeit zusammen und zeigt im Ausblick mögliche Anknüpfungspunkte auf.

2 Vorgehensmodelle und Prozessreferenzen für die System- und Softwareentwicklung

Der Begriff Systems Engineering beschreibt den ingenieurmäßigen Systementwurf. Dieser umfasst neben Prozess und Vorgehen die angewandten Methoden, sowie die verwendeten Werkzeuge. Prozesse widmen sich in diesem Kontext der Strukturierung und Regelung des zeitlich ausgedehnten Vorgangs der Systementwicklung. Methoden strukturieren und regeln ein Vorgehen zur Erlangung gewünschter Erkenntnisse oder Ergebnisse. Dieses Kapitel führt die verschiedenen Teilaspekte der Produktkomplexität ein, die die Notwendigkeit eines strukturierten und ingenieurmäßigen Vorgehens in der Automobilentwicklung begründen. In Abschnitt 2.2 werden die wichtigsten Grundlagen der Systementwicklung mit Bezug zur Elektrik/Elektronik (E/E) in Automobilen dargelegt. Darauf folgend stellt das Kapitel verschiedene Vorgehensmodelle aus der System- und Softwareentwicklung, sowie deren Anwendung in einem exemplarischen Produktentwicklungsprozess der Automobilindustrie dar.

2.1 Komplexität des Systementwurfs

Der Begriff der Komplexität[1] wird häufig unspezifisch verwendet und unterliegt unterschiedlichen und kontextabhängigen Definitionen. Renner [9] liefert die in Abbildung 2.1 dargestellte Zusammenfassung der systemtheoretischen Bedeutung des Begriffs Komplexität im Kontext des Automobils:

[1] „Vielschichtigkeit; das Ineinander vieler Merkmale“, Duden, http://www.duden.de/rechtschreibung/Komplexitaet

Definition 2.1 (Komplexität)**.** *Die* Komplexität *von Produkten setzt sich aus der Vielfalt (Anzahl und Unterschiedlichkeit der einzelnen Elemente), der Konnektivität (Anzahl und Unterschiedlichkeit an Beziehungen), der internen Dynamik des Systems, den Wechselwirkungen mit dem Umfeld sowie den im System enthaltenen Unschärfen zusammen.*

Weitere Komplexitätstreiber, und damit Gründe für ein methodisches Vorgehen in der Automobilentwicklung, sind die weltweilt verteilten Absatzmärkte und einhergehenden gesetzlichen und gesellschaftlichen Rahmenbedingungen, sowie ein durch den Kunden erwarteter mehrjähriger Produktlebenszyklus und der damit entsprechend langen Wartungsverpflichtung des Herstellers.

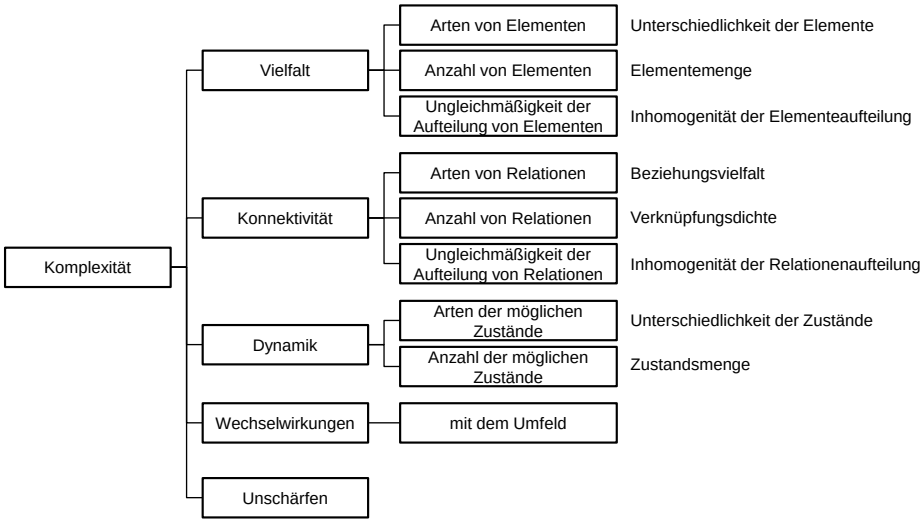


Abbildung 2.1: Verschiedene Teilaspekte der Produktkomplexität [9]

Für den Umgang mit diesen vielfältigen Aspekten der Produkt- und Entwicklungskomplexität existieren unterschiedliche Ansätze des Systems und Software Engineering. Den verwendeten Ansätzen liegen verschiedene Prinzipien zugrunde. Durch Abstraktion wird ein komplexes Objekt auf die für die jeweilige Tätigkeit relevanten Informationen reduziert, um dessen Handhabung zu vereinfachen. Eine ähnliche Informationsreduktion wird mit dem Prinzip der Kapselung und dem verstecken von Informationen (Information-Hiding) verfolgt. Dekomposition und Modularisierung beschreiben die Aufteilung eines großen, komplexen Problems in mehrere kleinere, einfacher zu lösende Teilprobleme. Eine detaillierte Definition in Bezug auf das Software Engineering bietet Bourque [10].

2.2 Strukturierung der Elektrik/Elektronik-Entwicklung im Automobilsektor

Das Prinzip der Kapselung findet in der Automobilentwicklung durch die übliche Unterteilung in fachliche Kompetenzbereiche Anwendung, die ihren Ursprung in der mechanischen Modularisierung der Fahrzeuge haben. Diese Modularisierung spiegelt sich in der Organisationsstruktur der Entwicklungsabteilungen und der technischen Architektur der E/E-Systeme wider [11]. Diese Spiegelung wird als „Conway's Law" bezeichnet [12]. Die Kommunikationsnetzwerke sind dezentral ausgelegt und entsprechend der Kompetenzbereiche in sogenannte Fahrzeugdomänen strukturiert [13]. Ein zentrales Gateway dient als Knotenpunkt zur Quervernetzung zwischen den voneinander unabhängigen Teilnetzwerken der einzelnen Domänen [14, 15].

Weber [11] beschreibt eine gängige Aufteilung der E/E-Systeme und Ausstattungsmerkmale (Feature) in die Fahrzeugdomänen Antriebsstrang, Chassis, Karosserie (Body), Infotainment und passive Sicherheit (Safety). Eine ähnliche Aufteilung wird auch in einem Kompendium zur Automobilelektronik der Robert Bosch GmbH beschrieben [16]. Tabelle 2.1 beinhaltet die durch Weber vorgenommene Zuordnung von Ausstattungsmerkmalen zu den Fahrzeugdomänen.

Fahrzeugdomäne	**Zugehörige Beispielfeatures**
Antriebsstrang	Motorsteuergerät, Getriebesteuergerät, Kraftstoffpumpe, On-Board-Diagnose
Chassis	Antiblockiersystem (ABS), Elektronische Stabilitätskontrolle (Electronic Stability Control, ESC), Traktionskontrolle
Karosserie	Schließsystem, Fensterheber, Scheibenwischer, Sitze, Klimatisierung
Infotainment	Kombiinstrument, Navigationssystem, Audio, Video, Telefonie
Passive Sicherheit	Airbags, Gurtstraffer, Überrollschutzsystem

Tabelle 2.1: Zuordnung von E/E-Features zu Fahrzeugdomänen nach [11, 16]

Die Unterteilung der mechanisch dominierten Baugruppen Antriebsstrang, Chassis und Karosserie reflektiert den Produktionsprozess. Die Montage der Baugruppen erfolgt zunächst unabhängig voneinander. Die Komponenten des Antriebsstrangs und des Chassis werden abhängig der gewählten Umsetzung[2] zusammengeführt

[2] z. B. Plattformrahmen, „Space Frame" oder Monocoque

und während des als Hochzeit bezeichneten Produktionsschritts mit der Karosserie vereinigt [17]. Diese Modularisierung unterstützt die Spezialisierung der in Entwicklung und Produktion eingesetzten Methoden und Techniken, sowie der beteiligten Personen. Die ergänzenden Fahrzeugdomänen Infotainment und Safety verdeutlichen deren großen Anteil an der Gesamtheit der E/E Komponenten, bzw. die besondere Kritikalität der darin enthaltene Features. Die Domäne Safety beinhaltet, wie in Tabelle 2.1 dargestellt, Features der passiven Sicherheit und sollte nicht mit der funktionalen Sicherheit (Functional Safety) verwechselt werden.

Neben der Digitalisierung ist die Elektrifizierung aktuell ein wichtiger Innovationstreiber der Automobilindustrie. Die Einführung von 48 Volt-Netzen für Hybrid-Elektrofahrzeuge und von Hochspannungsnetzen für vollelektrische Fahrzeuge beruht auf dieser Entwicklung. Um sie zu berücksichtigen, werden die nach Weber [11] etablierten Fahrzeugdomänen, dargestellt in Tabelle 2.1, in dieser Arbeit um eine Bordnetz-Domäne ergänzt. Diese erlaubt eine von der übrigen Entwicklung losgelöste Betrachtung dedizierter Fragestellungen der Batterie-, Starkstrom- und Hochspannungstechnik und eine Konzentration des erarbeiteten Fach- und Expertenwissens. Die resultierende Aufteilung der Fahrzeugdomänen ist in Abbildung 2.2 dargestellt.

Abbildung 2.2: Aufteilung des Gesamtfahrzeugs in fachspezifische Fahrzeugdomänen

In einer stark vereinfachten und abstrakten Darstellung besteht das E/E-System eines Fahrzeugs aus der in Abbildung 2.3 dargestellten Wirkkette eines Sensors zur Wahrnehmung, eines Steuergeräts (Electronic Control Unit, ECUs) zur Informationsverarbeitung und eines Aktuators zur Ausführung.

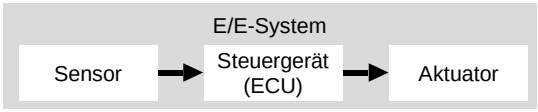


Abbildung 2.3: Vereinfachte Darstellung eines aus einem Sensor, einem Steuergerät und einem Aktuator bestehenden E/E Systems
(vergl. [18])

Die Definitionen der Begriffe Steuergerät, Sensor und Aktuator liefert [18] mit:

Definition 2.2 (Steuergerät)**.** *Ein* Steuergerät (Electronic Control Unit, ECU) *ist die physische Umsetzung eines eingebetteten Systems. Es ist die Kontrolleinheit eines mechatronischen Systems, das Sensoren und Aktoren als Peripherie besitzen kann.*

Definition 2.3 (Sensor)**.** *Ein* Sensor *(lat. sentire - fühlen), ist ein Fühler oder Aufnehmer, der eine physikalische oder chemische (meist nichtelektrische) Größe in eine elektrische Größe umsetzt.*

Definition 2.4 (Aktuator)**.** *Ein* Aktuator *(lat. agere - handeln), setzt Signale geringer Leistung, die Stellinformationen beinhalten, in leistungbehaftete Signale um, die in einer zur Prozessbeeinflussung notwendigen Energieform vorliegen. Sie bilden die Schnittstelle zwischen elektrischer Signal- bzw. Informationsverarbeitung und dem Prozess, also der Mechanik.*

Die E/E Systeme im Fahrzeug sind, bis auf wenige Ausnahmen in der Infotainment Domäne, als Echtzeitsysteme realisiert. Sax [19] definiert dazu:

Definition 2.5 (Echtzeitsystem)**.** *Verarbeitet ein System seine Eingangs- und Zustandsgrößen innerhalb eines vorgegebenen Zeitintervalls garantiert und erzeugt in diesem Zeitintervall die zugehörigen Ausgangs- und Zustandsgrößen, so spricht man von einem* Echtzeitsystem*.*

Definition 2.6 (Echtzeit)**.** *Von „harter* Echtzeit*“ spricht man, wenn ein Echtzeitsystem die Einhaltung des Echtzeitkriteriums unter allen Umständen garantiert, bzw. Verstöße detektiert und geeignete Fehlermaßnahmen einleitet.*
Von weicher Echtzeit *spricht man dagegen, wenn ein System nur in den meisten Fällen das Echtzeitkriterium erfüllt.*

2.2.1 Fahrzeugkommunikationsstruktur

Im Kontext des Gesamtfahrzeugs wird die Komplexität des E/E Anteils an der E/E-Topologie aktueller Fahrzeuggenerationen sichtbar. Eine Vielzahl verteilter Steuergeräte ist über die Fahrzeugkommunikationsstruktur untereinander verbunden und Fahrzeugfunktionen werden über diese verteilten Steuergeräte hinweg ausgeführt. Ein etablierter Ansatz zur Reduktion der Komplexität von E/E-Topologien ist deren Aufteilung in eigenständige Bussegmente, die über ein zentrales Gateway in Sterntopologie untereinander verbunden sind [3]. Die Aufteilung erfolgt dabei entsprechend der einzelnen Fahrzeugdomänen. Abbildung 2.4 skizziert einen Domaincontroller Ansatz und die enthaltenen Kommunikationstechnologien.

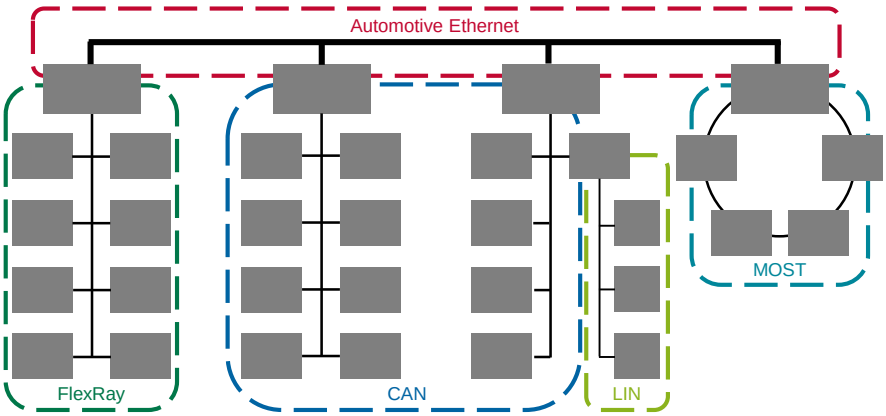


Abbildung 2.4: Skizze einer Domaincontroller basierten E/E-Architektur nach [13, 20]

Die wichtigsten Kommunikationstechnologien in der Automobilbranche umfassen neben den Multimaster-Systemen Controller Area Network (CAN) und FlexRay, die Singlemaster-Technologie Local Interconnect Network (LIN) und die üblicherweise als Ringtopologie im Infotainment eingesetzte Media Oriented Systems Transport (MOST)-Spezifikation [3]. Mit dem Hintergrund steigender Datenraten und dichterer Vernetzung setzen aktuellere E/E-Topologien zunehmend auf leistungsstarke Domaincontroller, die über einen automotive Ethernet basierten Backbone untereinander verknüpft sind [13, 20]. In diesen Bussystemen werden die einzelnen Signale in zu Botschaften zusammengefassten Datenpaketen übertragen.

2.2.2 Systemabstraktionsebenen

Neben der Kapselung einzelner Bereiche des Gesamtsystems in unterschiedliche Domänen nimmt das Prinzip der Dekomposition für die Komplexitätsreduktion in der E/E-Entwicklung eine Schlüsselstellung ein. Das Gesamtsystem wird in aus E/E-Sicht sachlogisch zusammenhängende Teile strukturiert, die eine weitestgehend unabhängige Realisierung ermöglichen. Die gesamte Dekomposition umfasst üblicherweise mehrere Hierarchieebenen.

Auch das Automotive SPICE[3] Prozess-Referenz- und Prozess-Assessment-Modell (siehe Abschnitt 2.7) sieht eine entsprechende Unterteilung des Gesamtsystems in Abstraktionsebenen vor. Als Beispiel für eine geeignete Abstraktionshierarchie werden im Rahmen dieser Dissertation die Ebenen System, Subsytem, Komponente (Component) und Einheit (Unit) verwendet. Diese Abstraktion vereint die Ansätze von Schäuffele (System, Subsystem, Komponente) [14] mit den Definitionen von Automotive SPICE (System, Komponente, Einheit). Weiter werden die Begriffe System, Subsystem, Komponente und Einheit im Sinne der ISO/IEC/IEEE24765 [21] verwendet:

Definition 2.7 (System)**.** *Ein* System *beschreibt die Kombination interagierender Elemente, die entsprechend ausgestattet sind, um einen oder mehrere festgelegte Bestimmungen zu erfüllen.*

Definition 2.8 (Subsystem)**.** *Ein* Subsystem *stellt ein sekundäres oder untergeordnetes System innerhalb eines größeren Systems dar.*

Definition 2.9 (Komponente)**.** *Eine* Komponente *beschreibt eine Einheit mit diskreter Struktur innerhalb eines Systems, wie beispielsweise eine Baugruppe oder ein Softwaremodul, das auf einer bestimmten Analyseebene betrachtet wird.*

Definition 2.10 (Einheit)**.** *Eine* Einheit *ist ein für sich prüfbares Element, das im Entwurf einer Softwarekomponente spezifiziert ist.*

Eine weitere durch Automotive SPICE definierte Unterscheidung betrifft die Begriffe Element (Element) und Gegenstand (Item). Die Einzelteile des Systems werden

[3] **S**oftware **P**rocess **I**mprovement and **C**apability **D**etermination

während der Spezifikation und dem Entwurf als Elemente und mit der Umsetzung und Implementierung als realisierte Gegenstände (Items) bezeichnet. Dabei entsprechen die realisierten Items den spezifizierten Elementen, wobei mehrere Elemente in einem Item zusammengefasst werden können.

2.2.3 Automatisierungsstufen

Neben der Zuordnung zu einer spezifischen Fahrzeugdomäne und der Einordnung in Systemabstraktionsebenen können die E/E-Features anhand ihres Automatisierungsgrades eingeordnet werden. Der amerikanische Ingenieursverband SAE International definiert sechs Automatisierungsstufen und spezifiziert Richtlinien zur Einordnung von Features in diese Stufen [22]. Die Einordnung analysiert dabei die Fähigkeit des Features zur Übernahme der Längs- und der Querregelung des Fahrzeugs sowie angemessen auf die Fahrzeugumgebung und Ereignisse zu reagieren. Weitere Unterscheidungskriterien sind im Feature enthaltene Rückfallebenen und die vorgesehene Betriebsbedingung (Operational Design Domain, ODD). Die ebenso in sechs Stufen unterteilte Definition des Verbands der Automobilindustrie e. V. (VDA) ist in Abbildung 2.5 dargestellt.

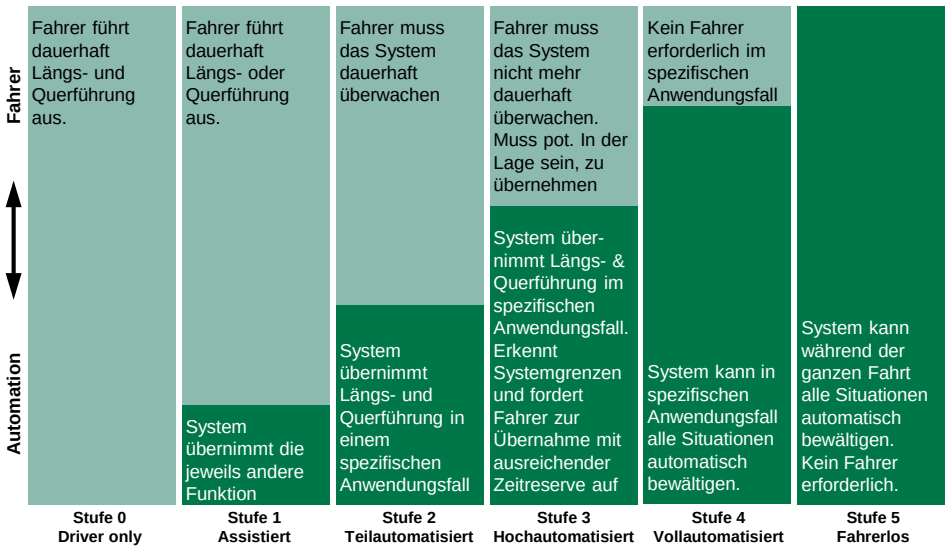


Abbildung 2.5: Stufen des automatisierten Fahrens entsprechend eines Expertengremiums des VDA [23]

Ein Fahrzeug mit Automatisierungsstufe 0 bietet weder für die Längs- noch für die

Querführung eine Automatisierung. In Stufe 1 kann eine der beiden Fahraufgaben (Dynamic Driving Task, DDTs) durch eine Automatisierung übernommen werden, man spricht vom assistierten Fahren. Werden beide DDTs gleichzeitig durch das Fahrzeug übernommen, jedoch vom Fahrer überwacht, spricht man auf Stufe 2 vom teilautomatisierten Fahren. Ab der dritten Stufe, dem hochautomatisierten Fahren, überwacht das Feature seine Systemgrenzen selbsttätig und der Fahrer dient lediglich als Rückfallebene. In Stufe 4, dem vollautomatisierten Fahren, kann das Fahrzeug innerhalb der ODD ohne die Rückfallebene Fahrer auskommen. Mit Stufe 5, dem fahrerlosen Fahren, kann die Automatisierung die Fahraufgabe dauerhaft und vollständig in allen Betriebsbedingungen übernehmen [24].

2.3 Wasserfallmodell

Royce beschreibt in seiner Veröffentlichung „Managing the development of large software systems“ [25] ein lineares Vorgehensmodell für die Entwicklung von Softwaresystemen, welches in der Folgezeit als Wasserfallmodell bekannt wurde [26], [27]. Der Begriff des Vorgehensmodells (engl. life cycle model) ist nach dem Standard ISO/IEC 12207 [28] wie folgt definiert:

Definition 2.11 (Vorgehensmodell)**.** *Ein* Vorgehensmodell *bezeichnet ein Rahmenkonzept von Prozessen und Aktivitäten, die sich mit dem Lebenszyklus befassen, in Stufen organisiert sein können und als gemeinsame Referenz für die Kommunikation und das Verständnis dienen.*

Die aufeinander folgenden Phasen des Wasserfallmodells strukturieren die Entwicklungstätigkeiten und erlauben den geplanten Einsatz unterschiedlicher Spezialisten entsprechend ihren Fachgebieten. Aufgrund der klar unterscheidbaren Tätigkeiten, Anforderungen und Zielsetzungen der einzelnen Phasen ermöglicht dieses Vorgehen laut Royce einen optimierten Einsatz von Projektressourcen. Iterationen sind innerhalb des Wasserfallmodells ursprünglich lediglich zwischen aufeinander folgenden Phasen vorgesehen, um Änderungen auf handhabbare Umfänge zu begrenzen. Abbildung 2.6 skizziert diese ursprüngliche Auslegeung des Wasserfallmodells.

Die zentralen Phasen des Modells sind die Analyse, während derer zur Zielerfüllung notwendige analytische Lösungen erarbeitet werden, und die Programmierung, in

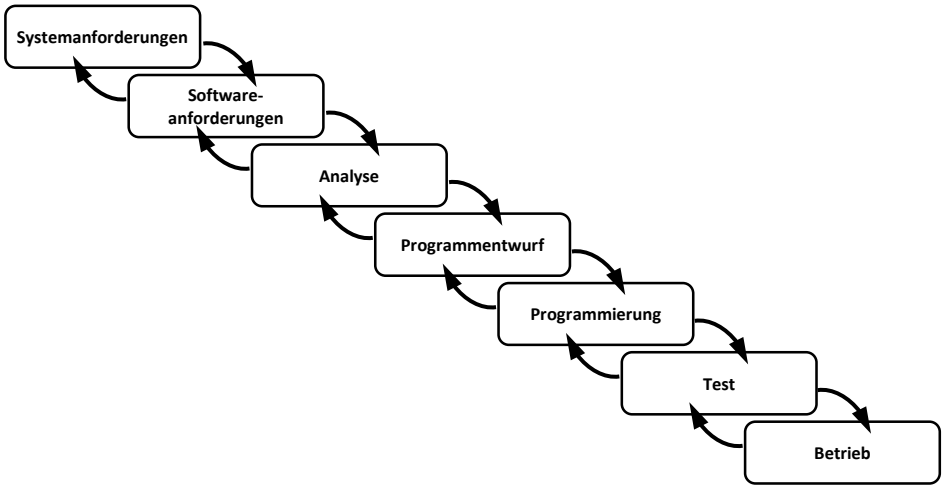


Abbildung 2.6: Ursprüngliche Struktur des Wasserfallmodells mit Iterationen zwischen aufeinander folgenden Phasen [25]

welcher das Programm implementiert und die Analyseergebnisse in den Programmablauf eingebettet werden. Vor der Analysephase werden im Wasserfallmodell in zwei aufeinander folgenden Phasen die Systemanforderungen und die Softwareanforderungen erarbeitet. Mit Abschluss dieser Phasen ist jederzeit eine Rückfallmöglichkeit gewährleistet, die die Arbeitsaufwände der vorhergehenden Projektphasen absichert. Die eigentliche Softwareprogrammierung wird durch eine explizite Phase für den Programmentwurf vorbereitet. Dieser Entwurf beinhaltet die finale Systemarchitektur, die Schnittstellendefinitionen sowie den Testplan. Nach Abschluss der Programmierung wird das Ergebnis in der Testphase entsprechend des Testplans überprüft. Sie ist laut Royce die ressourcenintensivste Phase des Wasserfallmodells. Nach Abschluss der Testaktivität und der daran gekoppelten Freigabe wird die Betriebsphase des Systems eingeleitet.

Aufgrund der späten Überprüfung essentieller System- und Softwarebestandteile, wie Laufzeiten, Speicherauslastung und Schnittstellen, beschreibt Royce dieses sequentielle Vorgehen als riskant. Da diese Bestandteile analytisch nicht präzise untersucht werden können, führen sie bei Nichterfüllung der Testphase zu grundlegenden Anpassungen am Programmentwurf und tendenziell zu notwendigen Änderungen der Softwareanforderungen. Können diese Änderungen nicht bereits in frühen Phasen abgefangen werden, entstehen durch die folgenden Änderungen im Softwareentwurf, der -implementierung und die erneuten Testaufwände hohe Kosten und Zeitverluste. Um dieses Risiko zu minimieren empfiehlt Royce die

Umsetzung von fünf ergänzenden Merkmalen.

Der **vorläufige Programmentwurf** wird in einer weiteren Phase zwischen der Erarbeitung der Softwareanforderungen und der Analysephase entwickelt. Dieser Entwurf wird fehlerhaft sein, kann aber bereits zu diesem frühen Zeitpunkt der Entwicklung Rückschlüsse auf das Laufzeit- und Speicherverhalten liefern. Daraus lassen sich für die folgende analytische Entwicklungsphase technische Rahmenbedingungen ableiten.

Eine **Ausführliche Dokumentation** trägt entscheidend zum Projekterfolg bei. Sämtliche Kommunikation zwischen Entwicklern, Management und Kunden sollte auf Basis einer formalen und schriftlichen Dokumentation durchgeführt werden, um jederzeit als eindeutige Basis Entscheidungen und Fortschrittsbewertungen zu gewährleisten. In frühen Projektphasen entspricht die Dokumentation direkt den Arbeitsergebnissen der Spezifikation und Analyse. Der höchste Mehrwert einer ausführlichen Dokumentation liegt in der Test- und der Betriebsphase. Sie ist grundlegend für eine optimierte Testdurchführung, den korrekten Betrieb der Software sowie für eine effektive Pflege und Wartung des Systems.

Das Prinzip der **doppelten Durchführung** (englisch Do-It-Twice) ermöglicht es den Entwicklern zunächst Erfahrung im Umgang mit einer Technologie oder einem System zu sammeln. Der Innovationsgrad eines Produkts ist ein entscheidender Faktor für das Gelingen der Entwicklung. Werden grundlegende technische Fragestellungen im Verlauf der Produktentwicklung originär beantwortet, sollte zunächst eine prototypische Entwicklung durchgeführt werden. Ziel dieser ist die Aufdeckung und Untersuchung eventueller Hindernisse und zugehöriger Lösungsalternativen. Dazu fokussieren sich die Entwickler auf die neuartigen Fragestellungen. Die Erfahrungen aus der prototypischen Entwicklung fließen im Anschluss in die regulären Entwicklungsphasen ein.

Die **Bedeutung der Testphase** wird häufig unterschätzt. Die Testphase bedarf einer geeigneten Planung, Steuerung und Überwachung. Test- und Absicherungsaufgaben sollten von Spezialisten durchgeführt werden und einen großen Anteil manueller Reviews enthalten. Ziel von numerischen Tests ist eine vollständige Testabdeckung der logischen Pfade.

Eine hohe Bedeutung nimmt die **Einbindung des Kunden** in den Entwicklungsprozess zwischen Anforderungsdefinition und Inbetriebnahme ein. Hierzu eignen sich Reviews, in denen der vorläufige Programmentwurf, kritische Softwarekomponenten

des finalen Entwurfs und die Testergebnisse gemeinsam überprüft werden.

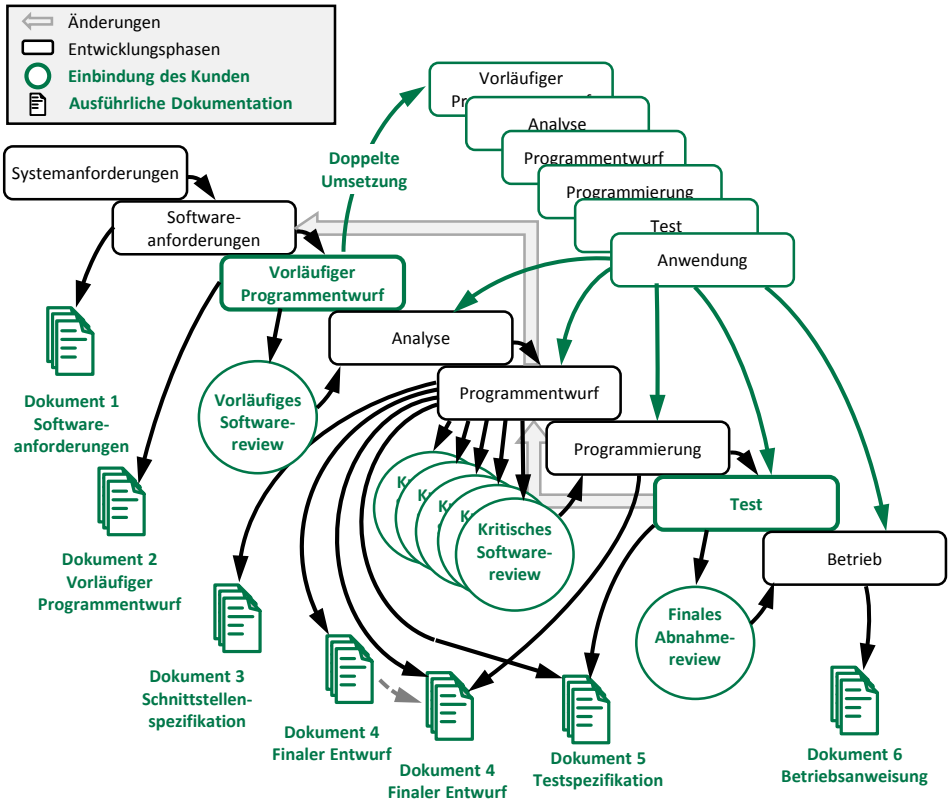


Abbildung 2.7: Wasserfallmodell zur Strukturierung der Entwicklung großer Softwareprojekte nach Royce [25] (Die fünf durch Royce ergänzten Merkmale sind in grün hervorgehoben)

Abbildung 2.7 stellt das um diese fünf Merkmale ergänzte Wasserfallmodell dar. Den Projektphasen sind die jeweils entstehenden Komponenten der Projektdokumentation zugeordnet, die prototypische Entwicklung mit ihrem Start im Anschluss an den vorläufigen Programmentwurf ist ergänzt und die Zeitpunkte für die Kundenreviews sind festgelegt.

2.4 Spiralmodell

Mit dem Spiralmodell spezifiziert Boehm [29] ein iteratives Vorgehensmodell als übergeordnete Grundlage für die Entwicklung von Softwareprodukten. Zentrales Element ist dabei ein umfassendes und fortlaufendes Risikomanagement [27], welches eine frühzeitige Identifikation möglicher Engpässe und Hindernisse sowie die Bewertung prototypisch umgesetzter Alternativlösungen ermöglicht. Das Spiralmodell bietet einen Rahmen für die Auswahl und den systematischen Einsatz verschiedener, etablierter Vorgehensweisen, Entwicklungsmethoden und -techniken. Der iterative Ablauf des Spiralmodells ist in Abbildung 2.8 dargestellt. Die vier Quadranten beschreiben die grundlegenden Phasen zur Identifikation von Zielen, Randbedingungen und Alternativen, zur Risikobewertung[27], zur Umsetzung und zur Planung. Der Drehwinkel veranschaulicht den Fortschritt im jeweiligen Entwicklungszyklus und der Spiralradius die damit verbundenen kumulativen Kosten. Am Abschluss jedes Zyklus wird ein Review durchgeführt, im Zuge dessen sich alle Beteiligten für die folgende Projektphase verpflichten oder das Projekt beenden. Dieses Review ist insbesondere zur Abstimmung mit Stakeholdern unverzichtbar [30].

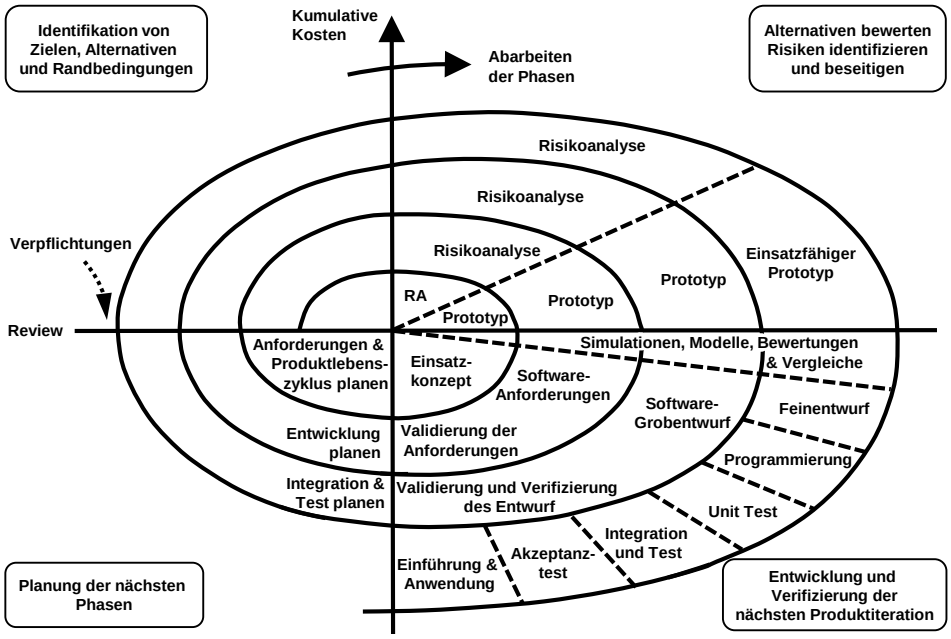


Abbildung 2.8: Spiralmodell für die Softwareentwicklung und -verbesserung nach Boehm [29]

Das Vorgehen innerhalb eines Iterationszyklus im Spiralmodell folgt dem beschriebenen grundlegenden Muster und wird entsprechend der Risikobewertung adaptiert. Ausgehend von der Identifikation des aktuellen Entwicklungsziels, potentieller Umsetzungsalternativen und der vorliegenden Rahmenbedingungen werden zu Beginn des Entwicklungszyklus mögliche Projektrisiken identifiziert und Realisierungsalternativen bewertet. Neben personellen Engpässen und unrealistischen Zeit- und Finanzplänen werden die technische Realisierbarkeit, unbekannte Nutzerwünsche und unerwartetes Nutzerverhalten sowie fehlerhafte Umsetzung der Implementierung und der Schnittstellen als wichtigste Risikofaktoren für Softwareprojekte genannt. Ziel ist die Definition notwendiger und geeigneter Handlungsstrategien zur Risikominimierung und zur Erreichung des jeweiligen Iterationsziels. Für die Validierung erarbeiteter Nutzerkonzepte und zur Bewertung neuer Technologien wird Prototyping-, Simulations- und Benchmarktechniken in frühen Projektphasen eine hohe Relevanz zugemessen. Wenn in späteren Projektphasen der Fokus auf die korrekte Umsetzung und Absicherung der validierten Konzepte wechselt, steht laut Boehm die Minimierung von Risiken durch fehlerhafte Implementierungen im Vordergrund. Dies kann durch die Integration eines strukturierten Vorgehensmodells, Boehm nennt das Wasserfallmodell, in den jeweiligen Iterationszyklen erreicht werden.

Das Spiralmodell stellt damit ein durch die Minimierung des Projektrisikos getriebenes agiles Vorgehensmodell dar. Die Positionen der einzelnen Methoden- und Handlungsbausteine in Abbildung 2.8 sind lediglich einer Tendenz entsprechend angeordnet. Die Anwendung und der Anwendungszeitpunkt einzelner Bausteine im Projektverlauf muss basierend auf der Risikobetrachtung spezifisch und individuell für jedes Projekt entschieden werden. Das Spiralmodell verzichtet explizit auf die Vorgabe eines linearen Vorgehens zur Produktentwicklung und bietet stattdessen die notwendigen Rahmenbedingungen zur Auswahl und Planung der Methoden- und Handlungsbausteine entsprechend den dominierenden Projektrisiken. Neben der damit gewonnen Agilität sieht Boehm die Vorteile des Spiralmodells insbesondere in der frühen Validierung technologischer Konzepte. Auftragsarbeiten, deren Ergebnisse bereits zu Beginn vertraglich geregelt sind, können vom Spiralmodell nur gering profitieren. Aufgrund der hohen Bedeutung der Risikoanalyse und -bewertung wirken sich dabei entstehende Fehler negativ auf den gesamten Entwicklungszyklus und im schlimmsten Fall auf den gesamten Projektverlauf aus.

2.5 V-Modell

Das V-Modell beschreibt in seiner ursprünglichen Form, wie das Wasserfallmodell, ein strukturiertes, sequentielles Vorgehensmodell. Es wurde definiert für die Entwicklung, Wartung und Weiterentwicklung von informationstechnisch geprägten Systemen der Verwaltung der Bundesrepublik Deutschland [31]. Das V-Modell in Form des Standardvorgehensmodells wurde seit seiner Veröffentlichung durch die Bundeswehr 1992 [32] mehrfach überarbeitet. Neben einer Verbesserung der Systemqualität und einer Kostenreduktion ist die Zielsetzung des V-Modells eine verbesserte und standardisierte Kommunikation zwischen den Vertragsparteien. Neben verschiedenen Behörden wird das V-Modell, da es „keine organisationsspezifischen Festlegungen“ [31] berücksichtigt, auch in industriellen Domänen angewandt. Im Bereich der Entwicklung eingebetteter Systeme [33] und der Automobilindustrie [19] dient das Submodell Systemerstellung des V-Modell 97 [31] häufig als Prozessreferenz. Arbeiten im Bereich Verifikation und Validierung [34], der funktionalen Sicherheit [35] und Security [36] liefert es eine Grundlage zur Einordnung der Arbeitsschritte und Konzepte in die bekannten und klar definierten Entwicklungsphasen.

2.5.1 V-Modell 97

In der Überarbeitung von 1997 definiert das V-Modell die während einer Entwicklung auftretenden Aktivitäten, die entstehenden Produkte inklusive deren Zustände sowie die beteiligten Rollen als Grundelemente. Das Vorgehensmodell ist dazu in die vier Submodule Systemerstellung (SE), Qualitätssicherung (QS), Konfigurationsmanagement (KM) und Projektmanagement (PM) untergliedert. Eine Übersicht über die Submodule und Grundelemente bietet Abbildung 2.9.

Der Kern des V-Modells, eine Erweiterung des Wasserfallmodells, ist im Submodul Systemerstellung (SE) beschrieben. Den Top-Down orientierten Dekompositionsphasen der Systementwicklung zur Spezifikation und Realisierung werden ihre zugehörigen Bottom-Up orientierten Prüf- und Integrationsphasen gegenübergestellt [37]. Diese im V-Modell vorgenommene Detaillierung spiegelt die bereits durch Royce vermittelte besondere Bedeutung der Testphase (Abschnitt 2.3) wider. Die durchzuführenden Aktivitäten werden auf die Systemebene und die Ebene der Software- und Hardwareeinheiten aufgeteilt. Die Systemerstellung erfolgt nicht linear, stattdessen sieht das V-Modell vor, „daß[sic] standardmäßig inkrementell

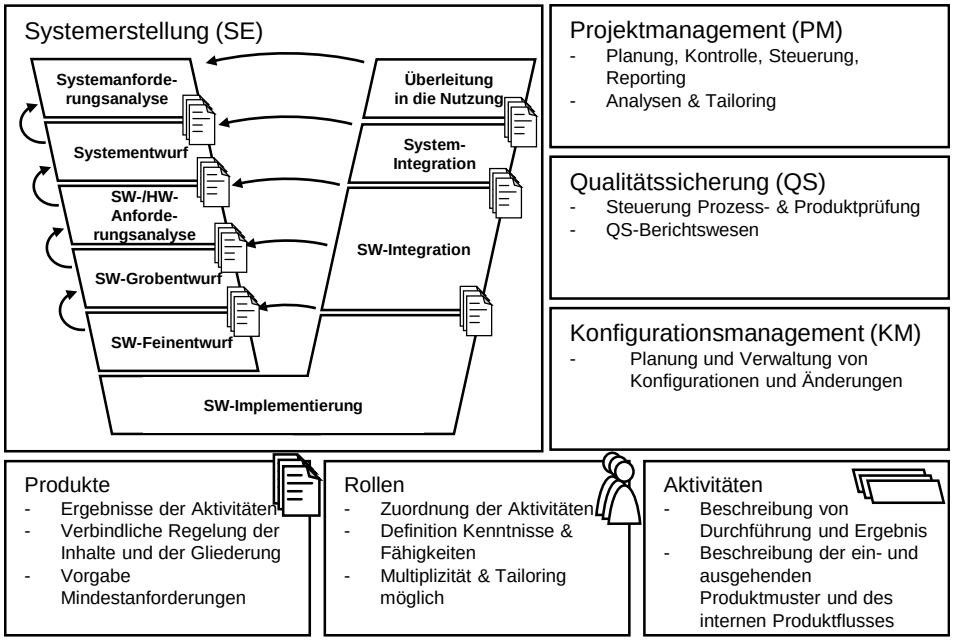


Abbildung 2.9: Der V-Modell 97 Softwareentwicklungsstandard der Bundesbehörden [31]

entwickelt wird“ [31].

Während im Submodul Systemerstellung (SE) in den Implementierungs- und Integrationsphasen eine Selbstprüfung der Arbeitsprodukte durch die SW-Entwickler und HW-Entwickler [31] vorgesehen ist, werden zur Produktfreigabe im Submodul Qualitätssicherung (QS) Prüfaufgaben durch dediziertes Personal vorgeschrieben [31]. Die Aktivitäten im rechten Schenkel des V-Modells prüfen entsprechend ihrer Abstraktionsebene die Umsetzung der im linken Schenkel erarbeiteten Spezifikationen. Die in Abschnitt 2.3 für das Wasserfallmodell geforderte ausführliche Dokumentation wird im V-Modell „durch Vorgabe verbindlicher Gliederungen“ [31] für die Produkte sichergestellt.

Im Submodell Konfigurationsmanagement (KM) werden Aktivitäten zur Steuerung und Überwachung von Änderungen und Varianten spezifiziert [31]. Dies beinhaltet die Verwaltung der Produkte, ihrer Zustände und Zugriffsrechte um eine ständige Nachvollziehbarkeit und eindeutige Zuordnung von Dokumenten, Produkten und Konfigurationen sicherzustellen.

Das Submodell Projektmanagement (PM) beschreibt übergeordnete Aktivitäten

zur Planung, Steuerung und Überwachung des Projektfortschritts. Dies schließt ein analog zum Spiralmodell (Abschnitt 2.4) periodisch durchzuführendes Risikomanagement sowie die projektspezifische Anpassung des V-Modells durch sogenanntes Tailoring [31] ein.

Definition 2.12 (Tailoring)**.** *Das* Tailoring *[engl. to tailor = schneidern] bezeichnet das Vorgehen, einen Entwicklungsprozess auf eine Organisationsstruktur oder ein Projekt anzupassen, bzw. maßzuschneidern.*

Während die Hauptprinzipien des Spiralmodells auch im V-Modell untergeordnet Anwendung finden, ist der Übergang zwischen Projektphasen nur mit vollständigen Arbeitsprodukten möglich (vergl. Aktivitätsbeschreibungen Systemerstellung [31]). Boehm identifiziert in dieser auch im Wasserfallmodell auftretenden Eigenschaft eine primäre Problemquelle für frühe Phasen in Softwareentwicklungsprojekten mit geringem Wissen über die technische Realisierbarkeit, die detaillierten Nutzerwünsche oder die benötigte Rechenleistung [29]. Das Risiko liegt in dieser Konstellation in hohen Zeit- und Arbeitsaufwänden für Spezifikationen, die aufgrund fehlerhafter Annahmen keinen oder lediglich einen geringen Nutzen bieten.

2.5.2 V-Modell XT

Erfahrungen aus dem Einsatz und im Umgang mit dem V-Modell 97 in Entwicklungsprojekten unterschiedlicher Domänen führten zur Weiterentwicklung des Vorgehensmodells zum V-Modell XT. Es wurde 2006 veröffentlicht [38] und liegt aktuell in der Version 2.0 vor [39]. Wichtige Ziele der Weiterentwicklung waren eine stärkere Fokussierung auf die projekt- und organisationsspezifische Anpassung durch Tailoring, die Integration eines Vorgehens zur Prozessverbesserung, die Ausweitung auf den gesamten Systemlebenszyklus und die Einbeziehung neuer Erkenntnisse aus Wissenschaft und Anwendung.

Die Grundelemente des V-Modells 97 wurden im V-Modell XT dahin gehend erweitert, dass die Aktivitäten (Version 2.0 spricht hier auch von Abläufen) um sogenannte Entscheidungspunkte erweitert wurden und die Anpassung des Modells durch Tailoring als viertes, eigenständiges Grundelement ergänzt wurde. Im Zuge dieser Umstellung wurde die Aufteilung des V-Modells in vier Submodule durch modulare Ablauf- und Vorgehensbausteine ersetzt. Vorgehensbausteine gruppieren

Aktivitäten entsprechend einer „konkreten Aufgabenstellung“ [39] und spezifizieren die mit den Aktivitäten assoziierten Produkte und Rollen. Die Vorgehensbausteine Projektmanagement, Qualitätssicherung, Konfigurationsmanagement und Problem- und Änderungsmanagement bilden den Kern des V-Modell-XT. Sie werden spezifisch entsprechend dem Ergebnis des Tailorings durch weitere Bausteine, wie beispielsweise Anforderungsfestlegung, Systemerstellung, Lieferung und Abnahme oder Messung und Analyse ergänzt [39]. Die Reihenfolge der Produkterstellung wird durch die Projektdurchführungsstrategie definiert. In ihr werden die vier Ablaufbausteine inkrementelle Systementwicklung, komponentenbasierte Systementwicklung, prototypische Systementwicklung und Unterauftrag integriert [39].

Der Ansatz des „extreme Tailoring“ bietet mit dem V-Modell eine sehr flexible Grundlage zur Entwicklung projekt- und unternehmensspezifischer Prozesse. Gleichzeitig stellt die damit verbundene Komplexität eine Einstiegshürde dar und erschwert das Verständnis und die Implementierung des Vorgehensmodells. Möglichen Missverständnissen in Lieferantenbeziehungen wird durch die Referenzierung von Projektdurchführungsstrategien entgegengewirkt [39].

2.6 Stage-Gate-Modell

Das Stage-Gate-Modell nach Cooper [40] repräsentiert ein lineares Prozessmodell mit der Zielsetzung einer verbesserten Steuerung und Verwaltung parallel verlaufender Innovations- und Entwicklungsprozesse [41]. Die Entwicklung wird dabei in mehrere Phasen (Stages) unterteilt in denen sachlogisch ähnliche Aktivitäten gruppiert werden. Meilensteine (Gates) synchronisieren die parallelen Aktivitäten während und am Ende der Phasen. Eine Phase besteht immer aus der Ausführung von Entwicklungsaktivitäten und der Analyse und Bewertung der erreichten Ergebnisse. Im darauf folgenden Gate wird basierend auf den Analyseergebnissen und der Bewertung eine Entscheidung zur Fortsetzung oder zum Abbruch des Projekts getroffen [40]. In diesem Kontext ist der Prozessbegriff nach dem Standard ISO/IEC 12207 [28] wie folgt definiert:

Definition 2.13 (Prozess)**.** *Ein* Prozess *ist ein Satz von zusammenhängenden oder interagierenden Aktivitäten, die Eingaben in Ausgaben umwandeln.*

Der Stage-Gate Prozess nach Cooper setzt sich aus fünf Meilensteinen und Phasen

zusammen. Er beginnt mit der Ideenfindung und einer ersten Umsetzungsentscheidung im ersten Gate. In der ersten Phase wird eine Sondierungsstudie (Scooping) umgesetzt. Ist diese erfolgversprechend folgt die Entwicklung eines Geschäftsmodells, die Produktentwicklung und eine Test- und Validierungsphase. Abgeschlossen wird der Stage-Gate Prozess in der fünften Phase mit der Markteinführung. In den Unternehmen werden diese fünf grundlegenden Stufen häufig erweitert und ergänzt. Unternehmensprozesse, die auf dem Stage-Gate-Modell beruhen, definieren bis zu 15 Phasen [42]. Abbildung 2.10 stellt mit dem „Integrated Defense Acquisition, Technology, & Logistics Life Cycle Management Framework“ [43] des amerikanischen Verteidigungsministerium einen detaillierten Stage-Gate basierten Entwicklungsprozess dar.

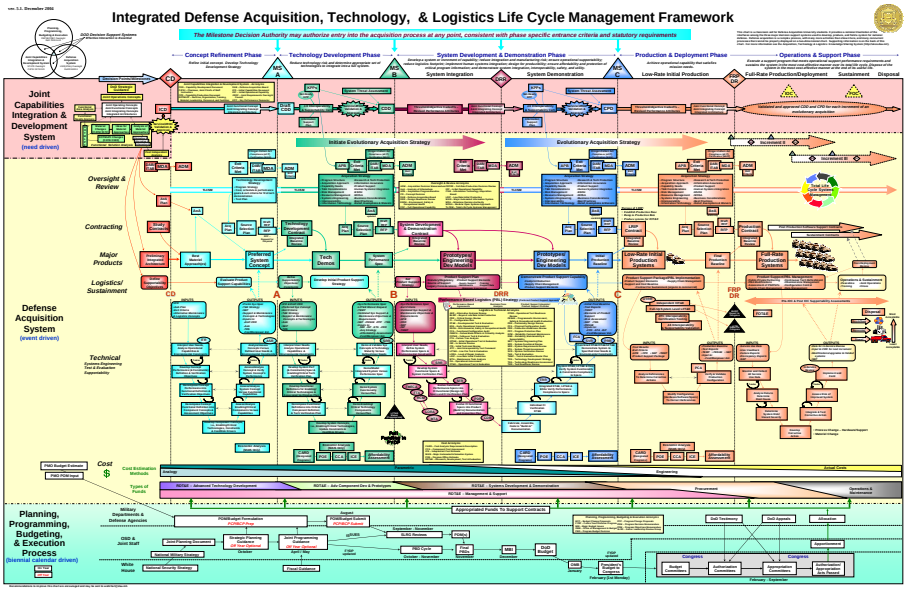


Abbildung 2.10: Beispiel für einen Stage-Gate Prozess: Integrated Defense Acquisition, Technology, & Logistics Life Cycle Management Framework des amerikanischen Verteidigungsministeriums mit sechs Phasen(stages), die durch Meilensteine(gates) begrenzt werden [43]

Im Bereich der deutschen Automobil- und Automobilzulieferindustrie basieren Engineeringprozesse häufig auf einem Stage-Gate Ansatz. Der Verband der Automobilindustrie e.V. (VDA) definiert sieben Meilensteine für den Projektablauf vom Projektauftrag, über die Freigaben der Grobentwicklung, der Detailentwicklung, der Detailplanung des Produktionsprozesses, der Beschaffung und Bereitstellung von Produktionsressourcen und der Serienproduktion zum Projektabschluss [44].

2.7 Automotive SPICE

Das Automotive SPICE[4] Prozess-Referenz- und Prozess-Assessment-Modell [45] wurde durch die Automotive Special Intrest Group (SIG) für die Bewertung von Entwicklungsprozessen in der Automobilindustrie definiert und wird durch den Arbeitskreis 13 des Qualitäts Management Center (QMC) des VDA weiterentwickelt. Das in Automotive SPICE beinhaltete Prozessreferenzmodell dient als Grundlage und Rahmenwerk für die produktbezogenen Entwicklungs- und Managementprozesse der Automobilentwicklung. Es definiert Prozesse mit Bezug auf die Produktentstehung im Bereich System- und Softwareentwicklung. Die Entwicklung von mechanischen oder Hardware-Komponenten ist nicht Teil des Referenzmodells. Die Integration der spezifischen Prozesse der Mechanik- und Hardwareentwicklung ist über das sogenannte „Plug-in"-Konzept vorgesehen [45]. Dieses ist wie folgt definiert:

Definition 2.14 (Plug-in Konzept)**.** *Das* Plug-in Konzept *beschreibt die Aufteilung der Entwicklung in die Domänen Systementwicklung, Softwareentwicklung, Hardwareentwicklung und Mechanikentwicklung sowie die damit verbundene Möglichkeit, die Automotive SPICE Prozessreferenz um nicht enthaltene Prozesse für Hardware und Mechanik zu ergänzen.*

Die referenzierten Prozesse werden in die drei Hauptkategorien primäre Prozesse des Produktlebenszyklus (Primary Life Cycle Processes), unterstützende Prozesse des Produktlebenszyklus (Supporting Life Cycle Processes) und organisatorische Prozesse des Produktlebenszyklus (Organizational Life Cycle Processes) unterteilt. Für jeden Prozess werden im Referenzmodell eine eindeutige ID, ein eindeutiger Name, die Zweckbestimmung des Prozesses sowie die angestrebten Ergebnisse spezifiziert. Zusätzlich werden für die Prozesse über grundlegende Verfahren (Base practices) und Arbeitsprodukte (Output work products) Umsetzungsindikatoren für die Prozessbewertung (Process Performance Indicators) geliefert [45]. Ihre überwiegende Erfüllung ist Voraussetzung für das Erreichen einer Automotive SPICE Level 1 Einstufung. Zur Bewertung der höheren Automotive SPICE Stufen werden Fähigkeitsindikatoren gemäß des Prozessbewertungsstandards ISO/IEC 33020 [46] in Form allgemeiner Verfahren (generic practices) und Mittel (generic resources) vorgegeben [45].

[4] **S**oftware **P**rocess **I**mprovement and **C**apability **D**etermination

Alle Aktivitäten mit direktem Bezug zur Produktentstehung werden in der Kategorie primäre Prozesse des Produktlebenszyklus zusammengefasst und in Prozessgruppen für Beschaffung (Acquisition Process Group (ACQ)), Auslieferung (Supply Process Group (SPL)), Systems Engineering (System Engineering Process Group (SYS)) und Software Engineering (Software Engineering Process Group (SWE)) unterteilt. Das Vorgehen der Systementwicklung orientiert sich am V-Modell [45]. Den Prozessen zur Systemdekomposition und -erstellung auf dem linken Schenkel werden entsprechende Prozesse zur Verifikation, Integration und Freigabe gegenüber gestellt. Automotive SPICE ermöglicht individuelle Hierarchieebenen zur Dekomposition des Systems. Abbildung 2.11 stellt die Prozessgruppen für Systems und Software Engineering dar und skizziert die im Rahmen dieser Forschungsarbeit genutzte und in Kapitel 2.2 eingeführte Abstraktionshierarchie (Systemebene, Subsytemebene, Komponentenebene und Einheitenebene).

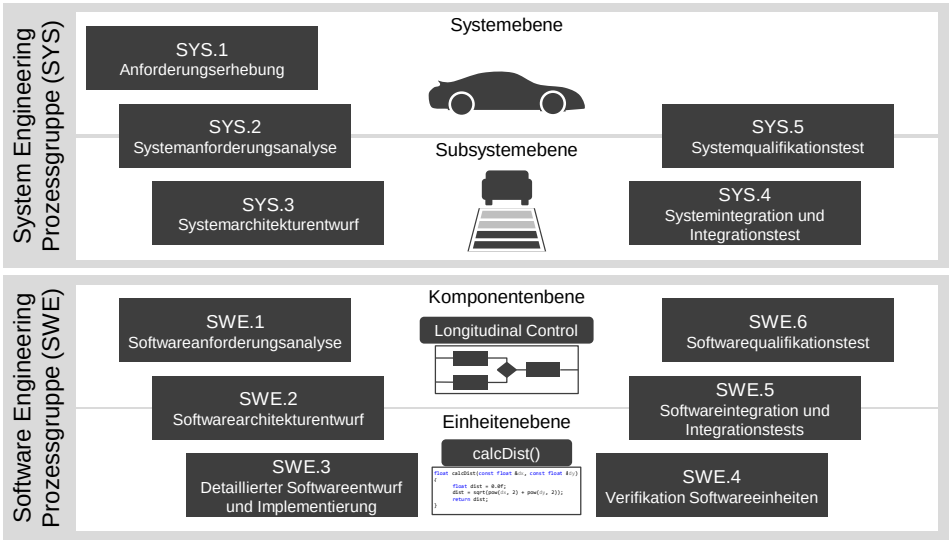


Abbildung 2.11: Darstellung der Automotive SPICE Prozessschritte und einer Unterteilung des Systems in die vier Abstraktionsebenen System, Subsystem, Komponente und Einheit am Beispiel eines Abstandsregeltempomaten

Auf die Anforderungserhebung (SYS.1 Requirements Elicitation) zu Beginn des Entwicklungsprojekts folgt eine Analysephase auf Systemebene (SYS.2 System Requirements Analysis). Aus dieser Analysephase wird der Systemarchitekturentwurf mit den beinhalteten Subsystemen abgeleitet (SYS.3 System Architecture Design). Nach Abschluss des Systemarchitekturentwurfs werden die Entwicklungspfade zwischen Mechanik, Hardware und Software getrennt.

In der Software Engineering Gruppe wird zunächst der Softwareanteil der Anforderungen analysiert (SWE.1 Software Requirements Analysis) und im Softwarearchitekturentwurf auf einzelne Komponenten abgebildet (SWE.2 Software Architectural Design). Der Architekturentwurf und die beinhalteten Komponenten werden im Folgenden verfeinert und in einzelnen Units umgesetzt (SWE.3 Software Detailed Design and Unit Construction). Auf Unit-Ebene wird die korrekte Umsetzung der in den Items implementierten Units nachgewiesen (SWE.4 Software Unit Verification). Im darauffolgenden Prozessschritt werden die einzelnen Items entsprechend der Architekturspezifikation zu Software-Komponenten integriert und die Konformität der Units und Items mit den spezifizierten Schnittstellen wird getestet (SWE.5 Software Integration and Integration Test). Der Softwarequalifikationstest (SWE.6 Software Qualification Test) weist als letzter Prozessschritt des Software Engineerings für die integrierte Software die Einhaltung und Übereinstimmung mit den spezifizierten Anforderungen nach.

Auf Systemebene werden die erstellten System-Items (Mechanik, Hardware und Software) entsprechend der Systemarchitektur zunächst in Subsysteme und schließlich das finale Gesamtsystem integriert und bezüglich der Einhaltung des spezifizierten Entwurfs und der Schnittstellen getestet (SYS.4 System Integration and Integration Test). Die Systementwicklung wird mit den Qualifikations- und Abnahmetests abgeschlossen (SYS.5 System Qualification Test). Diese weisen die Übereinstimmung mit den Systemanforderungen und die Bereitschaft zur Auslieferung des Systems nach.

Die primären Prozesse greifen auf die Querschnittprozesse der zweiten Prozesskategorie zurück. Diese umfassen die Produktentstehung begleitende Aktivitäten wie die Qualitätssicherung (Quality Assurance), die Dokumentation (Documentation), das Konfigurationsmanagement (Configuration Management) und das Änderungsmanagement (Change Request Management). Prozesse zur Steuerung der Entwicklungsprojekte (Management Process Group (MAN)), zur Prozessverbesserung (Process Improvement Process Group (PIM)) und zur Planung von Produktwiederverwendungen (Reuse Process Group (REU)) dienen vornehmlich dem Erreichen von Geschäftszielen und sind als übergeordnete, organisatorische Prozesse der Unternehmung zugeordnet. Die Managementgruppe definiert dabei die Zweckbestimmung und angestrebten Ergebnisse der Prozesse Projektmanagement, Risikomanagement und Berichtswesen.

Im Referenzmodell wurde in den definierten Prozessen besonderer Wert auf sogenannte Kernkonzepte gelegt [45]. Zwischen den Ergebnissen der einzelnen Prozesse

wird bidirektionale Nachverfolgbarkeit (Traceability) gefordert. Dies gilt insbesondere für spezifizierte Testfälle und die zugehörigen Ergebnisse der Testausführung sowie für Änderungen und die davon betroffenen Produkte. Die zugrundeliegende Terminologie [45] und ihre Verwendung in den Prozessbeschreibungen stellt die notwendige Widerspruchsfreiheit (Consistency) des Referenzmodells und abgeleiteter Prozesse sicher. Explizit wird die Bedeutung der Begriffe Evaluation, Verifikation, Test und Konformität (Compliance) im Kontext von Automotive SPICE spezifiziert. Evaluation bedeutet mögliche Umsetzungsalternativen in den Entwurfs- und Umsetzungsprozessen zu bewerten. Verifikation beschreibt die Überprüfung der Umsetzung auf Konformität gegenüber den spezifizierten Anforderungen und Entwürfen. Es wird zwischen Verifikation funktionaler System- oder Softwareanforderungen durch Qualifizierung und das Testen des entwickelten Quellcodes zur Sicherstellung der Konformität mit der Entwurfsspezifikation und nicht-funktionalen Änderungen unterschieden.

2.8 Agile Software Entwicklung

Die Agile Software Entwicklung fasst die Prinzipien mehrerer in den 1990er Jahren entstandener Softwareentwicklungsmethoden zusammen. Zu den bekanntesten Vertretern zählt neben Scrum [47] und Extreme Programming (XP) das Test Driven Development. Geprägt wurde der Begriff Agile Software Entwicklung durch das 2001 von Repräsentanten verschiedener agiler Methoden unerzeichnete Agile Manifest [48]. Motivation für die Formulierung und Unterzeichnung des Agilen Manifests war die Erkenntnis der Autoren, dass starre Entwicklungsprozesse, insbesondere das Festhalten an fehlerhaften Planungen und Zielen, übertriebene Anforderungen an die Dokumentation sowie ein fehlendes Bewusstsein und Verständnis für die Probleme der Softwareentwickler auf Managementebene einen entscheidenden Anteil an Kostensteigerungen und Verzügen von Softwareprojekten verantworten. Das Agile Manifest stellt das Tun in der Entwicklung vor Planung, Steuerung und Dokumentation. Projekterfolg wird nach agiler Interpretation primär durch die beteiligten Einzelpersonen und deren Kommunikation, durch die Fähigkeit auf Veränderungen zu reagieren und durch funktionierende Software sichergestellt.

Der Begriff der Methode findet vielfältig Anwendung und ist phonetisch mit dem Begriff der Methodik verwandt. Um ein eindeutiges Verständnis im Kontext dieser Arbeit sicherzustellen, werden die Begriffe im Folgenden definiert. Die allgemeine Bedeutung der Begriffe Methodik und Methode sind durch [49] definiert und der

Begriff (Software-)Methode folgt dem Standard ISO/IEC/IEEE 24765 [21].

Definition 2.15 (Methodik)**.** *Eine* Methodik *[griechisch methodik´ē (téchnē) = Kunst des planmäßigen Vorgehens] ist die Wissenschaft von der Verfahrensweise einer Wissenschaft, die Wissenschaft von den Lehr- und Unterrichtsmethoden und eine festgelegte Art des Vorgehens.*

Definition 2.16 (Methode)**.** *Eine* Methode *[griechisch mēthodos = Weg oder Gang einer Untersuchung, eigentlich = Weg zu etwas hin] ist ein auf einem Regelsystem aufbauendes Verfahren zur Erlangung von [wissenschaftlichen] Erkenntnissen oder praktischen Ergebnissen sowie die Art und Weise eines Vorgehens.*

Definition 2.17 ((Software-)Methode)**.** *Eine* (Software-)Methode *ist die Implementierung einer Operation, bzw. eine Anweisung bezüglich der Kombination von Werten zur Erreichung eines Ergebnisses.*

2.8.1 Scrum

Einer der prominentesten Vertreter der agilen Methoden ist Scrum. Die Autoren des Scrum Guide, Sutherland und Schwaber, beschreiben Scrum als „ein Rahmenwerk zur Entwicklung und Erhaltung komplexer Produkte“ [47]. In dieses Rahmenwerk lassen sich, wie auch beim Spiralmodell, verschiedene Vorgehen und Techniken der Produktentwicklung integrieren. Scrum stellt einen erkenntnistheoretischen, iterativen Ansatz der Prozesssteuerung dar, bei dem alle prozessrelevanten Entscheidungen auf gesammelten Erfahrungen beruhen. Grundlage für einen derartigen Ansatz sind Transparenz (Transparency) und gemeinsames Verständnis über das Vorgehen, die Ziele und die Fertigstellungskriterien (definition of „Done“), eine regelmäßige Sichtung und Prüfung der Arbeitsprodukte (Inspection) sowie die Anpassung der Arbeitsweise und -mittel (Adaptation) entsprechend während der Inspektion identifizierter Herausforderungen.

Scrum ist eine sehr schlanke Herangehensweise an die Produktentwicklung. Anstelle von detailliert spezifizierten Prozessschritten und Arbeitsprodukten wird Wert auf die Eigenverantwortung der einzelnen Mitglieder des Entwicklungsteams gelegt. Über einen fest definierten Zeitraum von üblicherweise 2-4 Wochen, dem sogenannten Sprint, wird eine Untermenge der zur Zielerreichung notwendigen Aufgaben

abgeschlossen. Jeder Sprint wird durch vier wiederkehrende und verpflichtende Termine strukturiert, der Sprint Planung, dem täglichen Scrum, dem Sprint Review und der Sprint Retrospektive. Dieses Vorgehen ist in Abbildung 2.12 dargestellt.

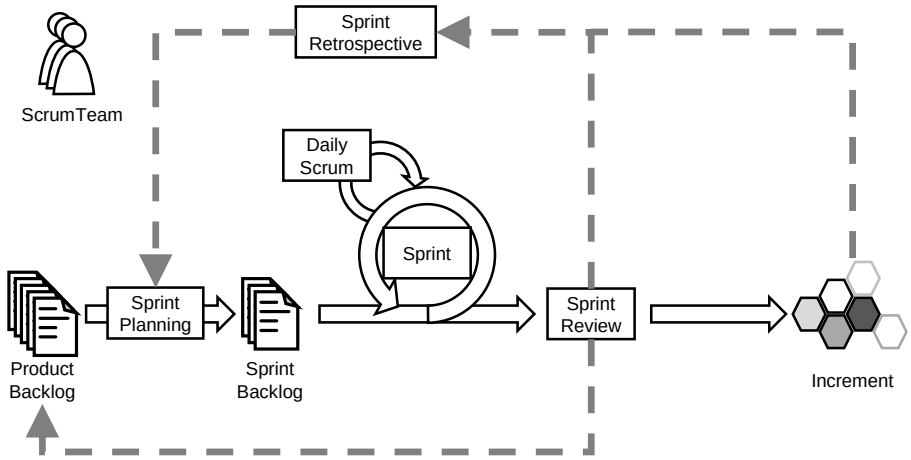


Abbildung 2.12: Scrum Framework für die Produktentwicklung nach [50]

Das Scrum Framework kennt lediglich drei Artefakte. Das Product Backlog listet alle Elemente, die das Produkt potentiell benötigt. Sie werden vom Product Owner priorisiert. Während der Sprint Planung wird durch das Entwicklungsteam eine Untermenge des Product Backlogs in das Sprint Backlog übernommen. Durch eine Aufwandsabschätzung, dem sogenannten Sprint Poker, wird dabei versucht die Untermenge an Elementen zu bestimmen, für die innerhalb des nächsten Sprints ein vollständiger Abschluss möglich ist. Im täglichen Scrum Meeting stimmt sich das Team über die am Vortag geleistete Arbeit, die für den jeweiligen Tag geplante Arbeit sowie potentielle Hindernisse ab. Am Ende des Sprints findet das Sprint Review statt, dessen Ziel die Besprechung und Prüfung der erreichten Inkremente und die Anpassung des Product Backlogs an neue Erkenntnisse ist. Das letzte Ereignis innerhalb eines Scrum-Zyklus stellt die Retrospektive dar. Sie dient dem Team sich selbst und das Vorgehen zu überprüfen und Verbesserungen im Arbeitsablauf und der Zusammenarbeit zu planen. Um eine Fokussierung auf das Wesentliche zu erreichen, schreibt Scrum für alle Treffen feste Zeitfenster vor. Das Sicherstellen einer regelmäßigen Durchführung der Termine, der Einhaltung der Zeitfenster und einer funktionierenden Arbeitsumgebung sowie die Förderung des Prozessverständnisses ist Aufgabe des Scrum Masters. Er besetzt damit eine koordinierende und vermittelnde Rolle innerhalb des Scrum Frameworks.

Scrum setzt auf selbstorganisierte und funktionsübergreifende Teams um das Ver-

ständnis und die Kommunikation zwischen verschiedenen Disziplinen zu fördern. Während Automotive SPICE [45] und V-Modell dedizierte Prozesse und Rollen [39] zur Prozessverbesserung vorsehen sind im iterativen Vorgehen des Scrums Mechanismen zur kontinuierlichen Verbesserung fester Bestandteil der Produktentwicklung. Anstelle von einzelnen Experten tragen alle an der Entwicklung beteiligten Personen in regelmäßigen Abständen zur Anpassung der Entwicklungsziele und zur Verbesserung des Vorgehens bei. Dem Scrum Master kommt dabei als Prozessexperte vor allem eine unterstützende Rolle zu.

2.9 Produktentstehungsprozess in der Automobilindustrie

Die Automotive SPICE Prozessreferenz stellt de facto einen Standard für die implementierten Entwicklungsprozesse der Fahrzeughersteller dar. Aus der Kombination des Stage-Gate-Modells mit dem in der Prozessreferenz festgehaltenen Vorgehen zur Entwicklung eingebetteter Systeme entstehen die Produktentstehungsprozesse (PEPs) der Erstausstatter (Original Equippment Manufacturer, OEMs) der Automobilindustrie. Abbildung 2.13 skizziert das vorherrschende Prozessschema aus Sicht der E/E-Entwicklung. Die allgemeine Spezifikation des PEP wird für unterschiedliche Baureihen parallel implementiert. Der PEP kann, wie dargestellt, in die vier Hauptphasen Produktstrategie, Funktionsentwicklung, Absicherung & Validierung und Überführung in die Wartung unterteilt werden [11] und erstreckt sich üblicherweise über einen Zeitraum zwischen 48 und 60 Monaten.

Definition 2.18 (Baureihe)**.** *Die* Baureihe *stellt einen Rationalisierungsansatz für Produktentwicklungen dar, bei denen dieselbe Funktion mit dem gleichen Lösungskonzept und möglichst gleichen Eigenschaften für einen breiteren Größenbereich zu erfüllen ist [9].*

Zum ersten Meilenstein, dem Projektstart, wird die prinzipielle Produktstrategie der Baureihe verabschiedet. Sie beinhaltet eine grobe Zielsetzung sowie eine Zeit- und Budgetvorgabe. Die Entwicklungsphase Produktstrategie dient der detaillierten Ausarbeitung des Funktionskonzepts und der Inhalte der jeweiligen Funktionen. Die anschließende Phase der Funktionsentwicklung beinhaltet die Prozessschritte und Aktivitäten zur Dekomposition des Systems. Mit dem Abschluss der Funktionsentwicklung wird ein sogenannter *Function-Freeze* durchgeführt. Ab diesem

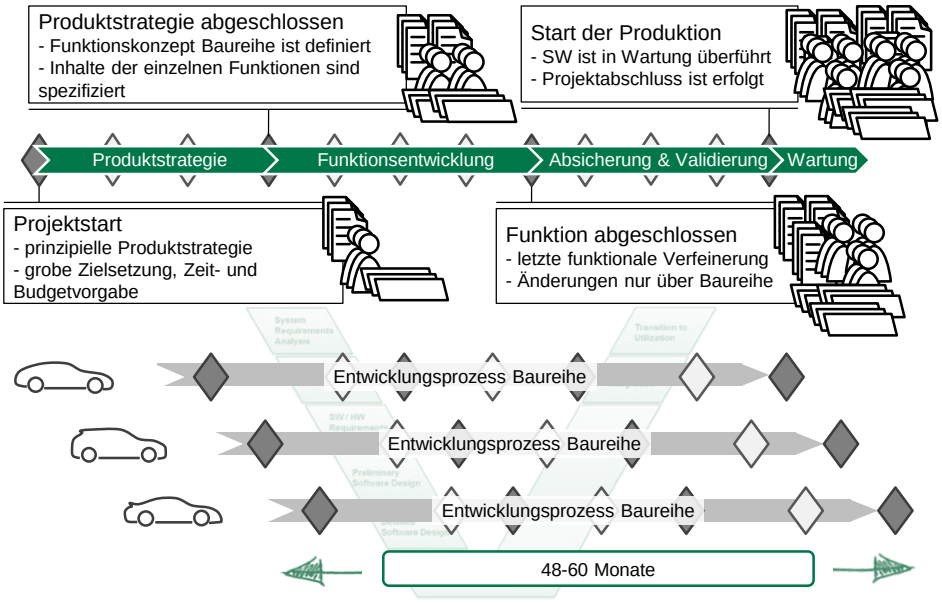


Abbildung 2.13: Prozessschema eines Automotive SPICE konformen Stage-Gate-PEP für verschiedene Fahrzeugbaureihen aus Sicht der E/E-Entwicklung. Gestaffelte Musterstände und entsprechende Releases sind aus Gründen der Übersicht nicht berücksichtigt.

Zeitpunkt sind lediglich letzte funktionale Verfeinerungen, wie Parameteranpassungen, zugelassen. Alle weiteren Änderungen können nur über die Prozesse des Baureihenänderungsmanagement umgesetzt werden. In der anschließenden Phase werden die für die Freigabe notwendigen Absicherungs- und Validierungstätigkeiten durchgeführt. Die einzelnen Funktionen und Komponenten werden mit dem Start der Produktion (Start of Production, SOP) in die Wartung überführt.

Das Schema skizziert den prinzipiellen Aufbau des PEP. In einem realen Prozess werden unterschiedliche Musterstände berücksichtigt. Für diese Stände existieren gestaffelte *Function-Freeze*-Meilensteine, deren funktionaler Umfang sukzessive ansteigt. Aus Gründen der Übersichtlichkeit und der Verständlichkeit wurden diese in der Abbildung abstrahiert.

2.9.1 Forschung und Vorentwicklung

Die lineare Struktur des PEP erlaubt aufgrund der Abhängigkeiten zwischen einzelnen Subsystemen und Komponenten nur geringe Abweichungen von der spezifizierten Produktstrategie und dem zugrundeliegenden Funktionskonzept. Daher können im Entwicklungsprozess der Baureihe lediglich Funktionen mit begrenzten Unsicherheiten und Realisierungsrisiken umgesetzt werden. Für die Analyse von disruptiven Technologien und Funktionsinnovationen werden dem PEP daher, wie in Abbildung 2.14 dargestellt, häufig Forschungs- und Vorentwicklungsprojekte vorgeschaltet [11, 15].

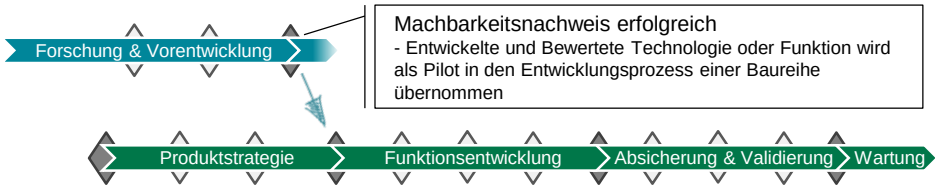


Abbildung 2.14: Forschung und Vorentwicklung innovativer Funktionen als dem PEP vorgelagerte Aktivität. Bei erfolgreichem Machbarkeitsnachweis wird das neue Konzept in den Produktentstehungsprozess einer Baureihe übernommen.

Ziel der Forschungs- und Vorentwicklungsprojekte ist die Analyse und Bewertung disruptiver Technologien und innovativer Funktionskozepte. Neben der Führung eines Machbarkeitsnachweises ist eine wichtige Aufgabe der Forschung und Vorentwicklung Erfahrungen im Umgang mit den neuen Technologien, Konzepten und Algorithmen zu sammeln. Diese ermöglichen die präzise und eindeutige Erhebung, Spezifikation und Analyse der Anforderungen in den nachgelagerten und im PEP realisierten Prozessschritten SYS.1, SYS.2 und SWE.1 des Automotive SPICE Referenzmodells (Abschnitt 2.7). Ist der Machbarkeitsnachweis einer innovativen Technologie oder Funktion erfolgreich und stellt einen klaren Kundenmehrwert dar, wird die Innovation auf Basis einer Pilotumsetzung innerhalb einer Baureihe in die Serienentwicklung überführt. Dazu wird die im Rahmen eines Vorentwicklungsprojekts eingesetzte innovative Technologie durch die relevanten Stakeholder aktueller Baureihenentwicklungen anhand des umgesetzten Prototypen bewertet. Fällt diese Bewertung positiv aus, wird die Technologie als neue Kundenfunktion in die Produktstrategie einer aktuellen Entwicklung übernommen.

2.10 Fazit

Die Prozesse der Automobilindustrie sind durch die Entwicklung von mechanischen Komponenten geprägt. Dies wird auch in der Verteilung von Zuständigkeiten und Verantwortungen, sowie der Organisationsstruktur sichtbar [11]. Durch das schnelle Wachstum Software getriebener Innovationen steigt die Produktkomplexität der E/E-Anteile aktueller und zukünftiger Fahrzeuge rasant an [6]. Diese Zunahme erfordert eine Erweiterung des Vorgehens und der Prozesse in der Automobilentwicklung.

Die stetige Innovation etablierter Features in der Automobilentwicklung ermöglicht eine detaillierte Abschätzung der Entwicklungsumfänge und eine klare Anforderungsspezifikation zu Beginn der Entwicklung. Disruptive Innovationen, wie sie durch die Digitalisierung und Elektrifizierung im Automobilsektor stattfinden, erschweren die eindeutige Spezifikation der Kundenanforderungen und erfordern eine frühzeitige Validierung [14].

Definition 2.19 (Disruptive Technologie)**.** *Eine* disruptive Technologie *[engl. to disrupt = unterbrechen] ist eine Innovation, die eine bestehende Technologie, ein bestehendes Produkt oder eine bestehende Dienstleistung möglicherweise vollständig verdrängt [51].*

Agile Methoden und Ansätze motivieren sich aus der höheren Flexibilität von Softwaresystemen bezüglich der Auslieferung von verbesserten Releases. In den frühen Phasen der Funktionsentwicklung ist das Wissen über das System und die Erwartungen der Nutzer mit Unsicherheiten verbunden. Die sukzessive Entwicklung vollständiger Inkremente ermöglicht durch eine regelmäßige Priorisierung und Anpassung der Entwicklungsaufwände diese Unsicherheiten zu kompensieren. Dieses Vorgehen ist in der reinen Softwareentwicklung sehr erfolgreich. Den Anforderungen der Entwicklung eingebetteter Systeme im Automobilbereich in Bezug auf die benötigte Abstimmung zwischen verschiedenen Zuständigkeiten und Domänen, der Planungssicherheit der Lieferketten und der fest definierten Entwicklungsziele der einzelnen Baureihen werden sie jedoch nicht vollständig gerecht. Die stufenweise Integration der Software in die Hardware und Mechatronik erfordert definierte Synchronisierungspunkte. Ohne zuverlässige Zusagen wird die Entwicklung abhängiger Systemfunktionen gegebenenfalls blockiert. Die synchronisierte Herangehensweise erleichtert zudem die Interaktion mit den Zulieferern.

Um der Forderung des Spiralmodells nach einer frühzeitigen Validierung der Kundenanforderungen gerecht zu werden, ist in den frühen Phasen der Produktentwicklung eine Erweiterung der Prozesse und der vorhandenen Methoden notwendig. Dem aktuellen PEP vorgelagerte und in frühen Phasen parallele Prozessschritte für innovative Funktionsentwicklungen könnten die benötigte Flexibilität einbringen. Zusätzliche ergänzende Methoden und Werkzeuge sollten dabei die eigentliche Zielsetzung der Validierung des Funktionskonzeptes vor der Übernahme in die Baureihe unterstützen.

3 Entwicklung, Verifikation und Validierung von Softwaresystemen für Fahrzeuge

Neben einem auf das Gesamtprodukt, in diesem Fall das Fahrzeug, zugeschnittenen Systementwicklungsprozess erfordert eine erfolgreiche Entwicklung eingebetteter Softwaresysteme spezifisch für die unterschiedlichen Systemkomponenten und aufeinanderfolgenden Entwicklungsphasen ausgewählte und angepasste Entwicklungs-, Verifikations- und Validierungsmethoden. Neben der Umsetzung der jeweiligen Arbeitsschritte einer Entwicklungsphase muss gewährleistet sein, dass die Zielerreichung und Korrektheit des Arbeitsergebnisses geprüft werden kann.

Ein wichtiges Merkmal zur Auswahl geeigneter Methoden ist dabei der grundlegende Charakter eines Features (vergleiche Kapitel 4.1). Das Blockschaltbild in Abbildung 3.1 stellt die logische Systemsicht der Integration eines Features in ein Fahrzeug dar.

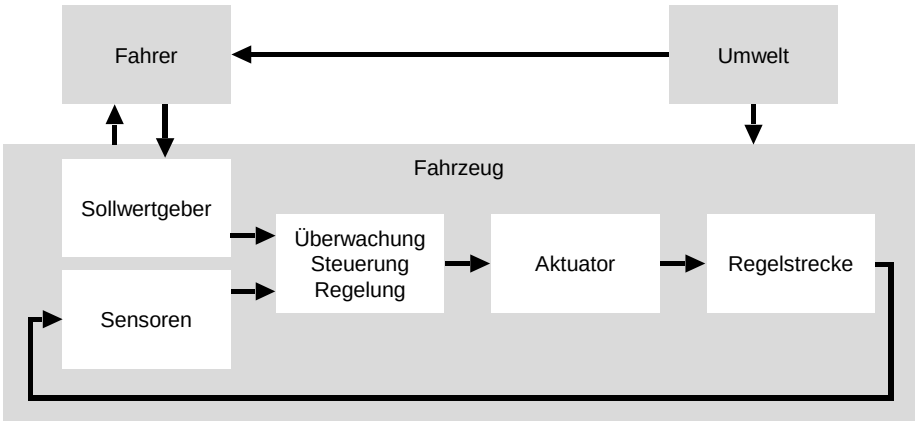


Abbildung 3.1: Integration eines E/E-Features aus logischer Systemsicht nach [14]

Aus steuerungs- und regelungstechnischer Sicht können Features eine überwachende, steuernde oder regelnde Aufgabe erfüllen [14]. Überwachende Features wirken

über Anzeigeelemente auf den Fahrer. Steuernde Funktionen arbeiten in einem offenen Regelkreis. Die Ansteuerung der Aktuatoren durch die Funktion hat keine unmittelbare Auswirkung auf die Sensoreingänge der Funktion. Ein gutes Beispiel für ein steuerndes Assistenzsystem stellt eine automatisierte Fernlichtfunktion dar [52]. Eine unmittelbare Auswirkung der Systemfunktion auf die Regelstrecke und damit auf die Sensoren, wie in Abbildung 3.1 dargestellt, liegt nur für eine regelnde Funktion innerhalb eines geschlossenen Regelkreises vor.

Definition 3.1 (Steuerung)**.** *Unter* Steuerung *wird die zielgerichtete Beeinflussung eines dynamischen Systems bezeichnet. [53]*

Definition 3.2 (Regelung)**.** *Eine* Regelung *bezeichnet im Ingenieurswesen die Überwachung der Systemausgabe, um diese mit dem erwarteten Ausgang zu vergleichen und Korrekturmaßnahmen zu ergreifen, falls die tatsächliche Ausgabe nicht mit der erwarteten Ausgabe übereinstimmt. [21]*

In diesem Kapitel werden zunächst Definitionen der Begriffe Qualität, Verifikation, Validierung, Fehler und Test eingeführt. Im Abschnitt 3.3 werden in der Automobilentwicklung verbreitete Methoden und Ansätze vorgestellt. Die darauf folgenden Abschnitte bieten einen Überblick über die gängigen statischen und dynamischen Verifikationsverfahren und die Verfahren zur Validierung von Software- und E/E-Systemen im Fahrzeug. Der Abschnitt 3.7 führt Methoden zur Erstellung von Testfällen und zur Ermittlung der Testtiefe ein und schließt das Kapitel ab. In einem Resümee wird die Eignung der Methoden für prädiktive Fahrzeugregelungsfunktionen diskutiert und es werden offenen Herausforderungen herausgearbeitet.

3.1 System- und Softwarequalität

Software- und Systementwicklung findet nach Automotive SPICE (Abschnitt 2.7) ihren Abschluss mit den Qualifikationstests. Der Qualitätsbegriff im Kontext des Systems und Software Engineering wird durch den Standard ISO/IEC/IEEE 24765 wie folgt definiert:

Definition 3.3 (Qualität)**.** *Die* Qualität *beschreibt die Fähigkeit eines Produkts, eines Dienstes, eines Systems, einer Komponente oder eines Prozesses, Kunden- oder Benutzerbedürfnisse, Erwartungen oder Anforderungen zu erfüllen und den Grad, in dem spezifizierte Anforderungen erfüllt werden.*

Eine detailliertere Definition ist mit den Qualitätsanforderungen und -kriterien zur Bewertung von Systemen und Softwareprodukten des Standards ISO/IEC 25010 [54] spezifiziert. Die Norm definiert zwei Qualitätsmodelle, ein Modell spezifiziert Kriterien zur Ermittlung der Nutzungsqualität und das zweite Modell definiert Kriterien für die externe und interne Qualität. Letzteres beinhaltet folgende acht Kriterien und zugehörige Unterkategorien:

- Funktionalität: Die Vollständigkeit, Angemessenheit und Richtigkeit der Umsetzung spezifizierter und implizit erwarteter Anforderungen
- Leistungsfähigkeit und Effizienz: Die aufgewendeten Ressourcen, das Zeitverhalten und die Kapazität bei der Erfüllung der Anforderungen.
- Kompatibilität: Die Eignung eine Umgebung mit weiteren Systemen zu teilen oder mit diesen zusammenzuarbeiten.
- Usability: Der Umfang in dem das System oder die Software seine Nutzer bei der Zielerreichung unterstützt.
- Zuverlässigkeit: Der Reifegrad und die Verfügbarkeit, Fehlertoleranz und Fähigkeit sich nach einem Fehlerfall zu regenerieren.
- Sicherheit: Die nachweisbare Gewährleistung von Vertraulichkeit, Integrität und Authentizität.
- Wartbarkeit: Der Grad der Modularisierung, Wiederverwendbarkeit, Analysierbarkeit, Modifizierbarkeit und Testbarkeit.
- Übertragbarkeit: Die Fähigkeit zu Anpassung, Integration und Austausch.

Das Modell zur Nutzungsqualität definiert die Kriterien Effektivität, Effizienz, Zufriedenheit, Risikofreiheit und Abdeckung des Anwendungsbereichs. Ein System oder Softwareprodukt soll nach dieser Definition den Nutzer bei einer möglichst vollständigen Zielerreichung (Effektivität) unter Einsatz möglichst geringer Aufwände

(Effizienz) unterstützen. Die Nutzerzufriedenheit ist definiert als das Zusammenspiel aus Zweckmäßigkeit, Nutzungskomfort und dem Vertrauen und der Freude im Umgang mit dem System oder der Software. Die Risikofreiheit des Produkts bezieht sich auf die Fähigkeit zur Minimierung möglicher Wirtschafts-, Gesundheits- und Umweltschäden. Das Kriterium Abdeckung des Anwendungsbereichs misst die benötigte Flexibilität und Eignung eines Produkts, die zur Erfüllung des gewünschten Anwendungszwecks benötigt wird.

3.2 Verifikation, Validierung und Testen

Der Einsatz spezifischer Methoden in der Funktionsentwicklung unterstützt die Einhaltung von Zeit- und Budgetplänen, erhöht die Qualität und trägt zu einer substantiellen Risikominimierung bei. Neben der eigentlichen Funktionsentwicklung sind für den Erfolg eines Entwicklungsprojekts insbesondere die Verifikation und Validierung von großer Bedeutung. Die Interpretation dieser Begriffe unterliegt einem gewissen Spielraum. Im Bereich der Softwareentwicklung wird häufig Boehm zitiert. Er interpretiert die Bedeutung von Validierung und Verifikation in „Software Risk Management“[27] mit den Worten:

- *Verification. “Am I building the product right?“*
- *Validation. “Am I building the right product?“*

Eine detailliertere, deutschsprachige Definition dieser Begriffe aus der Sicht der Automobilentwicklung liefert Schäuffele [14]:

Definition 3.4 (Verifikation)**.** *Die* Verifikation *ist der Prozess zur Beurteilung eines Systems oder einer Komponente mit dem Ziel festzustellen, ob die Resultate einer gegebenen Entwicklungsphase den Vorgaben für diese Phase entsprechen. Software-Verifikation ist demnach die Prüfung, ob eine Implementierung der für den betreffenden Entwicklungsschritt vorgegebenen Spezifikation genügt.*

Definition 3.5 (Validierung)**.** *Die* Validierung *ist der Prozess zur Beurteilung eines Systems oder einer Komponente mit dem Ziel festzustellen, ob der Einsatzzweck oder die Benutzererwartungen erfüllt werden. Funktionsvalidierung ist demnach die Prüfung, ob die Spezifikation die Benutzeranforderungen erfüllt, ob überhaupt die Benutzerakzeptanz durch eine Funktion erreicht wird.*

Während in mechanischer und elektronischer Hardware Fehler und Mängel neben fehlerhafter Spezifikation und Umsetzung auch durch Alterung oder Verschleiß auftreten können, liegt die Ursache für mangel- oder fehlerbehaftete Software immer in der Entwicklung. Die „Ursachenkette für Fehler“ nach [55] ist in Abbildung 3.2 dargestellt. Eine Fehlerhandlung führt zu einem Fehlerzustand, beispielsweise eine fehlerhafte Anforderung oder eine Fehlerstelle im Programmcode, welcher zu einer nach außen sichtbaren Fehlerwirkung führt.

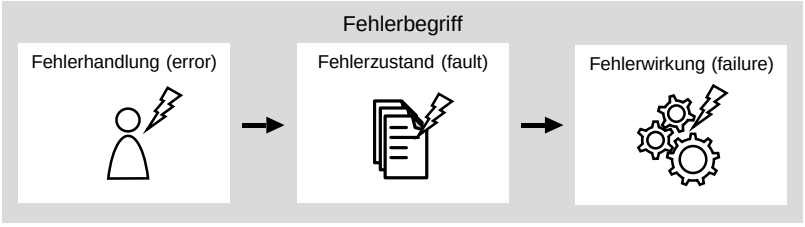


Abbildung 3.2: Der Fehlerbegriff im Deutschen [55] und im Englischen [56]

Nach [21] definieren sich die Fehlerbegriffe wie folgt:

Definition 3.6 (Fehlerhandlung)**.** *Die* Fehlerhandlung *ist eine menschliche Aktion, die ein falsches Ergebnis erzeugt, wie z. B. Software, die einen Fehlerzustand enthält.*

Definition 3.7 (Fehlerzustand)**.** *Der* Fehlerzustand *ist die Manifestation einer Fehlerhandlung in einer Software oder ein Defekt in einem Hardwaregerät oder in einer Komponente.*

Definition 3.8 (Fehlerwirkung)**.** *Die* Fehlerwirkung *ist die Beendigung der Fähigkeit eines Produkts, eine erforderliche Funktion auszuführen oder seine Unfähigkeit, innerhalb vorher festgelegter Grenzen seinen Zweck zu erfüllen.*

Aus der Definition der Begriffe Verifikation und Validierung kann der üblicherweise weit gefasste Begriff des Testens abgeleitet werden. In Bezug auf die Verifikation und Validierung sind insbesondere die Aktivitäten Debugging, Testen und Erproben zu unterscheiden. Unter Debugging versteht man eine üblicherweise unsystematische Verifikation durch die Entwickler während der Implementierung der Software und die Analyse der Software zur Lokalisierung von Fehlerursachen, die identifizierten Fehlerzuständen zugrunde liegen. „Testen impliziert eine systematische Herangehensweise“[19]. Beim Testen werden Fehler in der Umsetzung aufgedeckt, die nach Spillner und Linz [55] als „die Nichterfüllung einer festgelegten Anforderung“ definiert sind. Die Unterscheidung zwischen Entwicklung & Debugging und systematischem Testen bilden sich im Automobilsektor in den dedizierten Rollen Entwickler und Tester ab [19]. Nach [21] sind Debugging und Testen wie folgt definiert:

Definition 3.9 (Debugging)**.** Debugging *dient dem Aufspüren, Lokalisieren und Korrigieren von Fehlerzuständen innerhalb eines Computerprogramms.*

Definition 3.10 (Testen)**.** *Das* Testen *beschreibt eine Aktivität, während der ein System oder eine Komponente unter spezifizierten Bedingungen ausgeführt wird, die Ergebnisse dieser Ausführung beobachtet oder aufgezeichnet werden und ein Aspekt des Systems oder der Komponente bewertet wird.*

Ziel der Validierung ist die Identifikation fehlerhaft spezifizierter oder interpretierter Anforderungen. Das Resultat einer fehlerhaften Anforderungsspezifikation wird als Mangel bezeichnet. Der Begriff Mangel wird durch [55] wie folgt definiert:

Definition 3.11 (Mangel)**.** *Ein* Mangel *ist die Nichterfüllung einer angemessenen Erwartung oder die Beeinträchtigung der Verwendbarkeit bei gleichzeitiger Erfüllung der Funktionalität.*

Der Erprobung in Form von Testfahrten (Abschnitt 3.6.2 & 3.6.3) kommt in der Fahrzeugentwicklung eine wichtige Rolle zu, da „es häufig eher subjektive Wahrnehmungen von Menschen sind, die letztlich auch den Markterfolg eines Fahrzeugs ausmachen (Stichwort: PoPo-Sensor)“ [19]. Die Erprobung dient damit insbesondere der Aufdeckung von Mängeln. Um neben der gesamtheitlichen Analyse auf Systemebene durch Testfahrten auf Testgeländen und im öffentlichen Straßenverkehr eine

systematische Analyse unter exakt definierten und reproduzierbaren Bedingungen zu ermöglichen, werden in der Automobilbranche in allen Entwicklungsphasen und auf allen Integrationsebenen vermehrt simulationsgestützte Ansätze verfolgt (siehe Abschnitt 3.3.4 und 3.5.2). Der Begriff der Erprobung wird durch [19] wie folgt definiert:

Definition 3.12 (Erprobung)**.** *Die* Erprobung *ist eine Analyse des Gesamten, der eine vergleichsweise geringe Systematik zugrunde liegt. In Erproben steckt das Ausprobieren in einer unbestimmten Situation.*

Test und Erprobung beschreiben Methoden, die durch eine stichprobenhafte[1] Ausführung der Software Fehlerwirkungen und damit die zugrunde liegenden Fehlerzustände aufdecken sollen. Die beiden Methoden unterschreiden sich primär in der zugrunde liegenden Systematik. Daneben decken statische Verifikationsmethoden Fehlerzustände rein analytisch durch die Begutachtung von Spezifikation und Quelltext auf [10]. Die Anwendung von Metriken auf die Ergebnisse der Verifikation und Validierung ermöglicht eine objektive Bestimmung der Softwarequalität. Dies dient der Risikobewertung für den jeweiligen Einsatz der Software und fördert das Vertrauen in das Softwareprodukt [55].

Die Aktivitäten und das systematische Vorgehen zur Bestimmung der System- oder Softwarequalität werden in einem Testprozess spezifiziert. Der Testprozess ist nach [57] wie folgt definiert:

Definition 3.13 (Testprozess)**.** *Der* Testprozess *gibt Auskunft über die Qualität eines Softwareprodukts und besteht aus einer Reihe von Aktivitäten, die in einem oder mehreren Teilprozessen gruppiert sind.*

Der Software Testing Standard ISO/IEC/IEEE 29119-2 [58] spezifiziert einen organisationsbezogenen Testprozess, einen verwaltenden Testprozess und einen dynamischen Testprozess. Diese Prozesse sind in Abbildung 3.3 dargestellt. Aufgabe des organisationsbezogenen Prozesses ist die Erstellung, Überwachung und Pflege einer organisationsweit einheitlichen Teststrategie. Auf Projektebene regelt der verwaltende Testprozess die Erstellung des Testplans, prüft und steuert dessen

[1] Im Kontext des dynamischen Softwaretests beschreibt die Stichprobe, die für die Tests ausgewählten Eingangswerte und Zustände. Abschnitt 3.7.1 beschreibt Methoden zur systematischen Spezifikation dieser Stichprobe in Form von Testfällen.

Umsetzung und schließt die Testaktivitäten am Ende der Entwicklung ab. Der dynamische Prozess ist in vier Schritte untergliedert und beschreibt das operative Testvorgehen im Rahmen der Produktentwicklung. Die Teilprozesse Testenturf & -implementierung und Testausführung werden durch den Teilprozess Bereitstellung und Wartung der Testumgebung unterstützt. Abgeschlossen wird der dynamische Testprozess mit der Analyse und der Dokumentation der Testergebnisse und der Berichterstattung aufgetretener Ereignisse.

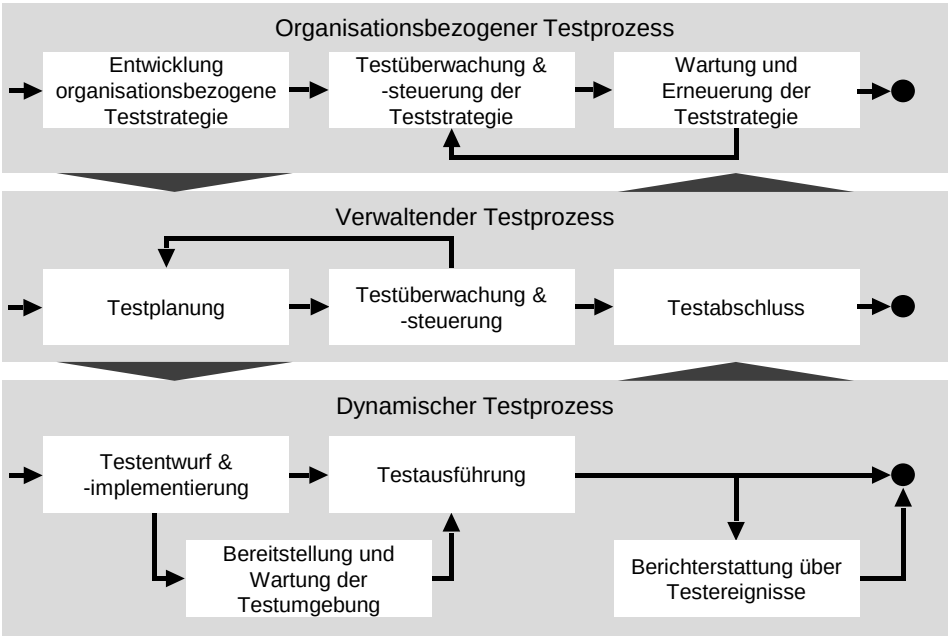


Abbildung 3.3: Testprozesse des Software Testing Standard ISO/IEC/IEEE 29119 [58]

Ein Testfall dient dem systematischen Nachweis einer Anforderungserfüllung. Die auszuführenden Testaktivitäten werden als inhaltlich gruppierte Testfälle im Testplan spezifiziert. Die Planung basiert auf der aus dem organisationsbezogenen Testprozess abgeleiteten Teststrategie. In ihr sind die anzuwendenden Testmethoden und die anzustrebende Testtiefe definiert. Die aufgeführten Begriffe sind nach ISO/IEC/IEEE 29119 [57] wie folgt definiert:

Definition 3.14 (Testfall)**.** *Ein* Testfall *umfasst die notwendigen Voraussetzungen, Eingaben (inklusive evtl. Aktionen)und erwarteten Ergebnisse, die entwickelt wurden, um ein Testobjekt auszuführen und zu prüfen, ob es die Testziele (Korrekte Implementierung, Identifikation von Fehlern, Qualitätsprüfung und weitere geschätze Informationen) erreicht.*

Definition 3.15 (Testplan)**.** *Der* Testplan *ist eine detaillierte Beschreibung der zu erreichenden Prüfziele und der Mittel und dem Zeitplan für deren Erreichung. Er wird ausgestaltet, um die Prüfungsaktivitäten eines Testobjekts oder einer Gruppe von Testobjekten zu koordinieren.*

Definition 3.16 (Testspezifikation)**.** *Die* Testspezifikation *ist die vollständige Dokumentation des Testentwurfs, der Testfälle und Testverfahren für ein bestimmtes Testobjekt.*

Die wiederholte Ausführung von Testfällen zur Prüfung ausgewählter Systemabläufe unterstützt zum einen die Analyse einer Fehlerwirkung, ermöglicht aber auch den Nachweis, dass ein System trotz umgesetzter Änderungen weiterhin fähig ist diese Systemabläufe korrekt umzusetzen. Die systematische Wiederholung von Testfällen wird nach ISO/IEC/IEEE 24765 [21] als Regressionstest bezeichnet:

Definition 3.17 (Regressionstest)**.** *Der* Regressionstest *ist die wiederholte Überprüfung eines Systems oder einer Komponente, um zu verifizieren, dass Änderungen keine unbeabsichtigten Auswirkungen verursacht haben und das System oder die Komponente weiterhin den spezifizierten Anforderungen entspricht.*

Eine Grundlage für die Definition der Testtiefe liefert die Einstufung nach Automotive Safety Integrity Level (ASIL). Diese ist in der Norm ISO 26262 [59] wie folgt definiert:

Definition 3.18 (Automotive Safety Integrity Level)**.** *Der* Automotive Safety Integrity Level *ist eine von vier Einstufungen, die für ein Item oder ein Element die notwendigen Anforderungen und Sicherheitsmaßnahmen nach ISO 26262 zur Vermeidung eines unangemessenen Restrisikos spezifizieren. Dabei ist D die strengste und A die am wenigsten strenge Einstufung.*

Die automotive SPICE Prozessreferenz fordert für alle Teilprozesse auf dem rechten Schenkel des V-Modells die Erstellung eines Testplans zur Qualitätssicherung als Teilergebnis [45]. Der System- oder Softwareteil, der durch einen spezifizierten Testfall untersucht wird, wird als Testobjekt bezeichnet [19]. Alle Einrichtungen, Werkzeuge und Programme zur Durchführung und zur Dokumentation von Tests für die Verifizierung und Validierung eines Systems werden im Begriff der Testumgebung zusammengefasst [58]. Der Begriff Testobjekt wird durch [19] wie folgt definiert:

Definition 3.19 (Testobjekt)**.** *Das* Testobjekt*, auch System under Test, ist die zu testende Einheit.*

3.3 Entwicklungsmethoden und -ansätze

Im Bereich der Software- und Regelungsentwicklung für Kraftfahrzeuge existieren verschiedene methodische Entwicklungsansätze. Zielsetzungen für den Einsatz von Entwicklungsmethoden sind, analog zu den Entwicklungsprozessen, die Minimierung des Projektrisikos, eine verbesserte Komplexitätsbeherrschung und Qualitätssicherung.

Die Anforderungen an die anzuwendenden Systems Engineering Methoden unterscheiden sich zwischen verschiedenen Funktionen und verändern sich mit den unterschiedlichen Entwicklungsstufen. Wie in Abschnitt 2.7 eingeführt, liegt der Fokus eines Automotive SPICE konformen Prozesses in den frühen Phasen auf der Definition von Anforderungen und der Entwurfsspezifikation. Dies erfordert Methoden, die eine systematische Analyse und Prüfung der erstellten Arbeitsprodukte ermöglichen. An der Spitze des Vs verschiebt sich dieser Fokus auf die korrekte Umsetzung der spezifizierten Elemente und den Nachweis spezifikationskonformer Implementierung der Software-Units in den zugehörigen Software-Items. Mit der sukzessiven Integration der einzelnen Software-Items auf der Hardware zu Komponenten und der Verbindung der Komponenten zu Subsystemen und letztlich zum Gesamtsystem auf dem rechten Schenkel des Vs verschiebt sich der Schwerpunkt der Verifikation von der detaillierten Beschreibung auf Unit-Ebene hin zu einem gesamtheitlichen Nachweis auf Systemebene.

Die Herausforderung besteht in der Identifikation einer auf die jeweilige Problemstellung zugeschnittenen Kombination von Methoden. Universelle Methoden basieren häufig auf komplexen Werkzeugumsetzungen [34], die eine flexible Anpassung

der Methode auf die spezifischen Rahmenbedingungen des Entwicklungsprojekts erschweren. Lücken im methodischen Unterbau führen zu Aufwänden, die nicht unmittelbar zur Zielerreichung beitragen. Eine Überladung des methodischen Unterbaus erschwert hingegen eine schnelle Einarbeitung und die flexible Umsetzung der Methoden.

3.3.1 Modellbasierte Entwicklung

Modelle dienen in der Wissenschaft und dem Ingenieurwesen dem vereinfachten Verständnis und Umgang mit komplexen Zusammenhängen. Stachowiak [60] definiert ein Modell wie folgt:

Definition 3.20 (Modell)**.** Modelle *sind [...] Abbildungen, Repräsentationen natürlicher oder künstlicher Originale (Abbildungsmerkmal).* Modelle *erfassen nicht alle Attribute, [...] nur solche, die [...] relevant scheinen (Verkürzungsmerkmal).* Modelle *erfüllen ihre Ersatzfunktion für bestimmte Subjekte innerhalb bestimmter Zeitintervalle und unter Einschränkung auf bestimmte gedankliche oder tatsächliche Operationen (Pragmatisches Merkmal).*

Der modellbasierter Systementwurf (Model-Based Systems Engineering, MBSE) adressiert die Herausforderung zunehmender Komplexität, verteilter Entwicklung und involvierter Rollen durch den Einsatz standardisierter Modellierungen. Im Kontext des MBSE dient die Modellierung nach Törngren [61] unter anderen der Analyse, der Beschreibung und dem Entwurf von Systemen. Weiter unterscheiden sich Modellierungen nach Törngren im Grad und der Art der Formalisierung, dem Anwendungsbereich und dem Abstraktionsgrad. Unterschieden wird zwischen formalen[2] Modellen, die eine Aussage über das erwartete Systemverhalten ermöglichen, konzeptuellen Modellen, die insbesondere die Kommunikation und Dokumentation unterstützen, und konstruktiven Modellen, die sich auf das operative Verhalten fokussieren und eine direkte Grundlage für oder einen Teil des Systementwurfs darstellen [61].

Eine detaillierte Sicht auf den MBSE mit formalen Modellen liefert Long [63]. Ein Modell basiert in diesem Kontext auf einer formalen Sprache, beinhaltet eine Struk-

[2] Im Kontext des Software Engineering bezeichnen formale Methoden die Anwendung strikter, mathematisch-logischer Beschreibungen und Analysen für die Spezifikation, den Entwurf und den Test von Software [62]

tur, dient der Beweisführung und dem Nachweis der Zielerreichung und ermöglicht Mechanismen, die eine klare und verständliche Darstellung dieses Nachweises unterstützen. Nach Brambilla [64] ermöglicht eine Formalisierung mittels „impliziter aber eindeutig definierter Semantik [...] den präzisen Austausch von Informationen“. Die durch die Modellierung erreichte Formalisierung und Abstraktion verbessert das Verständnis der entwickelten Artefakte und vereinfacht eine maschinelle Interpretation. Dies ermöglicht den Einsatz von Werkzeugen zur Unterstützung und Automatisierung einzelner Arbeitsschritte. Statische Analyse und Verifikation (Abschnitt 3.4), Simulation (Abschnitt 3.5.2 und 3.3.4) und Modellausführung tragen zu einer erfolgreichen Entwicklung bei und Modell-zu-Modell-Transformationen ermöglichen den Übergang zwischen unterschiedlichen Abstraktionsgraden und Einsatzzwecken [64, 65].

Modellbasierte Ansätze treten in der Softwaretechnik in unterschiedlichen Varianten auf. Abbildung 3.4 stellt die nach Brambilla [64] wichtigsten Vertreter dar. Die MBSE umfasst demnach alle Ansätze der Softwaretechnik, die auf dem Einsatz von Modellen im Entwicklungsprozess aufbauen. Die Modelle dienen als Unterstützung im Entwicklungsprozess. Beim modellgetriebener Systementwurf (Model-Driven Engineering, MDE) wird die Modellbildung weiter in den Fokus gerückt. Die Modelle sind die wichtigsten Artefakte und zentralen Treiber aller Teilprozesse der Entwicklung. Die Ergebnisse aller Entwicklungsphasen und -tätigkeiten (Anforderungen, Analyseergebnisse, Implementierungen, Testfälle,...) liegen als formale Modelle vor. Wird lediglich die Untermenge der unmittelbar die Entwicklung betreffenden Teilprozesse durch Modelle getrieben spricht man von einer modellgetriebenen Entwicklung (engl. modellgetriebene Entwicklung (Model-Driven Development, MDD)). Es existieren verschiedene Interpretationen und Umsetzungen der MDD. Brambilla [64] führt hier als Beispiel die modellgetriebene Architektur (Model-Driven Architecture, MDA) der Object Management Group (OMG) auf.

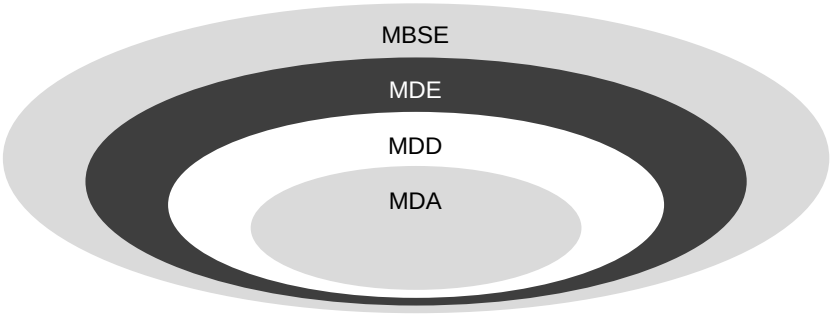


Abbildung 3.4: Einordnung verschieden abgestufter, modellbasierter Ansätze nach [64]

Die MDA der OMG setzt mit der Unified Modeling Language (UML) [66] auf eine standardisierte Modellierungssprache zur Spezifikation von Softwareprodukten, „in ihrer Absicht ähnlich den lange in technischen Disziplinen verwendeten Blaupausen“ [67]. Die Systems Modeling Language (SysML) [68] erweitert und spezialisiert diesen Ansatz für systemrelevante Aspekte und unterstützt damit das Systems Engineering.

Die UML definiert sechs Modellebenen, die eine starke Ähnlichkeit mit den in Abschnitt 2.7 vorgestellten Arbeitsprodukten von Automotive SPICE vorweisen. Im Use-Case-Modell werden die funktionalen Anforderungen beschrieben, d.h. die erwartete Verwendung der Software und die Interaktionen zwischen den beteiligten Rollen und Systemen. Inhaltlich entspricht dieses Modell dem Ergebnis des Automotive SPICE Prozessschritts SYS.1.

Analog zu SWE.1 werden die Anforderungen im Analysemodell verfeinert und das Systemverhalten wird initial auf einzelne Systemteile aufgeteilt. Beispielsweise dienen Aktivitätsdiagramme der detaillierten Analyse anwendungsbezogener Anforderungen. Das Entwurfsmodell stellt die Blaupause der Implementierung dar und entspricht dem Prozessschritt Softwareentwurf (SWE.2). Basierend auf dem Analysemodell wird die technische Systemarchitektur aus Subsystemen, Klassen und Schnittstellen definiert und die spezifizierten Use-Cases werden auf diese Strukturen abgebildet. Beispielsweise kann das dynamische Verhalten eines Elements im Entwurfsmodell mit Hilfe eines Sequenzdiagramms beschreiben werden. Im Implementierungsmodell wird die Art der Umsetzung der Elemente des Entwurfsmodells in Form von Quellcode-Komponenten den spezifizierten Klassen zugeordnet. Die Erstellung dieses Modells entspricht der Phase des detaillierten Softwareentwurfs und der Implementierung (SWE.3).

Die in den Phasen SWE.4, SWE.5 und SWE.6 spezifizierten Unit-, Integrations- und Systemtests werden bei Anwendung der UML im Testmodell beschrieben. Dies beinhaltet die angewandten Testfälle und Testmethoden sowie die verwendete Testumgebung. Aufgrund der Beschränkung der UML auf Softwareentwicklungen, können die Prozessschritte zu Integration, Test und Validierung auf Systemebene keinen direkten Modellelementen zugewiesen werden. Ähnlich verhält es sich mit dem Bereitstellungsmodell, das die physische Struktur des Computersystems und die Abbildung der implementierten Komponenten auf diese Struktur beschreibt. Das Bereitstellungsmodell ist vergleichbar mit der im folgenden Abschnitt beschriebenen technischen Systemarchitektur der E/E-Anteile eines Fahrzeugs.

3.3.2 Modellbasierte Ansätze in der E/E-Entwicklung eines Fahrzeugs

In der Automobilentwicklung nehmen modellbasierte Methoden insbesondere im Bereich der E/E-Systemarchitektur und im Software- und Regelungsentwurf eine zunehmend wichtige Rolle ein. Ein wichtiger Treiber dieser Entwicklung ist die Zunahme an Systemfunktionen, deren Verteilung über mehrere Steuergeräte und die damit steigende Vernetzung zuvor unabhängig entwickelter Systemkomponenten [69]. In Verbindung mit einer zunehmenden Anzahl an Modellvarianten und -konfigurationen einzelner Baureihen führt dies zu einer gesteigerten Vielfalt und Konnektivität und damit zu steigender Komplexität des Gesamtsystems (vergl. Abschnitt 2.1). Zur Bewältigung dieser Herausforderung werden im Bereich der E/E-Architekturentwicklung verbreitet modellbasierte Ansätze verfolgt. Grundlage ist dabei eine Abstraktion der Systemarchitektur auf verschiedenen, sachlogisch zusammenhängenden Ebenen.

Eine etablierte Partitionierung ist die Unterscheidung einer logischen und einer technischen Systemarchitektur [14, 3, 69]. Durch den verstärkten Softwareeinsatz im Fahrzeug trägt eine weitere Abstraktionsebene und Perspektive in Form der Software Systemarchitektur zu einem umfassenden Systemverständnis und einer durchgängigen Modellierung bei. Ein Beispiel für eine umfangreiche und durchgängige Modellstruktur bietet [70]. Abbildung 3.5 fasst die genannten Ansätze [14, 15, 3, 69, 70] in einer reduzierten Form zusammen und ergänzt für diese Arbeit relevante Betrachtungen. Die Architekturebenen werden in Anlehnung an [14, 15, 3, 69, 70] wie folgt definiert:

Definition 3.21 (Logische Systemarchitektur)**.** *Die* logische Systemarchitektur *bildet das funktionale Verhalten des Systems mit Hilfe logischer Funktionsblöcke ab.*

Definition 3.22 (Software Systemarchitektur)**.** *Die* Software Systemarchitektur *beschreibt die Softwarekomponenten und -einheiten des Systems sowie ihre Schnittstellen und Interaktionen.*

Definition 3.23 (Technische Systemarchitektur)**.** *Die* technische Systemarchitektur *spezifiziert den Aufbau der gesamten Fahrzeugelektronik aus Hardwaresicht.*

Die abstrakte Verhaltensbeschreibung wird auf System- und Subsystemebene während des Systemarchitekturentwurfs erstellt (vergl. Abschnitt 2.7) und folgt einem

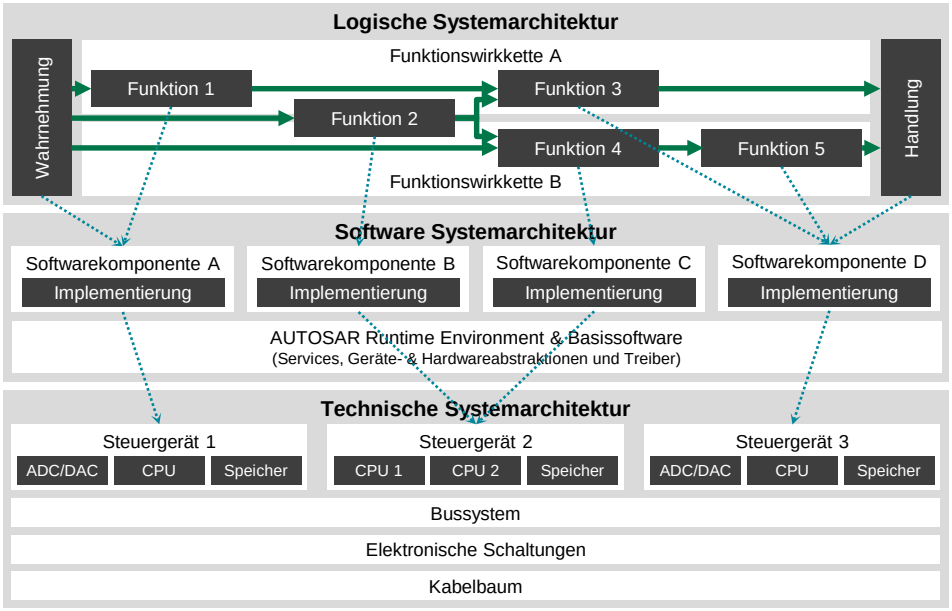


Abbildung 3.5: Zusammenfassende Darstellung der für diese Dissertation relevanten Abstraktionsebenen der E/E Systemarchitektur und Zuordnung des verteilten, funktionalen Verhaltens auf Softwarekomponenten und Steuergeräte.

kausalen Zusammenhang. Basierend auf einer Wahrnehmung werden einzelne Funktionen stimuliert. Diese Stimuli werden durch die logischen Funktionsblöcke propagiert bis eine Handlung des Systems resultiert. Vollständige Fahrzeugfunktionen werden durch Funktionswirkketten repräsentiert. Diese fassen alle an der Wirkkette beteiligten Funktionsblöcke zusammen und ermöglichen die Abbildung einzelner Funktionspfade in komplex vernetzten Systemarchitekturen. Auf Ebene der logischen Systemarchitektur konnte sich der Einsatz formaler Methoden in der Automobilentwicklung bis heute nicht flächendeckend etablieren. Dies liegt unter anderem an der Verfügbarkeit performanter Prototyping Systeme (vergl. Abschnitt 3.3.3), die eine frühzeitige Überprüfung von Systementwürfen ermöglichen. Die logische Systemarchitektur liegt daher zumeist in Form eines konzeptuellen Modells vor (3.3.1).

Die Software Systemarchitektur wird während des Softwarearchitekturentwurfs spezifiziert (vergl. Abschnitt 2.7). Die Funktionen der logischen Systemarchitektur werden einzelnen Softwarekomponenten zugeordnet, in deren Umfang sie implementiert werden. Neben dieser in Abbildung 3.5 durch Pfeile skizzierten exemplarischen Zuordnung beinhaltet die Software Systemarchitektur die Spezifikation von Softwa-

remodulen, -schnittstellen, verwendeten Tasks und zugeordneten Speicherbereichen. Mit AUTOSAR (AUTomotive Open System ARchitecture) [71] existiert in der Automobilbranche auf der Ebene der Software Systemarchitektur eine Standardisierung der Basissoftware. Sie abstrahiert die Hardwarekomponenten eines Steuergeräts und stellt die für den Betrieb einer funktionalen Komponente notwendigen Elemente, wie Betriebssystem, Kommunikationsmodul und Bootloader [72], zur Verfügung. Eine funktionale Komponente wird im AUTOSAR-Kontext als Softwarekomponente (Software Component, SWC) bezeichnet. Sie wird auf dem Application-Layer über die AUTOSAR Runtime Environment in die Basissoftware integriert.

In der technischen Systemarchitektur werden die integrierten Steuergeräte über ihre Hardwarearchitektur als Einzelkomponenten spezifiziert. Diese Spezifikation beinhaltet integrierte Hardwaremodule wie Recheneinheiten (z.B. Prozessor (Central Processing Unit, CPU), Grafikprozessor (Graphics Processing Unit, GPU) und Field Programmable Gate Array (FPGA)), Analog-Digital-Wandler (Analog-to-Digital Converter, ADC), Digital-Analog-Wandler (Digital-to-Analog Converter, DAC) und Speichermodule (Random-Access Memory (RAM) und Read-Only Memory (ROM)). Die Vernetzung der einzelnen Hardwarekomponenten wird im Bussystem oder der Netzwerktopologie definiert. Weitere Elemente, die auf Ebene der technischen Systemarchitektur modelliert werden, sind elektronische Schaltpläne und der benötigte Kabelbaum inklusive aller Steck- und Klemmkontakte. Die Entwicklung der technischen Systemarchitektur wird in dieser Arbeit nicht betrachtet, die zugehörigen Prozesse lassen sich im Rahmen des in der Automotive SPICE (Abschnitt 2.7) enthaltenen „Plug-in“-Konzepts eingliedern.

Neben dem Entwurf der beschriebenen Ebenen der E/E-Systemarchitektur ist der Einsatz modellbasierter Techniken in der Funktionsentwicklung, insbesondere beim Entwurf von Steuerungs- und Regelungsalgorithmen, weit verbreitet [72]. Als wichtigsten Vertreter kommerzieller Werkzeugketten für die modellgetriebene Funktionsentwicklung im Bereich der Entwicklung eingebetteter Systeme benennt dabei eine Studie der Chalmers University of Technology [65] die integrierte Umgebung von MATLAB, Simulink und Stateflow. Neben Eclipse, Enterprise Architect und IBM Rational Software Modeller stellen im Bereich der Automobilentwicklung ASCET, Dymola (Modelica) und SimulationX wichtige weitere Vertreter dar. Die in der Studie meist genannten Einsatzzwecke der Modelle sind die Simulation und die Codegenerierung, gefolgt von der Verwendung von Modellen zur Dokumentation und Informationsweitergabe, Testfallgenerierung und zur strukturellen Konsistenzprüfung.

MATLAB, Simulink und Stateflow ermöglichen gleichermaßen die Entwicklung von zustandsbasierten Automaten und Steuerungs- und Regelungssystemen [73], sowie die „Modellierung gemischt diskret-/kontinuierlicher Systeme“ [69]. Rossi u. a. [74] beschreiben die Anwendung und Vorteile des durchgängig modellbasierten Entwurfs am Beispiel eines innovativen Traktionskontrollsystems. Die Kopplung von Steuerungsentwurf und Simulation ermöglicht eine frühzeitige Analyse des Systemverhaltens und der eingesetzten Algorithmen. Die Codegenerierung mittels Embedded Coder [75] oder TargetLink [76] ermöglicht eine plattformunabhängige Entwicklung und die Erzeugung eines „hocheffizienten und zuverlässigen C-Code[s] in Produktionsqualität“ [72]. Die Durchgängigkeit der Werkzeugkette wird neben ergänzenden Werkzeugen zur Simulation, vergleiche Abschnitt 3.5.2, durch Rapid Prototyping Techniken ergänzt [77].

Das beschriebene Vorgehen fokussiert sich auf dem jeweiligen E/E-Feature, dem durch den Kunden wählbaren Ausstattungsmerkmal. Steht in der Entwicklung die kundenerlebbare Funktionswirkkette im Vordergrung und ist eine Übertragung dieser Features auf verschiedene Baureihen vorgesehen, spricht man von Funktionsorientierung [9]. Eine Alternative zur Funktionsorientierung stellt die Serviceorientierung dar. Bei der Serviceorientierung steht nicht die gesamte Wirkkette im Fokus sondern die einzelnen Dienste. Dahinter steht das Konzept, dass diese Dienste in mehreren Features genutzt werden und daher nicht auf ein spezifisches Feature sondern für die übergeordnete Gesamtheit entwickelt werden. Traub [78] beschreibt den Einsatz der Serviceorientierung im Kontext zukünftiger E/E Architekturen.

3.3.3 Rapid Prototyping und Musterstände

Der Produktentwicklungsprozess in der Automobilindustrie erstreckt sich wie in Abschnitt 2.9 beschrieben wurde über einen Zeitraum von 48-60 Monaten. Eine regelmäßige Validierung des aktuellen Entwicklungsstands in der realen Zielumgebung trägt innerhalb dieses Zeitraums entscheidend zu einer erfolgreichen Projektdurchführung bei. Wie in Abschnitt 2.4 beschrieben ist, unterstützt eine Bewertung und Analyse prototypisch umgesetzter Alternativlösungen in frühen Projektphasen die Minimierung von Projektrisiken in der Softwareentwicklung.

In der Fahrzeugentwicklung haben sich verschiedene als Rapid Prototyping (RP) oder auch Rapid Control Prototyping bezeichnete Techniken zur frühzeitigen Va-

lidierung etabliert. Diese Techniken kommen insbesondere in der in Abschnitt 2.9.1 beschriebenen Forschung und Vorentwicklung, sowie in den in Abschnitt 2.7 beschriebenen Entwurfsphasen der Serienentwicklung zum Einsatz. In der Serienentwicklung spricht man in diesem Kontext von Musterständen (vergl. Abbildung 3.6). Dieses Unterkapitel beschreibt die Nutzung von RP-Technologien in der Forschung und Vorentwicklung und vermittelt den Einsatz verschiedener Musterstände im Verlauf der Serienentwicklung.

Im Kontext der Softwareentwicklung bezeichnet RP die Ausrüstung von mechatronischen Mustern, Vorserienfahrzeugen oder Serienfahrzeugen mit Systemen für die prototypische Ausführung von Software. Abhängig von der Integrationsebene der entwickelten Funktion wird zwischen vertikalen Fullpass-Prototypen und horizontalen Bypass-Prototypen unterschieden [72]. Eine Definition der Zielrichtung vertikaler und horizontaler Prototypen liefert Schäuffele [14]:

Definition 3.24 (Horizpntaler Prototyp)**.** Horizontale Prototypen *zielen auf die Darstellung eines breiten Bereichs eines Software-Systems, stellen aber eine abstrakte Sicht dar und vernachlässigen Details.*

Definition 3.25 (Vertikaler Prototyp)**.** Vertikale Prototypen *stellen dagegen einen eingeschränkten Bereich eines Software-Systems recht detailliert dar.*

Die Begriffe vertikaler und horizontaler Prototyp verdeutlichen sich am Abbild der Systemarchitektur (Abbildung 3.5). Horizontale Bypass-Prototypen sind auf die Darstellung der Funktionalität einer oder mehrerer SWCs fokussiert, die in der AUTOSAR Runtime Environment eingebettet werden. Vertikale Fullpass-Prototypen hingegen beinhalten auch Teile der Runtime und der Basissoftware, beispielsweise Services und Treiber für digitale und analoge Hardwareschnittstellen.

Vertikale Fullpass-Prototypen kommen zum Einsatz, wenn die Neuentwicklung einen wesentlichen Anteil an Hardwarekomponenten beinhaltet oder kein geeignet erweiterbares Steuergerät für eine Bypass-Anwendung verfügbar ist [79]. Der Fullpass-Prototyp beinhaltet alle für die Anbindung notwendiger Sensoren und Aktuatoren relevanten elektronischen Schnittstellen, wie beispielsweise Analog-Digital-Wandler, Leistungstreiber und Verstärkerschaltungen [72]. Die Nähe zur Hardware und damit auch zu zugrundeliegenden physikalischen, mechanischen und elektrischen Prinzipien erfordert robuste Prototyping-Hardware und beinhaltet

harte Echtzeitanforderungen. Diese Anforderungen werden durch dedizierte Hardwareplattformen mit konfigurierbaren I/O-Schnittstellen, wie beispielsweise die dSPACE Microautobox [80], erfüllt.

Horizontale Bypass-Prototypen finden bei der Entwicklung von Fahrzeugfunktionen Verwendung, die von der Hardware abstrahiert dargestellt werden können. Sensoren und Aktuatoren werden über vorhandene oder ergänzte Software-Schnittstellen von bereits im Fahrzeug verbauten Steuergeräten abgebildet. Die Kommunikation zwischen Prototyp und Steuergerät auf Hardwareebene erfolgt über einen Zugang zum Bussystem des Fahrzeugs oder eine Diagnoseschnittstelle des Steuergeräts. Durch die umgesetzte Abstraktion der Hardware sind horizontale Prototypen von physikalischen, mechanischen und elektrischen Prinzipien abgekoppelt. Daher sind die Anforderungen an die Robustheit und Echtzeitfähigkeit der Hardware horizontaler Bypass-Prototypen gegenüber vertikalen Fullpass-Prototypen geringer.

Als Prototypen-Hardware eingesetzte PC-Systeme mit Schnittstellenerweiterungen für die gängigen Fahrzeugbusse [81] bieten heute eine große Flexibilität und hohe Rechenleistungen. Derartige Prototypensysteme erfüllen in der Regel lediglich weiche Echtzeitanforderungen. Der Einsatz von PC-Systemen erfordert zusätzlich die Verwendung einer Middleware, die die zur Ausführung der entwickelten Software notwendigen Funktionen des Steuergerätebetriebssystems, wie Task-Scheduler, Timer und Interrupts, emuliert. Zwei wichtige Vertreter aktuell verfügbarer Middleware Systeme sind das Automotive Data and Time-Triggered Framework (ADTF) [82] und das Robot Operating System (ROS) [83]. Middleware und PC-basierte Prototypensysteme bieten gegenüber dedizierten Systemen eine Reihe von Vorteilen.

- Die Verwendung einer identischen Software- und Hardwarearchitektur am Schreibtisch und im Fahrzeug vereinfacht Debugging-, Test- und Erprobungsaktivitäten erheblich.

- Der Einsatz von Consumer-Produkten führt zu einer Kostenreduktion und ermöglicht eine einfache Rekonfiguration und Anpassung an geänderte Anforderungen und Rahmenbedingungen.

- Die leistungsstarke Hardware unterstützt in frühen Entwicklungsphasen die Fokusierung auf Algorithmen, ohne durch technische Ursachen, wie Laufzeit oder Speicher, eingeschränkt zu sein.

- Die Hardwareleistung von PC-Systemen ermöglicht mit der Aufzeichnung aller externen und internen Schnittstellen die vollständige Messung einer Testfahrt.

- Der Einsatz von für PC-Systeme verfügbaren Programmbibliotheken für Visualisierung, Kommunikation und Analyse stellt eine adaptive und hochwertige Werkzeuggrundlage dar. Notwendige, aber nicht im Fokus der Entwicklung stehende Funktionen, wie Mensch-Maschine-Schnittstelle (Human Machine Interface, HMI) Mock-ups, können mit etablierten Bibliotheken einfach und robust abgebildet werden.

- PC-basierte Simulationslösungen können auf eine einfache Weise mit dem Prototypensystem gekoppelt werden.

PC-basierte horizontale Prototypen haben sich insbesondere in der Forschung und Entwicklung automatisierter und autonomer Fahrfunktionen etabliert [84, 85, 86, 87]. So wurde im Forschungsprojekt „Stadtpilot“ eine Linux Umgebung mit ADTF als Middleware eingesetzt [88]. Lenz u. a. beschreiben die prototypische Umsetzung einer pilotierten Parkhausfunktion mit Hilfe von ADTF und deren simulationsbasierte Bewertung [89]. Eine Zusammenfassung der Ergebnisse und Erfahrungen aus der Erprobung automatisierten Fahrens mit Prototypen auf öffentlichen Straßen bieten Pink u. a. [90]. Aeberhard u. a. [91] beschreiben den Einsatz von ROS als Middleware für die Entwicklung automatisierter Fahrfunktionen im Fahrzeug und in Kopplung mit einer Simulationssoftware.

Neben der Realisierung und Darstellung innovativer Funktionen im Bereich der Forschung und Vorentwicklung ist die zu Beginn des Abschnitts beschriebene regelmäßige Validierung des aktuellen Entwicklungsstands mit Hilfe von Musterständen etabliert. Eine übliche Klassifizierung der Entwicklungsstufen ist dabei eine Unterscheidung in A-, B-, C- und D-Muster [18]. Abbildung 3.6 skizziert den Einsatzzeitpunkt von RP und Musterständen über den Verlauf der Produktentwicklung.

Definition 3.26 (Musterstand)**.** Musterstände *dienen der Erprobung der entwickelten Systeme und Komponenten. Für die verschiedenen Projektphasen existieren mit A-, B-, C- und D-Muster unterschiedliche Abstraktionsgrade der Seriennähe.*

Der A-Musterstand ermöglicht in der frühen Serienentwicklung eine erste Bewertung

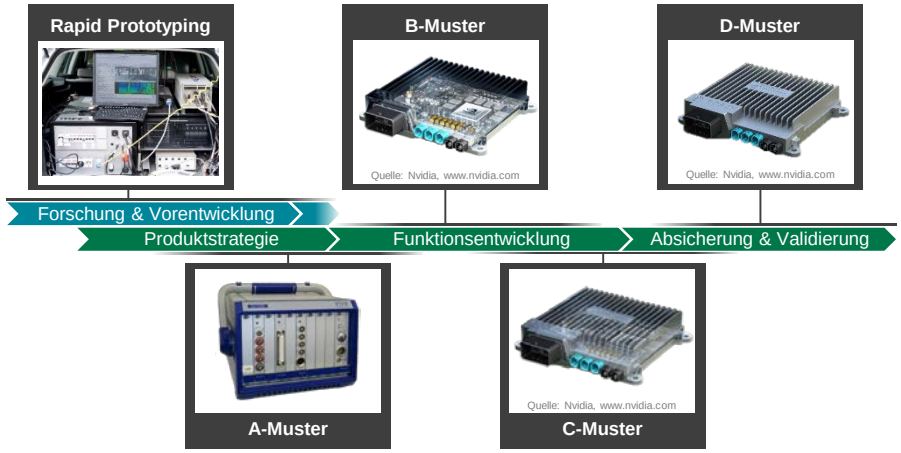


Abbildung 3.6: Im Laufe der Entwicklung eingesetztes Rapid Prototyping und Musterstände zur Funktionsbewertung.

des Funktionskonzepts auf System- und Subsystemebene. Er unterstützt die Entscheidungsfindung bezüglich alternativen Entwurfskonzepten und die Detaillierung der Anforderungsspezifikation. Die Software realisiert ausgewählte Funktionsbestandteile und entspricht dem aktuellen Stand der spezifizierten Kommunikationsschnittstellen und -nachrichten. Zur Ausführung der Software werden RP-Systeme eingesetzt. Die verwendeten Werkzeuge zur Erstellung der Software weichen in der Regel von den Serienwerkzeugen der Zielhardware ab. Auf diesem Entwicklungsstand werden häufig Teile des Subsystems oder verbundener Subsysteme nachgeahmt (vergleichbar mit dem in Abschnitt 3.5.3 beschriebenen Einsatz von Attrappen). Die Erprobungen der einzelnen Kundenfunktionen werden entsprechend dem spezifischen Bedarf und größtenteils unabhängig von anderen Funktionen der Baureihe durchgeführt.

Beim Übergang auf das B-Muster wird die RP-Umgebung durch die Zielhardware ersetzt. Die Software wird dazu mit den entsprechenden Zielwerkzeugen integriert und erstellt. Ziel dieser Musterstufe ist die Realisierung des vollen Funktionsumfangs, jedoch können einzelne Details noch von der Spezifikation abweichen. Der System- und Softwareentwurf ist abgeschlossen. Dies beinhaltet die Festlegung der Schnittstellen, der Steuergerätediagnose und aller eingesetzten Algorithmen. Mit dem B-Musterstand werden die ersten Dauererprobungen im Rahmen der Baureihe durchgeführt.

Der C-Musterstand beinhaltet die Umsetzung des vollen Umfangs der Spezifikation.

Wie für das B-Muster erfolgt die Softwareerstellung mit der für die Serienhardware spezifizierten Werkzeugkette. Für den C-Musterstand werden die Integrations- und Qualifikationstests aller Software-Einheiten und -Komponenten abgeschlossen und das Gesamtsystem ist bereit für die abschließende Systemqualifikation. Der C-Musterstand hat eine besondere Bedeutung für die Fahrzeugerprobung und die Systemqualifikation (vergleiche Abschnitt 3.6), da auf dieser Stufe neben den Sommer- und Wintererprobungen auch spezifische Länderfreigaben der Baureihe durchgeführt werden.

Der D-Musterstand, auch Nullserienstand, realisiert das spezifizierte System vollumfänglich und dient zum abschließenden Nachweis der Serienreife. Auf dem Stand des D-Musters ist die korrekte Umsetzung aller Anforderungen nachgewiesen und die Phase Systemqualifikationstest (Abschnitt 2.7) wird abgeschlossen. Die Produktfreigabe wird auf Basis der Integrations- und Qualifikationstests sowie der mit Hilfe des D-Musters durchgeführten Absicherung durchgeführt.

3.3.4 Simulationsgestützte Entwicklung

Die Erprobung entwickelter Funktionen für eingebettete Systeme durch RP ermöglicht eine kontinuierliche Validierung über den gesamten Entwicklungsprozess hinweg. Die für eine detaillierte Analyse benötigten Systemstimuli werden aus der realen Systemumgebung abgeleitet. Die damit verbundenen Echtzeitanforderungen wirken sich jedoch nachteilig auf Debugging und Fehleranalyse aus. Um bei der Erprobung aufgetretene Fehlerwirkungen (Abschnitt 3.2) zu analysieren müssen interne Zustände und Größen betrachtet werden. Die Ausführung in Echtzeit erschwert die Identifikation des Fehlerzustands, da dabei nicht alle Einzelschritte eines fehlerbehafteten Funktionszyklus betrachtet werden können.

Einen weiteren Nachteil der Erprobung in der realen Systemumgebung stellt die Reproduzierbarkeit der Umgebungsbedingungen dar. Beim Einsatz von RP-Systemen führt die Sicherstellung einer geeigneten Reproduzierbarkeit zu hohen Aufwänden. Beispielsweise müssen zur Nachstellung einer dedizierten Verkehrssituation auf der Autobahn, alle an dieser Situation beteiligten Fahrzeuge in der Erprobung durch instruierte Testfahrer nachgebildet werden. Für die Durchführung von Regressionstests, wie in Abschnitt 3.2 beschrieben, sind Erprobungen daher nicht geeignet. Für die Verifikation der Systemfunktionen wird daher die Systemumgebung durch Modelle nachgebildet und simuliert [33]. Die verwendeten Modelle reichen von

einfachen Kennfeldern zu detaillierten physikalischen Mehrkörpermodellen [92]. Der Begriff der Simulation ist durch [21] wie folgt definiert:

Definition 3.27 (Simulation)**.** *Die* Simulation *beschreibt die Nutzung eines Modells, das wie ein bestimmtes System arbeitet arbeitet oder sich verhält, wenn es mit einem Satz kontrollierter Eingaben beaufschlagt wird.*

Die Eignung und Zuverlässigkeit einer Simulation, für eine spezifische Fragestellung eine belastbare Aussage zu liefern, wird nach Muessig [93] durch drei grundlegende Kriterien bestimmt:

- Fähigkeit - die Modellierung muss die Realität (z.B. die Regelstrecke) in einer Detaillierung abbilden, die die angestrebte Verwendung ermöglicht.

- Genauigkeit - die verwendete Simulationssoftware muss innerhalb ihrer Spezifikation fehlerfrei sein, die Modellparameter müssen geeignet gewählt sein und die Simulationsergebnisse müssen eine ausreichende Korrelation mit der realen Welt aufweisen.

- Verwendbarkeit - die Simulation sollte Maßnahmen und Vorkehrungen enthalten, die einen fehlerhaften Einsatz außerhalb des definierten Nutzungskontextes verhindern.

Im Gegensatz zu einem Sensor- oder Steuersystem mit offenem Wirkungsweg (Open-Loop) erfordert der Test eines Regelungssystems einen geschlossenen Wirkungsweg (Closed-Loop). Die Regelgröße am Ausgang der Regelung beeinflusst das Verhalten der Strecke und wirkt sich damit mittelbar auf die Systemstimuli am Eingang des Reglers aus. Für den simulativen Test einer Regelungsfunktion ist daher die Modellierung der gesamten Regelstrecke notwendig. Das Blockschaltbild in Abbildung 3.7 stellt den Test der Regelung einer assistierenden oder automatisierenden Fahrzeugfunktion mit Hilfe eines Fahrzeug-, Fahrer- und Umgebungsmodells aus logischer Systemsicht dar.

Die Regelung ist als Testobjekt in der Simulationsumgebung eingebettet. Diese setzt sich aus verschiedenen Modellen der Regelstrecke, Aktuatoren und Sensoren zusammen. Für das Beispiel einer Tempomatfunktion umfasst das Fahrzeugmodell den Antriebsstrang und die Längsdynamik des Fahrzeugs. Die Funktion wirkt über den Motor als Aktuator auf den Antriebsstrang des Fahrzeugs und ein Sensormodell

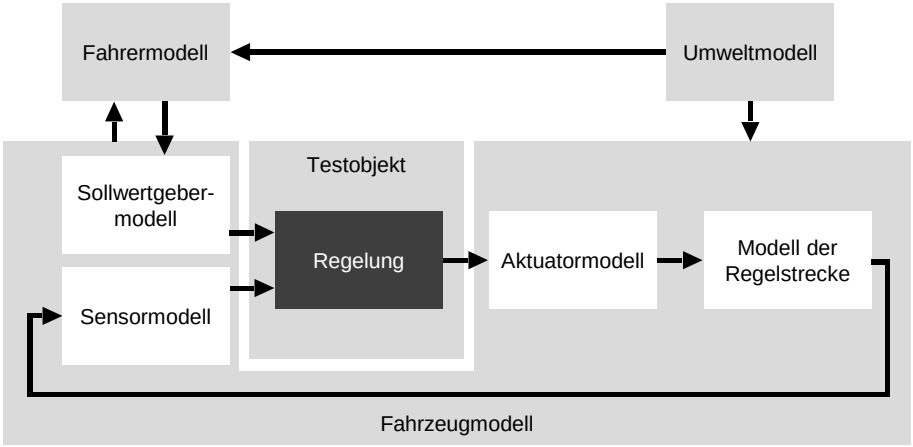


Abbildung 3.7: Test der Regelung einer assistierenden oder automatisierenden Fahrzeugfunktion (vergl. Abbildung 3.1) mit Hilfe eines Fahrzeug-, Fahrer- und Umgebungsmodells aus logischer Systemsicht.

gibt die Geschwindigkeit und die aktuelle Motordrehzahl zurück. Der Regelsollwert wird durch ein Fahrermodell über das Sollwertgebermodell eines Lenkstockschalters vorgegeben. Über ein Umweltmodell werden Störgrößen, wie Wind und Steigung, auf das Streckenmodell des Antriebsstrangs aufgeschlagen.

In der Automobilbranche ist die Verwendung dedizierter Gesamtfahrzeugsimulationen weit verbreitet [94]. Es existieren verschiedene Open-Source Gesamtfahrzeugsimulationen, wie die Projekte TORCS [95], DECOS [96] und Gazebo [97], sowie kommerzielle Plattformen, wie CarMaker [98], Virtual Test Drive [99] und PreScan [100]. Diese Simulationen sind dabei aus einzelnen Modellkomponenten zusammengesetzt, die das Gesamtsystem Fahrzeug nachbilden. Abbildung 3.8 bietet einen Überblick über die Komponenten einer Gesamtfahrzeugsimulation für den Test einer Fahrerassistenzfunktion.

Das dargestellte Fahrzeugmodell setzt sich aus den Komponenten Fahrerarbeitsplatz, Fahrzeugdynamik und Umfeldsensorik zusammen. Mit Pedalen, Lenkrad und Bedienelementen beinhaltet der Fahrerarbeitsplatz alle Schnittstellen zur manuellen Fahrzeugführung. Das Fahrzeugdynamikmodell enthält die für die Fahrzeugbewegung relevanten Baugruppen Antriebsstrang, Chassis und Karosserie. Die Umfeldsensorik setzt sich aus Modellen für die Umfelderfassung mittels Radar, Lidar, Kamera und Ultraschall sowie der Eigenlokalisierung mittels Satellitennavigation zusammen.

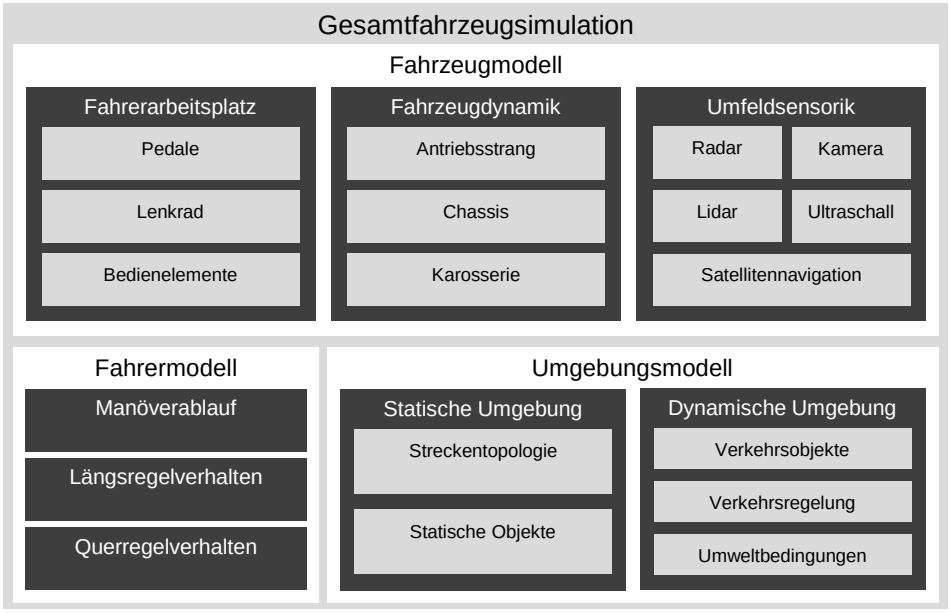


Abbildung 3.8: Komponenten einer Gesamtfahrzeugsimulation

Die Fahrzeugumgebung kann in einen statischen und einen dynamischen Teil unterschieden werden. Der statische Teil umfasst dabei die Topologie des simulierten Streckennetzes und alle enthaltenen, feststehenden Objekte, wie Gebäude, Bäume, Verkehrszeichen, Leitplanken und Ampeln. Für die Modellierung des statischen Teils existiert mit OpenDRIVE [101] eine offene, XML-basierte Spezifikation für die Beschreibung von Straßennetzen und ihrer Umgebung. Der dynamische Teil setzt sich aus simulierten Verkehrsobjekten, wie Fahrzeugen, Radfahrern und Fußgängern, der Verkehrsregelung durch Ampelphasen, elektronische Verkehrsschilder und Wechselverkehrszeichen, sowie den Umweltbedingungen, wie Regen, Sonnenstand und Temperatur, zusammen.

Die konsistente Beschreibung aller Teilaspekte einer Simulation basiert üblicherweise auf Szenarien. Bagschik [102] definiert die in Abbildung 3.9 dargestellten drei Abstraktionsebenen funktionale Szenarien, logische Szenarien und konkrete Szenarien. Diese Ebenen bieten für die verschiedenen Phasen und Aktivitäten der Produktentwicklung jeweils eine geeignete Abstraktion.

Zur Spezifikation von Simulationsszenarien existieren verschiedene modellbasierte Ansätze [103, 104, 105, JB1] und mit OpenSCENARIO eine Bestrebung zu einer

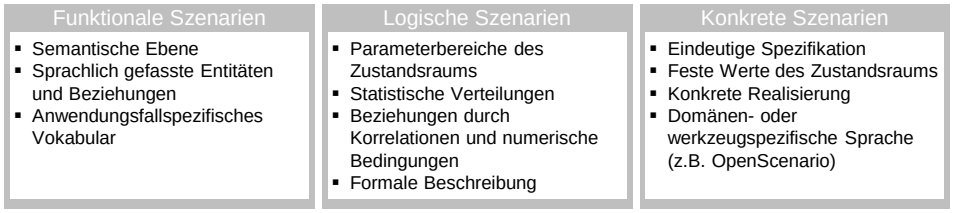


Abbildung 3.9: Abstraktionsebenen für die Beschreibung von Simulationsszenarien nach [102].

einheitlichen und universellen Spezifikationssprache [106]. Wichtige Begriffe zur Definition von Simulationsszenarien für den Test von Fahrzeugregelungsfunktionen sind dabei nach [103, 104, 105, JB1, 106] zusammengefasst wie folgt definiert:

Definition 3.28 (Szenario)**.** *Ein* Szenario *beschreibt die vorbestimmte zeitliche Reihenfolge oder Entwicklung mehrerer Szenen und der zugehörigen Situationen der Teilnehmer innerhalb der Szenerie.*

Definition 3.29 (Szenerie)**.** *Die* Szenerie *beinhaltet alle räumlich statischen Objekte eines Szenarios, wie geometrische Elemente (z.B. Straßennetz mit Verbindungen und Knoten) und globale Attribute (z.B. das Wetter und die Tageszeit). Der Zustand der räumlich statischen Objekte kann sich mit der Zeit verändern (z.B. Ampeln und Wetter).*

Definition 3.30 (Szene)**.** *Die* Szene *ist eine Momentaufnahme der Umgebung inklusive der Szenerie, der dynamischen Elemente, der Zustände aller Teilnehmer und der Beziehungen zwischen allen statischen und dynamischen Objekten. Nur die Repräsentation einer Szene in einer simulierten Welt kann allumfassend sein, da eine gemessene Repräsentation der realen Welt immer unvollständig und fehlerbehaftet ist.*

Definition 3.31 (Situation)**.** *Eine* Situation *repräsentiert den Blick eines Teilnehmers auf die jeweilige Szene. Sie fasst alle relevanten Teile der Szene zusammen, die um die geplanten Vorhaben der Teilnehmer und funktionsspezifische Aspekte angereichert werden.*

Definition 3.32 (Manöver)**.** *Ein* Manöver *ist die abstrakte Beschreibung des Verhaltens eines Teilnehmers während eines Akts.*

Das Fahrermodell setzt sich aus je einer Komponente zur Modellierung des Längsregelverhaltens und des Querregelverhaltens zusammen. Dedizierte Aktionen des simulierten Fahrers können über den Manöverablauf abgebildet werden [98]. Neben Änderungen der Fahrzeugführung, wie der Durchführung eines Spurwechsels oder eines Abbiegevorgangs, umfasst dies auch die Ausführung von Bedienhandlungen, wie das Setzen des Blinkers oder die Aktivierung einer in die Simulation integrierten Fahrerassistenzfunktion.

Simulationen ermöglichen die reproduzierbare Prüfung der vollständigen Wirkkette einer regelnden Funktion. Mit zunehmender Fähigkeit und Genauigkeit der verwendeten Simulationsmodelle steigen neben dem Rechenaufwand auch die benötigten Umfänge zur Spezifikation und Evaluierung der Simulationsszenarien und zur Validierung der Simulationsergebnisse. Die Entwicklung und exakte Parametrierung der verwendeten Modelle ist daher mit einem hohen finanziellen und zeitlichen Aufwand verbunden und führt zu „multi level“ Simulationen. Zur Reduktion der Aufwände stellt die automatisierte Generierung von Testfällen und -szenarien basierend auf statistischen Modellen eine häufig diskutierte Möglichkeit dar [107, 108], diese Herausforderung zu lösen.

3.3.5 Continuous Integration und Delivery

Als **Continuous Delivery** wird eine Entwicklungsmethode aus dem Umfeld der agilen Software Entwicklung bezeichnet, die einen schnelleren Markteintritt und kürzere Wartungszyklen sicherstellt [109]. Die Methode kann weitestgehend unabhängig vom implementierten Vorgehensmodell (vergl. Kapitel 2)in unterschiedlichen, an das Produkt oder die Organisation angepassten, Ausprägungen Anwendung finden. Gemeinsam haben alle Ausprägungen, dass Produktiterationen in kurzen zeitlichen Abständen einem möglichst großen Kreis von Entwicklern und Anwendern zur Verfügung gestellt werden.

Continuous Integration beschreibt dabei die vollautomatisierte Integration von neuem und überarbeitetem Quellcode in der gemeinsamen Codebasis. Alle Entwickler arbeiten dazu auf einem gemeinsamen Repository. Zur Übergabe neuer Ergebnisse in die gemeinsame Codebasis wird mit Hilfe einer automatisierten Werkzeugkette ausführbare Software erzeugt. Neben der Kompilierung können Werkzeuge zur statischen Codeanalyse in der Werkzeugkette integriert werden [110]. Schlägt ein Schritt der Erzeugung fehl, wird die Übergabe der Ergebnisse

abgelehnt. Dadurch wird fehlerhafter Quellcode in der gemeinsamen Codebasis reduziert. Um eine weitere Qualitätssteigerung der gemeinsamen Codebasis zu erreichen kann die Continuous Integration Werkzeugkette mit automatisierten Tests erweitert werden. **Continuous Verification** beschreibt dabei die prinzipielle Ausführung automatisierter Testfälle vor jeder Integration von Änderungen in die gemeinsame Codebasis. Abhängig von der verwendeten Testautomatisierung beinhaltet dies die automatisierte Ausführung von Unittests und gegebenenfalls auch die automatisierte Ausführung von Integrations- und Systemtests. Bei Letzteren stellt für umfangreiche Systeme eine geeignete automatisierte Auswertung von Testergebnissen eine besondere Herausforderung dar. Die letzte Ausbaustufe einer Continuous Delivery Umsetzung ist die Integration von **Continuous Deployment**. Dabei werden die zur Freigabe notwendigen Prozesse in der Werkzeugkette integriert und eine regelmäßige Bereitstellung neuer Softwareinkremente für interne und externe Kunden sichergestellt [111]. Durch den Einsatz einer Deployment Pipeline kann dabei der Lieferprozess durch Werkzeugeinsatz automatisiert werden.

Eine gängige Praxis ist die Integration manueller Codereview- und Testprozesse in die Prozess- und Werkzeugkette [112]. Manuelle Codereviews ermöglichen eine zusätzliche Überprüfung von Programmierrichtlinien vor der Integration der Änderungen in die Codebasis und fördern durch die direkte Kooperation den Wissens- und Erfahrungsaustausch zwischen Programmierern. Automatisiert nur bedingt prüfbare Anforderungen wie Nutzeroberflächendesigns und Aktivitätsabläufe innerhalb der erstellten Software können über manuelle Testprozesse vor der Auslieferung einer aktualisierten Version abgesichert werden.

3.4 Statische Testmethoden

Definition 3.33 (Statische Testmethoden)**.** Statische Testmethoden *überprüfen das Testobjekt gegenüber definierten Anforderungen und Qualitätskriterien ohne den Programmcode auszuführen [58].*

Es handelt sich dabei um Inspektionen (s. Abschnitt 3.4.1), die manuell oder automatisiert mit Hilfe geeigneter Werkzeuge durchgeführt werden. Statische Methoden können über alle Entwicklungsphasen hinweg Anwendung finden. Da die Arbeitsprodukte der frühen Entwicklungsphase größtenteils Spezifikationen umfassen, wie beispielsweise Anforderungsdokumente und Systemarchitekturbeschreibungen, ermöglichen statische Methoden bereits in diesen frühen Phasen eine effiziente

Qualitätsbewertung. Die Auswahl möglicher statischer Verifikationsmethoden ist in der in Abschnitt 3.2 beschriebenen Teststrategie definiert und wird im Testplan für das jeweilige Testobjekt konkretisiert.

3.4.1 Manuelle Prüfungen

Der **Walkthrough** beschreibt eine „manuelle und informale Prüfmethode“ [55]. Ziel ist es in einem Softwareprodukt Fehlerzustände und Verbesserungsmöglichkeiten zu finden, mögliche alternative Umsetzungen zu bewerten und die Einhaltung von Standards und Spezifikationen zu überprüfen [113]. Dazu werden an der Softwareentwicklung beteiligte Teammitglieder sowie weitere interessierte Personen von einem Entwickler durch sein Softwareprodukt geführt. Die Beteiligten stellen Fragen zur Software und der Umsetzung von Funktionen und können Anmerkungen und Kommentare zu möglichen Fehlerquellen einbringen. Neben der damit erzielten Qualitätsverbesserung und -sicherung werden bei einem Walkthrough neue Produkte oder Funktionen den beteiligten Personen vorgestellt und erklärt. Der Mehrwert dieser Methode umfasst auch über das Softwareprodukt hinausreichende Fragestellungen, wie den Austausch über angewendete Techniken, eingesetzte Funktionsbibliotheken und variierende Programmierstile. Nach dem Standard IEEE 1028 für Software Reviews [113] sind Rollen mit Management- oder Personalverantwortung von der Beteiligung an einem Walkthrough ausgeschlossen. Dies fördert eine Fokussierung auf fachlich-inhaltliche Fragestellungen in einer offenen Atmosphäre.

Reviews werden nach IEEE 1028 [113] in Managementreviews und technische Reviews unterschieden. Ziel eines Managementreviews ist die Prüfung des Projektfortschritts, der Projektausrichtung und der Verfügbarkeit von Ressourcen. Ein technisches Review beschreibt eine formalisierte Variante des Walkthrough. Es überprüft die korrekte Umsetzung der Spezifikation, die Einhaltung von Standards und Richtlinien und die Umsetzung von Änderungen. Das technische Review folgt formalen Vorgaben und bezieht lediglich offizielle Artefakte in die Prüfung ein. Nach einer individuellen Prüfung der Artefakte durch die jeweiligen Prüfer wird in einer moderierten Reviewsitzung ein einheitlicher Bericht verabschiedet. Managementrollen sind vom technischen Review ausgeschlossen.

Die abgeschwächte Variante des informellen Reviews [55] verzichtet auf eine Reviewsitzung und einen formalen Abschlussbericht. Einzelne Prüfer führen das Review

eigenständig durch und melden Auffälligkeiten zurück oder korrigieren diese direkt. Informelle Reviews werden in agilen Softwareentwicklungsprojekten häufig durch die Fertigstellungskriterien für Teilaufgaben gefordert. In Werkzeugketten für Continuous-Integration und Continuous-Depolyment werden Reviews vor der Übergabe eines aktualisierten Softwarestands in den Hauptzweig empfohlen [114]. Durch Reviews kann Erfahrung der Entwickler geteilt werden und ein zugrundeliegendes Paradigma besagt, dass lesbarer Code qualitativ hochwertigerer Code ist [115].

Die **Inspektion** beschreibt eine manuelle Prüfung mit strikt formalisiertem Ablauf[55]. Die Ziele der Inspektion sind nach dem Standard IEEE 1028 [113] die Verifikation der erfüllten Spezifikation, der Übereinstimmung mit Standards und die Identifikation von Abweichungen von Standards und Spezifikationen. Stilistische Fragestellungen oder alternative Implementierungsmöglichkeiten werden nicht betrachtet. Die Inspektion wird durch zwei bis sechs Teilnehmer, inklusive des Autors des inspizierten Produkts, durchgeführt und wird durch einen ausgebildeten Moderator geleitet. Die Artefakte werden gemeinsam überprüft und alle identifizierten Abweichungen werden aufgezeichnet. Die erstellte Liste der Abweichungen wird im Anschluss auf Vollständigkeit und Genauigkeit geprüft und gegebenenfalls ergänzt. Die Ein- und Ausgangsprodukte sowie die Start und- Endbedingungen einer Inspektion sind im Standard eindeutig spezifiziert.

3.4.2 Statische Codeanalyseverfahren

Die Statische Code Analyse beschreibt eine Testmethode zur automatisierten Prüfung des Quelltextes durch die Analyse der deklarierten Variablen und definierten Funktionen. Die Identifikation von formalen Fehlern wie fehlerhafter Syntax und der Verletzungen von Programmierstandards ist ein Hauptziel statischer Verifikation. Darüber hinaus können mit Hilfe der statischen Codeanalyse Fehlerzustände im Daten- und Kontrollfluss identifiziert werden [55]. Auch sekundäre Anforderungen wie Vorgaben an den Programmierstil und die Dokumentation können mit Hilfe statischer Verfahren ausgewertet werden. Der Einsatz und die Prüfung derartiger Richtlinien sind häufig projekt-, team- oder unternehmensspezifisch festgelegt und dienen einer gesteigerten Codequalität. Sie verbessern das Verständnis der implementierten Funktionen, deren Zuverlässigkeit und Wartbarkeit [116], [117].

Grundlegende statische Prüfungen sind üblicherweise bereits in den Compiler

integriert. Neben der Überprüfung der Syntax erfolgt zur Kompilierung eine Konformitätsprüfung der deklarierten Variablentypen und deren Verwendung. Mögliche identifizierbare Fehlerzustände sind unter anderen die Verwendung undefinierter Variablen sowie die (mehrfache) Definition von Variablen ohne zwischenzeitliche Verwendung [55]. Entsprechend der gewählten Analysetiefe werden Kontrollflusselemente und -strukturen in die automatisierte Prüfung aufgenommen. Typische Fehlerzustände, die durch eine automatisierte Kontrollflussanalyse identifiziert werden können, sind unvollständige oder nicht erreichbare If-Else-Bedingungen und mehrfach definierte oder nicht unterbrochene Switch-Case-Strukturen [118].

Die **Prüfung von Guidelines und Standards** bietet eine zusätzliche Absicherung zu diesen in den Compiler integrierten Analysen. Mit Hilfe von dedizierten Werkzeugen, wie PC-Lint [119], kann die Einhaltung von Programmierrichtlinien für einheitlichen Code [116] [120] und die Umsetzung spezifischer Standards wie beispielsweise MISRA-C [121] überprüft werden. Auch können Softwaremetriken [110], wie die HIS Source Code Metrics [122], erfasst werden. Diese Standards beschränken die Möglichkeiten der verwendeten Programmier- oder Modellierungssprache durch ein Verbot kritischer und fehleranfälliger Deklarationen, Definitionen und Konstruktionen [121].

Statische Codeanalysen werden üblicherweise in Form vollständiger Regressionstests implementiert und in die verwendete Build-Toolchain der Entwickler integriert [110]. Fortschrittliche Versionsverwaltungssysteme halten die Möglichkeit vor, Codeinkremente abzuweisen und aus dem Produktivcode fernzuhalten, falls diese statische Analysen verletzen [123], [124].

Die **Prüfung von Laufzeitfehlern** von Softwarekomponenten durch eine statische Analyse wird durch Methoden zur abstrakten Interpretation von Programmiersprachen ermöglicht. Mittels abstrakter Interpretation decken statische Analysemethoden über die Kontrollflussanalysen hinaus tief im Quelltext verborgene Laufzeitfehler auf, beispielsweise Speicherüberläufe, Division durch Null oder Zugriffe auf Speicherstellen außerhalb eines Arrays [125]. Die Absicherung gegen derartige Fehlerzustände ist mit dynamischen Testverfahren mit einem hohen Aufwand zur Implementierung von Testfällen verbunden. Zu aufgedeckten Fehlerwirkungen müssen mittels Debugging dann zunächst die ursächlichen Fehlerzustände identifiziert werden. Statische Verfahren decken hingegen direkt im Quelltext enthaltene Fehlerzustände auf. Damit bietet die Integration statischer Analyseverfahren in den Softwareentwicklungs- und Freigabeprozess einen hohen Mehrwert in Bezug auf die Softwarequalität und die Bewertung des Absicherungsumfangs.

3.5 Dynamische Testmethoden

Definition 3.34 (Dynamische Testmethoden)**.** Dynamische Testmethoden *zur Verifikation beinhalten alle Testfälle, die eine Ausführung des Programmcodes erfordern [58].*

Dynamische Tests finden auf allen Ebenen der Systementwicklung Anwendung. Nach Abschluss der Systemdekomposition werden, beginnend auf der Einheitenebene, alle Implementierungs- und Integrationsergebnisse durch dynamische Tests verifiziert.

3.5.1 Testumgebung und -stimuli

In den während der Entwicklung nach Automotive SPICE (Abschnitt 2.7) durchlaufenen Phasen zur Dekomposition und Integration existieren die verschiedenen Elemente und Gegenstände auf unterschiedlichen Realisierungsebenen (vrgl. Abschnitt 2.2 und Abbildung 2.11). Für den dynamischen Test eingebetteter Systeme wird für diese unterschiedlichen Realisierungsebenen jeweils eine angepasste Konfiguration der Testumgebung benötigt [126]. In den frühen Phasen des System- und Softwareentwurfs (SYS.3 & SWE.2) existiert eine Entwurfsspezifikation, gegebenenfalls in Form eines Funktionsmodells. Dieser Entwurf wird im Verlauf der Implementierung (SWE.3) in Software umgesetzt und zunächst in eine SWC (SWE.5), dann in der Steuergerätehardware (SYS.4) integriert.

Neben der Testumgebung werden für eine Ausführung der Elemente und Gegenstände geeignete Stimuli benötigt. Diese Stimuli können hinsichtlich ihres Open-Loop und Closed-Loop Verhaltens unterschieden werden. Für den Test mit geschlossenem Wirkungsweg werden, wie in Abschnitt 3.3.4 beschrieben, Simulationsmodelle eingesetzt, die die Stimuli zur Laufzeit des Tests erzeugen. Abschnitt 3.5.2 beschreibt die speziellen Eigenschaften des Einsatzes von Simulationsmodellen für den Test eingebetteter Systeme.

Definition 3.35 (Testvektoren)**.** *Der Begriff* Testvektoren *fasst alle Teststimuli mit offenem Wirkungsweg zusammen [126].*

Der Test mit Testvektoren kann auf allen Systemebenen an den jeweiligen Schnittstellen des Testobjekts durchgeführt werden. Die Verwendung von Testvektoren

für den Test eines Simulinkmodells wird durch Shorky [126] beschrieben: „...ein Vektor nach dem anderen wird angewandt und das jeweilige Ergebnis wird im Systemvalidierungsplan aufgezeichnet“. Die beim Test verwendeten Testvektoren können künstlich erzeugt werden oder durch Messungen am realen System gewonnen werden. Junior [127] beschreibt einen Ansatz zur Generierung von Testvektoren basierend auf der Analyse der Anweisungs-, Zweig- und Pfadüberdeckung. Die generierten Testvektoren unterstützen den funktionalen Test durch eine vollständige strukturelle Abdeckung.

Testvektoren eignen sich speziell für den Test von Systemen mit Open-Loop Verhalten. Regelnde Systeme können aufgrund der fehlenden Reaktion der Teststimuli auf die Regelgröße am Systemausgang nur sehr begrenzt mit statischen Testvektoren geprüft werden. Back-To-Back Verfahren [128], die das Verhalten von zwei Testobjekten beim Übergang zwischen unterschiedlichen Integrationsstufen und bei der Refaktorierung[3] miteinander vergleichen, sind von diesem Nachteil nicht betroffen.

Definition 3.36 (Capture & Replay Test)**.** Capture & Replay Test *bezeichnet die Verwendung von aufgezeichneten Daten für den Test von Softwaresystemen [55].*

In der Automobilbranche sind Replay-Tests insbesondere im Bereich der Umfelderfassung weit verbreitet. Die Testfälle werden aus Aufzeichnungen der realen Welt abgeleitet. Noack [129] beschreibt die Aufzeichnung von Rohsignalen eines Radarsensors für die Wiederverwendung der Messdaten in der Systementwicklung. Im wissenschaftlichen Umfeld der Bildverarbeitung werden annotierte Bildsequenzen für den objektiven Vergleich der Leistungsfähigkeit verschiedener Forschungsansätze verwendet [130, 131]. In den in Abschnitt 3.3.3 vorgestellten Rapid Prototyping Middleware Systemen ADTF und ROS sind Capture & Replay Techniken bereits fest integriert.

Die Auswahl geeigneter und in der realen Welt aufgezeichneter Vektoren kann dem erfahrungsbasierten Testentwurf zugerechnet werden (vgl. Abschnitt 3.7.1). Die Verwendung von Generatoren und Aufzeichnungen bietet eine ressourcenschonende Alternative zur manuellen Implementierung der Testfälle, insbesondere für aufwändige Testszenarien auf Systemebene. Auf diese Weise werden manuelle Aufwände zur Definition der in den Testvektoren enthaltenen Signalverläufe reduziert.

[3] Refaktorierung beschreibt die Überarbeitung des Quelltextes zur strukturellen Verbesserung ohne Änderung des funktionalen Verhaltens

3.5.2 X-in-the-Loop

Der Test von Regelungssystemen erfordert wie in Abschnitt 3.3.4 beschrieben einen geschlossenen Wirkungsweg. Für die simulationsgestützte Entwicklung von Closed-Loop Systemen existieren für alle Ausbaustufen entlang des Entwicklungspfads spezialisierte Methoden. Um eine wiederholte Verwendung der aufwändig erstellten Modelle über alle Phasen des Entwicklungsprozesses zu ermöglichen, werden die Werkzeuge für eine durchgängige Verwendung vorbereitet [34, 132]. Dieser Ansatz wird auch als X-in-the-Loop (XiL) Methode bezeichnet [126, 133].

Definition 3.37 (X-in-the-Loop)**.** X-in-the-Loop *bezeichnet die Verwendung eines einheitlichen Simulationsmodells für den Test eines Regelungssystems über alle Phasen des Entwicklungsprozesses hinweg [126, 133].*

Der Begriff „in-the-Loop“ impliziert dabei einen bidirektionalen Informationsfluss zwischen Testobjekt und Simulationsumgebung. Abbildung 3.10 skizziert vier gängige Abstraktionsebenen des X-in-the-Loop Tests im Umfeld der E/E-Entwicklung für Fahrzeuge [33, 72, 19, 34, 14, 134].

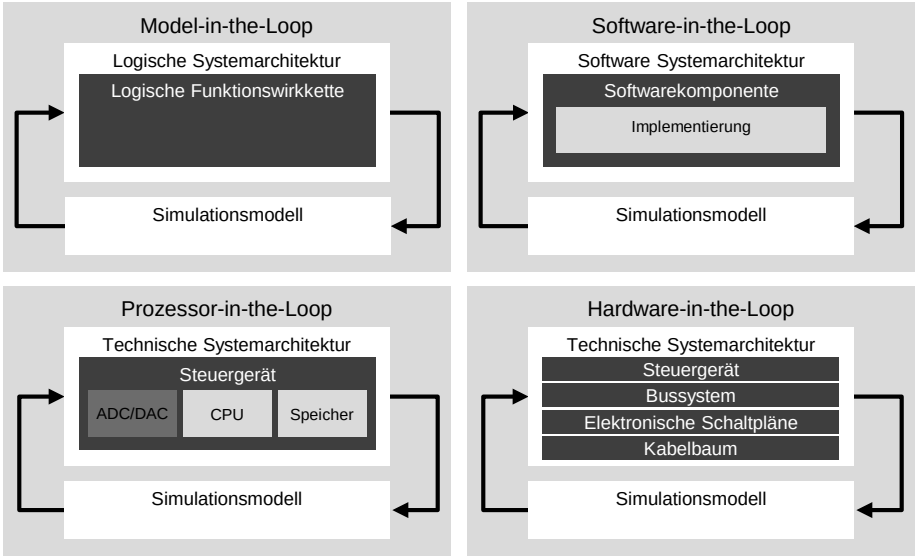


Abbildung 3.10: Wiederverwendung eines Simulationsmodells für den simulationsbasierten Test auf verschiedenen Abstraktionsebenen der Systemarchitektur

Model-in-the-Loop (MIL) definiert den Einsatz einer Simulationsumgebung zur Generierung der Stimuli für die Entwicklung und den Test spezifizierter Funktionsmodelle während der Automotive SPICE Phasen des Systementwurfs (SYS.3) und des Softwareentwurfs (SWE.2). Auf dieser Ebene werden die Funktion und die Systemumgebung unabhängig von der Softwarearchitektur und der technischen Systemarchitektur des Zielsystems simuliert ausgeführt. Häufig wird die MIL-Ebene mit dem Einsatz einer modellbasierten Entwurfssprache gleichgesetzt [72, 19, 14]. Der Einsatz Quelltext-basierter Programmiersprachen (z. B. C/C++ [82, 91, 135]) in der Konzeptentwicklung assistierender und automatisierender Fahrfunktionen wird dabei nicht berücksichtigt. Ziel der Analysen und Tests auf MIL-Ebene ist die Verifizierung des Entwurfs der logischen Systemarchitektur und der eingesetzten Algorithmen [126]. Im Rahmen dieser Arbeit werden auch C/C++ basierte Konzeptentwicklungen zur MIL-Ebene gezählt, wenn lediglich funktionale Bestandteile der logischen Systemarchitektur im Fokus der Entwicklung stehen und die Software- und Hardwarearchitektur des Zielsystems nicht berücksichtigt wird. Als Beispiel kann hier der funktionale Test einer nicht in die Basissoftware integrierten Softwarekomponente genannt werden, bei dem die relevanten Funktionen der Basissoftware (z. B. Scheduler, Interrupts und Schnittstellen) durch Attrappen emuliert werden.

Software-in-the-Loop (SIL) definiert den Einsatz einer Simulationsumgebung für die Ausführung und den Test von Seriencode auf Ebene der Software Systemarchitektur. Die implementierte Softwarekomponente wird dabei in die Basissoftware integriert auf einer virtualisierten Laufzeitumgebung betrieben [19]. Nach Broekman [33] ist die SIL-Ebene durch Ressourceneinschränkungen (z. B. Beschränkung auf 8 oder 16-bit Integer Verarbeitung) gekennzeichnet. SIL beschreibt damit die Ausführung der in die Software Systemarchitektur integrierten Funktion auf Desktop-Systemen unter Verwendung des Host-Compilers. Ziel der Tests auf SIL-Ebene ist der Nachweis der korrekten Funktionserbringung der Softwarekomponente im Kontext des Gesamtsystems und der Systemumgebung.

Prozessor-in-the-Loop (PIL) definiert den Einsatz einer Simulationsumgebung für die Ausführung und den Test der implementierten Softwarekomponente auf der Prozessorarchitektur des Zielsystems. Die Softwarekomponente wird dazu unter Verwendung des Target-Compilers „auf einem Evaluierungsboard, das typischerweise den im Steuergerät verwendeten Prozessor enthält," [72] integriert und ausgeführt. Die Basissoftware und Schnittstellen des Steuergeräts werden dabei emuliert und beispielsweise über eine Ethernet Verbindung mit der auf dem Desktop-System betriebenen Simulationsumgebung verbunden. Die Ausführung auf dem Evaluierungsboard und Kopplung mit einem Desktop-System erfordert eine echtzeitfähige

Simulationsumgebung und für den Zugriff auf interne Zustände und Speicherstellen eine spezielle Software- oder Hardware-basierte Instrumentierung [136]. Ziel der Tests auf PIL-Ebene ist die Verifikation der spezifizierten Laufzeiten, des Speicherbedarfs sowie die Prüfung der korrekten Funktionserbringung unter Verwendung des Target-Compilers.

Hardware-in-the-Loop (HIL) definiert den Einsatz einer Simulationsumgebung für die Ausführung und den Test der in die Zielhardware integrierten Softwarekomponente unter Verwendung der technischen Systemschnittstellen [19]. Auf HIL-Ebene wird zwischen Komponenten- und Integrationstests unterschieden [14]. Beim Komponententest wird die korrekte Funktion eines individuellen Steuergeräts geprüft. HIL-Tests am Prüfstand verifizieren die korrekte Interaktion mit der elektrischen und mechanischen Hardware der Systemkomponente (bspw. Motorsteuergerät am Motorenprüfstand [137]). Ein Wirkkettenprüfplatz dient der Verifikation eines zusammenhängenden und verteilten Subsystems (bspw. Systemverbund Fahrerassistenz [138]) und ein Gesamtfahrzeugprüfplatz verifiziert das Zusammenspiel aller im Fahrzeug enthaltenen E/E Systeme [139].

Ein großer Vorteil der durchgängigen Verwendung einheitlicher Simulationsmodelle ist der Nachweis der identischen Funktionserbringung auf unterschiedlichen Integrationsstufen. Dies ermöglicht die Umsetzung von Back-To-Back Tests [128] zwischen den verschiedenen Ebenen und erleichtert die Identifikation enthaltener Fehlerzustände. Diesem Vorteil steht gegenüber, dass sich die Anforderungen an die Simulationsumgebung zwischen den Integrationsstufen unterscheiden. Während in frühen Phasen für die Analyse des logischen Funktionsmodells eine grundlegende Flexibilität erwünscht ist, steht auf Hardwareebene die Einhaltung der Echtzeit im Fokus.

3.5.3 Unit-Tests

Tests auf Einheitenebene werden üblicherweise als Unit-Tests bezeichnet. Sie basieren auf der während der Entwurfsphasen (SYS.3, SWE.2 und SWE.3 in Abbildung 2.11) vorgenommenen Dekomposition und stellen Bottom-Up getriebene Testverfahren dar. Sie werden explizit für einzelne Funktionen oder vollständige Klassen und Schnittstellen zur Anwendungsprogrammierung (Application Programming Interface, APIs) implementiert [128]. In der Automotive SPICE Prozessreferenz (Abschnitt 2.7) sind Unit-Tests dem Prozess „SWE.4 - Verifikation Softwareeinheiten“

zuzuordnen.

Das Testobjekt der Unit-Tests ist, abhängig vom gewählten Entwicklungsvorgehen, das erstellte Funktionsmodell oder der geschriebene Quelltext. Analog zur MIL-Simulation findet die Testausführung losgelöst von der Softwarearchitektur statt. Im Gegensatz zu MIL, wird anstelle der funktionalen Zusammenhänge der logischen Systemarchitektur die korrekte Implementierung einzelner Funktionselemente verifiziert. Die Ausführung der Tests erfolgt auf dem Desktop-PC der Entwickler, bzw. der Tester. Werkzeuge zur Durchführung von Unit und Modultests sind häufig direkt in der Build-Toolchain, bzw. der integrierten Entwicklungsumgebung (Integrated Development Environment, IDE) integriert [140].

Für den Test einer Softwareeinheit, die von einem zweiten Element abhängt, muss dieses zweite Element vorgetäuscht werden. Die Vorteile einer Attrappe (Fake) für den Unit-Test sind die gesicherte Verfügbarkeit und Reproduzierbarkeit. Eine Softwareeinheit, die von Dritten entwickelt wird, ist auf Einheitenebene nicht gesichert verfügbar und liefert unter Umständen variierende Rückgaben. Zum Beispiel stellt MSTest in Microsoft Visual Studio drei Typen von Attrappen zur Verfügung: Stubs, Mocks und Shims [140]. Durch Stubs vorgetäuschte Methoden liefern vorgegebene Antworten auf Aufrufe durch das Testobjekt. Mocks sind aufwändigere Objekte und beinhalten einen internen Zustand. Sie simulieren ein definiertes Verhalten. Shims werden genutzt um Plattform-Methoden und API-Aufrufe abzufangen. Brader [140] nennt als Beispiel die Plattform-Methode *DateTime.Now*, die bei jedem Testaufruf einen andern Rückgabewert liefern würde.

3.5.4 Integrationstests

Softwareintegrationstests verifizieren die Interaktion der Softwareeinheiten einer Systemkomponente untereinander. Der Prozessschritt SWE.5 Softwareintegration und Integrationstests (Kapitel 2.7) baut auf der Verifizierung der einzelnen Softwareeinheiten auf und führt die Bottom Up Integrationsstrategie fort. Ziel der Software Integrationstests ist der Nachweis der Konformität mit dem spezifizierten Softwareentwurf.

Eine schrittweise Integration der einzelnen Einheiten ermöglicht eine hierarchische Teststrategie. Die während der Integration wachsende Software wird in jedem Integrationsschritt „an ihren hierarchisch höchstgelegenen Schnittstellen“ getestet

[128]. Abgeschlossen wird diese Phase mit der Integration der implementierten und integrierten Softwarekomponente in die Basissoftware des Steuergeräts. Für den Integrationstest der vollständigen Software existieren virtuelle Steuergeräte [141], die analog zu den Attrappen für Unit-Tests die Hardware des Steuergeräts vortäuschen.

Die in Abschnitt 3.5.2 vorgestellte SIL-Methode verifiziert die funktionalen Zusammenhänge der integrierten Softwaremodule. Viele der im Rahmen von SIL und Softwareintegrationstest verwendeten Werkzeuge sind identisch und eine explizite Trennung der Methoden ist schwierig. Im Automotive SPICE Referenzmodell werden die Softwareintegration und -qualifizierung getrennt. Der Softwareintegrationstest verifiziert die Umsetzung des Architekturentwurfs und die Schnittstellen zwischen einzelnen Modulen. Die Softwarequalifizierung verifiziert korrekt umgesetzte Softwareanforderungen. SIL stellt dazu die geeignete Methode dar.

Hardwareintegrationstests stellen eine weitere Stufe der Integrationstests dar und können dem Automotive SPICE Prozessschritt SYS.4 zugeordnet werden. Ziel des Hardwareintegrationstests ist der Nachweis der korrekten Interaktion zwischen Software und Hardware und damit verbunden der Nachweis der korrekten Umsetzung des spezifizierten Entwurfs und der Schnittstellen. Für die Prüfung der Software-Hardware-Interaktion sind dedizierte Laboraufbauten notwendig, die das Steuergerät aus dem Gesamtsystem lösen und die physikalischen Schnittstellen nachbilden [19]. Diese Nachbildung ist vergleichbar mit den Stub genannten Attrappen auf Einheitenebene. Die genutzten Laboraufbauten für die Hardwareintegrationstests und die oben beschriebenen HIL-Tests sind in der Regel identisch. Inhaltlich können Hardwareintegrationstests und HIL-Tests häufig nicht trennscharf abgegrenzt werden. In dieser Dissertation wird der HIL-Begriff im Sinne einer geschlossenen Regelschleife verwendet und damit gegenüber Hardwareintegrationstests mit Open-Loop Verhalten abgegrenzt.

Steht bereits vor Abschluss der Softwareintegration ein entsprechendes Steuergerät inklusive der benötigten Basissoftware zur Verfügung, können Hardwareintegrationstests für ausgewählte risikobehaftete Komponenten parallel zu den durchgeführten Softwareintegrationstests durchgeführt werden [128]. Dadurch können auf die Hardwareintegration zurückzuführende Fehler, wie unzureichende Ressourcen, Ausführungszeiten oder fehlerhafte Treiber, früher erkannt und behoben werden. Dieses Vorgehen besitzt eine große Ähnlichkeit mit den oben beschriebenen PIL-Tests, die ebenfalls eine frühzeitige Aussage über die Laufzeit und den Speicherverbrauch auf der technischen Zielarchitektur ermöglichen.

Systemintegrationstests verifizieren das Zusammenspiel einzelner Subsysteme und finden, wie Hardwareintegrationstests, im Prozessschritt SYS.4 Anwendung. Zur Verifikation der Interaktion zusammenhängender Teilsysteme werden bei den Fahrzeugherstellern üblicherweise die Steuergeräte einer einzelnen Fahrzeugdomäne am Domänenprüfplatz (vergleichbar und gegebenenfalls gleichzusetzen mit dem HIL-Wirkkettenprüfplatz) integriert. Fokus dieser Prüfung ist die Stabilität der Kommunikationsbusse und die Verträglichkeit der verschiedenen Steuergeräte unterschiedlicher Zulieferer am gemeinsamen elektrischen Versorgungsnetz.

Um eine Verifikation auf dieser Ebene zu ermöglichen, müssen die einzelnen Steuergeräte in spezifizierte Zustände versetzt werden. Beispielsweise müssen die im jeweiligen Testfall spezifizierten Funktionen aktiviert und entsprechende Fahrereingaben emuliert werden. Neben der Integration von Simulationsmodellen mittels HIL-Methode werden einfachere Aktivierungssignale mit Hilfe sogenannter Restbussimulationen in das Kommunikationsnetz eingespeist. Ein Beispiel für ein einfaches Aktivierungssignal ist die Information über ein geschaltetes Zündungsplus (Klemme 15). Die Laboraufbauten zur Systemintegration und die im HIL-Kontext beschriebenen Wirkkettenprüfplätze und Gesamtfahrzeugprüfplätze sind üblicherweise identisch.

3.5.5 Qualifikationstests

Qualifikationstest werden gemäß der in Abschnitt 2.7 vorgestellten Automotive SPICE Prozessreferenz jeweils zum Abschluss des Software Engineerings und des System Engineerings durchgeführt. Ihre Aufgabe ist es nachzuweisen, dass die entwickelte Software bzw. das entwickelte System mit den definierten Anforderungen übereinstimmt.

Softwarequalifikationstests

Der Softwarequalifikationstest (SWE.6) nutzt die beschriebenen Werkzeuge und Testumgebungen der Softwareintegration und der SIL-Methode. Im Fokus steht die Analyse des funktionalen Verhaltens der integrierten Software und der Nachweis anforderungskonformen Verhaltens. Während bei den Integrationstests insbesondere die Schnittstellen der Software-Items und die Interaktion der Softwarekomponente

mit der Basissoftware im Vordergrund stehen, liegt der Fokus der Qualifikationstests auf der gesamtheitlichen Betrachtung und Analyse der Funktionswirkkette.

Systemqualifikationstests

Systemqualifikationstests dienen der abschließenden Verifikation des Gesamtsystems. Sie werden dem Automotive SPICE Prozessschritt SYS.5 zugeordnet. „Der Zweck des Prozessschritts Systemqualifikation ist sicherzustellen, dass die Prüfung des integrierten Systems Nachweise für die Einhaltung der Systemanforderungen und die Auslieferungsbereitschaft des Systems liefert.“ [45].

Die Systemqualifikationstests zur Verifikation der E/E-Funktionalität werden zu einem großen Teil an den zuvor beschriebenen Wirkketten- und Gesamtsystemprüfplätzen durchgeführt. Die Verifikation im Labor wird durch reale Tests auf dedizierten Prüfgeländen sowie durch die Fahrzeugdauererprobung im realen Straßenverkehr ergänzt. Die durch die jeweiligen Testfahrer zu prüfenden Umfänge werden in Form von abstrakten Fahrmanövern spezifiziert.

3.5.6 Ressourcentests

Ressourcentests ermitteln die CPU-Laufzeit und den Speicherbedarf der Software. In frühen Entwicklungsstadien können erste Abschätzungen zum Ressourcenverbrauch durch statische Analysen und die Ausführung der Software in einer MIL-Umgebung getroffen werden. Analysen zum Ressourcenverbrauch sollten in regelmäßigen Abständen entwicklungsbegleitend durchgeführt werden [128]. Damit kann sichergestellt werden, dass die Laufzeit der einzelnen Tasks sowie der statische und dynamische Speicherbedarf mit zunehmender Funktionalität die beantragten und zugesicherten Ressourcen nicht überschreitet.

Um durch dynamische Ressourcentests eine möglichst zuverlässige Abschätzung zu erreichen, sollte parallel eine Überdeckungsanalyse durchgeführt werden [128]. Andernfalls ist nicht sichergestellt, dass alle ressourcenintensiven Zweige der Software auch durchlaufen wurden. Beispielsweise können Testszenarien, die auf MIL oder SIL Ebene eine entsprechende Testabdeckung erzeugen, für die Prüfung der Ressourcenlast auf der Zielhardware verwendet werden.

3.6 Erprobung

Die Automobilbranche stützt sich in der Wettbewerbsdifferenzierung einerseits auf objektive Kriterien, wie gesenkte Verbrauchszahlen, gesteigerte Antriebsleistung und innovative Features, andererseits sprechen die verwendeten Werbeslogans häufig die Emotionen der Kunden an. BMW wirbt mit der „Freude am Fahren“[4], Mercedes-Benz mit „Das Beste oder nichts“[5] und Porsche mit der „Porsche Intelligent Performance“[6]. Um diese individuellen und subjektiven Ansprüche an die Fahrzeuge der eigenen Marke zu validieren, führen die Automobilhersteller ausgiebige und weltweit ausgedehnte Erprobungen durch. Diese werden durch die für die Funktionsentwicklung zuständigen Mitarbeiter und Applikationsingenieure sowie Mitarbeitern von auf die Fahrzeugerprobung spezialisierten Firmen durchgeführt.

Während früher Phasen der Entwicklung werden die groß angelegten Erprobungen im kleinen nachgestellt. Fahrsimulatoren bilden das Fahrerlebnis virtuell nach und ermöglichen Validierungsstudien in frühen Entwicklungsphasen [142]. Prototypen und Vorserienfahrzeugen ermöglichen die Stichprobennahme im realen Fahrversuch. Zum Abschluss der Entwicklung ermöglicht die „kundennahe Fahrerprobung“ [143] eine breit angelegte Validierung unter Alltagsbedingungen, ist in der Regel aber nicht Teil der Systemqualifizierung nach Automotive SPICE.

3.6.1 Fahrsimulator

Fahrsimulatoren ermöglichen die Validierung prototypisch umgesetzter Fahrfunktionen und bieten den Vorteil reproduzierbare Fahrszenarien zu ermöglichen. Letzteres ermöglicht die Durchführung psychologischer Studien zur Erwartungshaltung von Probanden gegenüber automatisierenden Funktionen [144]. Zusätzlich ermöglicht die Evaluierung prototypischer Fahrfunktionen im Fahrsimulator eine Validierung durch Alltagsnutzer [145], die für Erprobungen in realen Prototypen nicht genügend qualifiziert sind. Zlocki [146] beschreibt die Wichtigkeit einer gründlichen Validierung der Übergabe der Fahraufgabe von einem automatisierenden System an den Fahrer und den Stellenwert von Fahrsimulatoren in der Konzeptphase. Da der Dynamikumfang von Fahrsimulatoren begrenzt ist, ist eine vollständige Übertragung der Ergebnisse auf die Realität nicht immer gegeben [142].

[4] http://www.wiwo.de/unternehmen/handel/bmw-freude-am-fahren/6190828-2.html
[5] https://blog.daimler.de/2010/06/10/das-beste-oder-nichts/
[6] http://www.kms-team.com/de/referenzen/porsche#details

(a) Statischer Fahrsimulator mit grundlegenden Cockpit-Komponenten.

(b) Statischer Fahrsimulator basierend auf einer realen Fahrzeugkarosserie.

Abbildung 3.11: Beispiel für statische Fahrsimulatoren für die Analyse des Nutzerverhaltens.

Mit der Verfügbarkeit leistungsfähiger Gesamtfahrzeugsimulationen hat sich eine große Variantenvielfalt im Bereich der Fahrsimulationen entwickelt. Statische Simulatoren reichen von Aufbauten die wie in Abbildung 3.11a lediglich die grundlegenden Komponenten des Cockpits beinhalten zu Simulatoren die, wie in Abbildung 3.11b auf realen Fahrzeugkarosserien basieren und damit eine verbesserte Immersion ermöglichen. Aufwändigere dynamische Simulatoren nutzen Hexapoden um Cockpit Mock-Ups oder vollständige Fahrzeuge im Raum zu bewegen und den visuellen Eindruck der Probanden durch real spürbare Beschleunigungen und Drehraten zu unterstützen. Für den Test von Funktionen des Antriebsstrangs existieren auch mit Rollenprüfständen gekoppelte Simulatoren.

3.6.2 Realer Fahrversuch

Die Validierung von assistierenden und automatisierenden Funktionen, die die Fahrdynamik beeinflussen, erfordert erfahrene Prüfer. Im Fahrversuch werden Fahrzeuge unter genau spezifizierten Bedingungen getestet. Die durch professionelle Testfahrer durchgeführten Fahrversuche basieren dabei auf definierten Prüf- und Fahrmanöverkatalogen [147]. Die Prüfung durch den Fahrversuch erfolgt sowohl auf öffentlichen Straßen, als auch auf dedizierten Prüfgeländen. Insbesondere die Validierung von sicherheitskritischen Fahrfunktionen wird auf dedizierten Prüfgeländen durchgeführt, deren Verfügbarkeit sehr begrenzt ist [148]. Baake [149] beschreibt ein Vorgehen, das den realen Fahrversuch und die simulationsbasierte Absicherung von ESP-Systemen verbindet. Eckstein [150] kombiniert den Einsatz eines Fahrsimulators mit Prüfgeländetests für die Validierung aktiver Sicherheits-

funktionen.

Die Eigenschaften und das Fahrverhalten einer Funktion sollen die markenspezifische Charakteristik widerspiegeln. Eine allgemeingültige Spezifikation des erwarteten Fahrverhaltens in Form von Kennfeldern zu Messgrößen, wie Beschleunigungen und Ruck, existiert aufgrund der benötigten Komplexität zur Abbildung spezifischer Situationen in der Regel nicht. Für spezifische Funktionen hingegen existieren analytische Methoden zur Bewertung der Funktionsqualität [151].

Um eine entwicklungsbegleitende Validierung zu ermöglichen, werden im Verlauf der Entwicklung, wie in Abschnitt 3.3.3 eingeführt, Erprobungsträger mit unterschiedlichen Musterständen ausgerüstet [152]. Die Musterstände ermöglichen eine sukzessive Hinzunahme der in der Baureihe vorgesehenen Kundenfunktionen. Am Ende der Entwicklung wird die Produktions- und Verkaufsfreigabe basierend auf den Ergebnissen der Test- und Qualifikationsphasen sowie der Fahrversuche mit dem D-Musterstand erteilt. Die Freigabe beinhaltet zudem die Ergebnisse der Winter- und Sommererprobung [153, 154] und länderspezifische Erprobungen.

Diese Freigabepraxis ist nach Wachenfeld [7] für automatisierende Fahrfunktionen nicht mehr anwendbar. Aus diesem Grund werden Aufwände für zukünftige Freigaben aus dem Fahrversuch vermehrt in virtuelle Tests verlagert [155]. Die Übertragung der Versuchsaktivitäten in eine virtuelle Umgebung stellt dabei eine große Herausforderung dar. Für die Aufzeichnung aller im Fahrzeug verbauten Sensoren und Kommunikationsbusse werden bereits heute hohe Aufwände in Kauf genommen. Gerechtfertigt wird dies unter anderem durch die Möglichkeit diese Daten in Form von Testvektoren am HIL wiederzuverwenden [129]. Einen entscheidenden Anteil zur Lösung dieser Herausforderung können die im folgenden Abschnitt 3.7 vorgestellten Methoden zu Testplanung und -entwurf beitragen und weiterentwickelte Methoden eröffnen zusätzliche Chancen um die Entwicklung automatisierender Fahrfunktionen entscheidend zu unterstützen.

3.6.3 Kundennahe Fahrerprobung

Die kundennahe Fahrerprobung, auch Breitenerprobung genannt, ermöglicht die Erprobung der entwickelten Features und Funktionen aus der Kundensicht [143, 147]. Das Unternehmen stellt dabei seinen Mitarbeitern und ausgesuchten Kundengruppen Fahrzeuge für die Freizeitnutzung zur Verfügung und erhält im Gegenzug

Feedback über den subjektiven Eindruck des jeweiligen Mitarbeiters. Üblicherweise erfassen detaillierte Fragebögen den gesamtheitlichen Eindruck des Mitarbeiters sowie die spezifischen Fragestellungen bezüglich einzelner Features und Funktionen. Dies ermöglicht einen Einblick in realistische Verhaltensmuster und ergänzt die entwicklungsspezifischen Ergebnisse des Fahrversuchs.

3.7 Testplanung und -umfang

Für eine erfolgreiche Systementwicklung muss aus den vorgestellten Methoden eine ausgewogene und auf die individuellen Eigenschaften der Subsysteme, Komponenten und Einheiten abgestimmte Auswahl getroffen werden. Diese Auswahl und der Umfang ihrer Anwendung wird im Rahmen der in Abschnitt 3.2 eingeführten Teststrategie getroffen. Aufgrund der großen Vielfalt der einzelnen Entwicklungen innerhalb einer Organisation gibt die Teststrategie lediglich einen Rahmen vor, innerhalb dessen die explizit anzuwendenden Methoden und Umfänge im Testplan spezifiziert werden. Im Folgenden werden zunächst Methoden für den Entwurf und die Auswahl geeigneter Testfälle vorgestellt und in Abschnitt 3.7.2 werden Metriken zur Bestimmung des Umfangs der umgesetzten Tests eingeführt.

3.7.1 Testentwurf und -auswahl

Der internationale Standard ISO/IEC/IEEE 29119 „Software and systems engineering - Software testing“ gruppiert die Testentwurfsmethoden für die dynamische Verifikation in die drei in Abbildung 3.12 dargestellten Kategorien spezifikationsbasierter Testentwurf, strukturbasierter Testentwurf und erfahrungsbasierter Testentwurf [156].

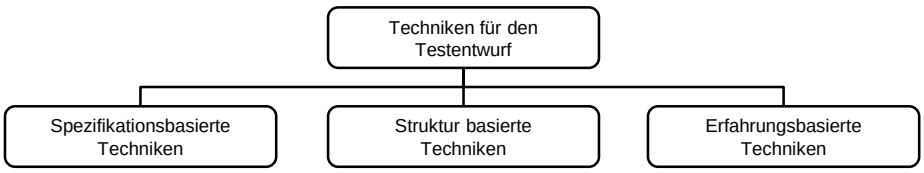


Abbildung 3.12: Unterscheidung von Testentwurfsmethoden nach ISO/IEC/IEEE 29119 [156]

Spezifikationsbasierte Techniken nutzen die für die Entwicklung identifizierten Anforderungen und Kundenwünsche, sowie möglicherweise erstellte Modelle (z.

B. Aktivitätsdiagramme, Abschnitt 3.3.1) als wichtigste Quelle zur Spezifikation der Testfälle. Diese Techniken können auch unter dem Begriff Blackbox-Verfahren zusammengefasst werden [55]. Der Begriff Blackbox symbolisiert dabei die Abstraktion aller internen und auf die Umsetzung bezogenen Informationen und die Fokussierung auf die externe Sicht der Spezifikation des jeweiligen Elements. Strukturbasierte Techniken nutzen primär den Quellcode oder erstellte Programmmodelle (z. B. Sequenzdiagramme, Abschnitt 3.3.1) zur Herleitung von Testfällen. Da diese Verfahren detaillierte Kenntnisse über das Testobjekt erfordern, spricht man hier auch von Whitebox-Verfahren [55]. Erfahrungsbasierte Techniken nutzen die Erfahrung der Entwickler zur Spezifikation von Testfällen. Beim Schätzen von Fehlerzuständen (Error Guessing) wird eine Prüfliste mit möglichen Typen von Fehlerzuständen erstellt, die im Testobjekt vorkommen können. Der Tester identifiziert basierend auf dieser Liste Stimuli, die diese Fehlerzustände aufdecken würden [156]. Blackbox und Whitebox Verfahren sind nach [19] wie folgt definiert:

Definition 3.38 (Blackbox Verfahren)**.** Blackbox Verfahren *beschreiben das Testen unter Betrachtung des Testobjekts als Ganzes, d.h. ohne interne Systemgrößen zu berücksichtigen.*

Definition 3.39 (Whitebox Verfahren)**.** Whitebox Verfahren *beschreiben Testverfahren, bei der die innere Inforamtion des Testobjekts genutzt wird und auf Systemgrößen innerhalb des Testobjekts zugegriffen wird.*

Die **Äquivalenzklassenbildung** (Equivalence Partitioning) basiert auf der Unterteilung der möglichen Eingabe- und Ausgabewerte in Äquivalenzklassen, „die durch das Testobjekt gleichwertig behandelt werden“[156]. Die Klassifizierung kann dabei semantisch erfolgen, beispielsweise entsprechend der Zugehörigkeit zu abstrakten Gruppen wie Beschleunigung und Verzögerung. Im Fall mathematischer Ausdrücke erfolgt die Äquivalenzklassenbildung entsprechend spezifizierter Definitionsbereiche. Beispielsweise können die möglichen Eingabewerte einer Sinusfunktion in Bereiche mit erwartetem positivem $((0, \pi) + 2n, n \in N)$, negativem Ergebnis $((1, 2\pi) + 2n, n \in N)$ und Null als Ergebnis $(n\pi, n \in N)$ unterteilt werden. Die Äquivalenzklassen werden sowohl für gültige als auch für ungültige Parameter gebildet.

Für jede abgeleitete Äquivalenzklasse wird mindestens ein Testfall generiert. Wird eine höhere Testtiefe gefordert, können ergänzende Testfälle durch eine statistisch

zufällige Auswahl von Werten aus der jeweiligen Klasse erstellt werden. Bei Funktionen mit mehreren Parametern müssen die Permutationen der unterschiedlichen Äquivalenzklassen geprüft werden. Ist eine vollständige Permutation zu aufwändig, wird mindestens für jede ungültige Klasse ein Testfall erstellt, in dem diese ungültige Klasse lediglich mit gültigen kombiniert wird [55].

Die **Grenzwertanalyse** (Boundary Value Analysis) fokussiert die Testabdeckung auf den Grenzen der abgeleiteten Klassen, da diese Übergänge besonders fehleranfällig sind. Die Wertebereiche der Ein- und Ausgänge werden „in geordnete Mengen und Untermengen mit identifizierbaren Grenzen„ [156] unterteilt. Jede dieser Grenzen stellt einen Testfall dar. Die aus der Grenzwertanalyse abgeleiteten Testfälle ergänzen die Testfälle der Äquivalenzklassenbildung [55]. Mathematische Funktionen werden durch die kombinierte Anwendung der beiden Methoden an mehreren zufällig gewählten Stellen innerhalb eines Definitionsraums, auf der Grenze des Definitionsraums und außerhalb des Definitionsraums geprüft.

Der **Ursache-Wirkungs-Graph** (Cause-Effect Graphing) dient der Veranschaulichung der kausalen Abhängigkeiten zwischen einer Eingangsgröße und einer Ausgangsgröße. Der Ursache-Wirkungs-Graph modelliert die logischen Beziehungen zwischen Ursachen (Eingänge) und Wirkungen (Ausgänge) des Testobjekts [156] in Form eines gerichteten Graphen. Logische Operatoren, wie UND und ODER, kombinieren Ursachen und ordnen sie einer gemeinsamen Wirkung zu. Für jede abgeleitete Beziehung, die eine einmalige Kombination zwischen Ursache und Wirkung erzeugt, wird ein Testfall angelegt.

Der **zustandsbezogene Test** (State Transition Testing) basiert auf der Modellierung möglicher Zustände eines Testobjekts und der möglichen Zustandsübergänge inklusive deren Bedingung und Aktionen. Eine Bedingung kann aus mehreren logischen Ausdrücken bestehen, die in Kombination den Zustandsübergang auslösen. Die Anzahl der Zustände ist diskret und begrenzt. Testfälle werden für alle Zustände oder alle Zustandsübergänge abgeleitet [156].

Der **Szenarien-Test** (Scenario Testing) leitet sich aus einem Modell der Interaktionen des Testobjekts mit anderen Systemen oder Nutzern ab. Die Interaktionen werden in Form von Sequenzen abgebildet und stellen die Verwendung des Testobjekts nach. Für jede Sequenz, die eine dedizierte Verwendung darstellt, wird ein Testfall definiert [156].

3.7.2 Testabdeckung

Die Ermittlung der Testabdeckung ermöglicht eine Abschätzung des Testumfangs und unterstützt damit die Bestimmung des Abschlusses einer Testphase [55]. Die Testabdeckung wird gemäß ISO 29119-4 [156] mit der folgenden Formel

$$Abdeckung = \left(\frac{N}{T} \cdot 100\right) \% \tag{3.1}$$

berechnet. Und ist nach [57] wie folgt definiert:

Definition 3.40 (Testabdeckung)**.** *Die* Testabdeckung *ist der Grad in Prozent, zu dem spezifizierte Testobjekte durch einen oder mehrere Testfälle beansprucht wurden.*

Dabei stellt N die Anzahl getesteter Instanzen dar und T definiert die Gesamtzahl der durch eine Testentwurfsmethode erzeugten Instanzen [156]. Die Testabdeckung kann damit Werte zwischen 0% und 100% einnehmen. Die innerhalb einer Phase zu erreichende Abdeckung wird im Testplan spezifiziert. Nicht alle für die Berechnung der Abdeckung relevanten Instanzen können durch Tests überprüft werden. Teilweise können Instanzen aufgrund technischer Voraussetzungen nicht geprüft werden, wie im Folgenden am Beispiel der Mehrfachbedingungsüberdeckung beschrieben wird. Diese Instanzen werden aus der Abdeckung ausgenommen und im Testbericht aufgezeichnet.

Eine entsprechende Ermittlung der Testabdeckung ist für alle Testentwurfsmethoden möglich, aber unter Umständen mit hohen Aufwänden verbunden. Für die Ermittlung der strukturbasierten Testabdeckung existieren Werkzeuge zur automatischen Berechnung. Diese Werkzeuge analysieren den Kontrollfluss basierend auf den Debug Informationen des Compilers (z. B. CPP-Coverage [157]) oder integrieren in den Compilevorgang eine automatisierte Instrumentierung des Quelltextes (z. B. BullseyeCoverage [158] oder gcov [159]). Im folgenden werden ausgewählte Vertreter der Testabdeckung vorgestellt.

Die **Anforderungsüberdeckung** (Requirements Coverage) ist das Verhältnis zwischen der Anzahl der Anforderungen, für die mindestens ein Testfall spezifiziert ist,

und der Gesamtanzahl an spezifizierten Anforderungen. Eine eiffiziente Ermittlung dieser Abdeckungsmetrik erfordert eine durchgängige Nachverfolgbarkeit und die enge Verzahnung von Test- und Anforderungsmanagement Werkzeugen [160]. Die Anforderungsüberdeckung berechnet sich aus der Anzahl getesteter Anforderungen (N) und der Gesamtzahl der Anforderungen (T).

Die **Anweisungsüberdeckung** (Statement Coverage) ermittelt den Anteil, der während einer Testphase ausgeführten Programmanweisungen. Sie ist die einfachste Form der strukturellen Überdeckungsanalyse [128], da lediglich die mindestens einmalige Ausführung jeder Anweisung geprüft wird. Die Anweisungsüberdeckung berechnet sich aus dem Verhältnis der Anzahl ausgeführter Anweisungen (N) und der Gesamtzahl der Anweisungen.

Die **Zweigüberdeckung** (Branch Converage) prüft die Ausführung aller Zweige im Kontrollfluss. Es wird ermittelt, ob während des Durchlaufs einer Testphase alle durch Bedingungen oder Schleifen auszuführende Zweige aufgerufen werden [156]. Die Zweigüberdeckung ermittelt sich aus der Anzahl durchlaufener Zweige (N) und der Gesamtzahl der Zweige (T).

Die **Bedingungsüberdeckung** (Branch Condition Coverage) misst die Variation aller Einzelelemente einer Bedingung. Es wird geprüft, ob während der Testphase jeder atomare Teil der Bedingung mindestens einmal logisch positiv und einmal logisch negativ ausgewertet wurde und alle aus der Bedingung erreichbaren Zweige durchlaufen wurden. Die Überdeckung berechnet sich aus dem Verhältnis [156] der Anzahl getesteter boolscher Einzelbedingungen plus der Anzahl resultierender Einzelergebnisse (N) und der Anzahl enthaltener boolscher Einzelbedingungen plus der Anzahl möglicher Einzelergebnisse (T).

Die **Mehrfachbedingungsüberdeckung** (Branch Condition Combination Coverage) hat das Ziel alle möglichen Permutationen der atomaren Teile einer Bedingung zu überprüfen. Dies ist nicht für alle Bedingungen vollständig möglich. Beispielsweise kann die Auswertung `(5 < x \: \&\& \: x < 10)` ($5 < x \,\&\&\, x < 10$) im Bereich der natürlichen Zahlen nicht für beide atomaren Teilbedingungen gleichzeitig einen logisch negativen Wert annehmen.

Die **minimale Mehrfachbedingungsüberdeckung** (Modified Condition Decision Coverage) definiert daher die relevante Untermenge an Permutationen. Sie „verlangt, dass jede der Teilbedingungen, die auf eine Programmverzweigung Einfluss haben kann, zeigen muss, dass sie unabhängig von den anderen den Programmfluss be-

stimmen kann“ [128]. Für das Beispiel ($5 < x \,\&\&\, x < 10$) würden drei Testfälle benötigt. Ein Testfall, bei dem beide Einzelauswertungen gleichzeitig wahr sind und je ein Testfall mit je einer negativen Einzelbedingung. Die minimale Mehrfachbedingungsüberdeckung berechnet sich aus dem Verhältnis zwischen der Anzahl der geprüften Kombinationen mit eigenständiger Ergebnisbeeinflussung (N) und der Gesamtanzahl der Kombinationen mit eigenständiger Ergebnisbeeinflussung (T).

Die **Parameterüberdeckung** (Parameter Value Coverage) spezifiziert den durch Tests abgedeckten Wertebereich kontinuierlicher Signale. Boehm [161] beschreibt die Parameterüberdeckung ausgehend von der Sicht auf eine Softwarefunktion als die Abbildung eines mehrdimensionalen Raumes von Eingangswerten auf einen mehrdimensionalen Raum von Ausgangswerten. Die Parameterüberdeckung ist damit das Verhältnis der getesteten Stichprobe zur gesamten Menge der möglichen Ein- und Ausgangswerte. Sie kann für einzelne Systemein- und Systemausgänge, aber auch für deren Kombinationen berechnet werden.

3.7.3 Testabdeckung in der Automobilentwicklung

In der Automobilentwicklung kommen spezifische Abwandlungen und Erweiterungen der vorgestellten Metriken zur Messung der Testabdeckung zum Einsatz. Nörenberg [152] definiert mit der BCE-Methode eine Methode zur Auswahl einer geeigneten Menge an Testfällen auf Systemebene. Testfälle werden dabei in die Kategorien

- „Basic: Selektion der Basistestfälle“
- „Coverage: Selektion von Testfällen zur Erreichung einer Anforderungsabdeckung“
- „Extended: Selektion von Testfällen, welche den Abdeckungsgrad in Bezug auf das Testobjekt stark erhöhen (Expertenwissen)“

eingeteilt und entsprechend der angestrebten Testtiefe ausgewählt. Auf der Basic Stufe wird lediglich die Grundfunktionalität eines Testobjekts geprüft. Auf der Coverage Stufe werden 100% Anforderungsabdeckung angestrebt und für alle als Extended klassifizierten Testobjekte werden „die Testfälle ausgewählt, die in einem

hohen Maße zur Testabdeckung beitragen“.

Sax [162] präsentiert mit dem NSPLF-Schema eine ähnliche, aber detailliertere Klassifizierung in fünf Kategorien:

- **N**one: Zufälliger oder impliziter Test des Testobjekts,
- **S**parsely: Mindestens ein Testfall pro Ausstattungsmerkmal,
- **P**artly: 100% Anforderungsüberdeckung wird erreicht,
- **L**argely: 100% Bedingungsüberdeckung wird erreicht,
- **F**ully: Basierend auf Erfahrungswissen, werden weitere Testfälle ergänzt.

Die erreichte Abstraktion in der Beschreibung der Testabdeckung unterstützt durch die Reduktion des Spezifikationsraums die Festlegung einer einheitlichen Teststrategie und vereinfacht die Kommunikation zwischen den beteiligten Rollen, Teams und Organisationen. Die vorgestellte Kategorisierung ermöglicht es, jedem spezifizierten Element eine Abdeckungsstufe zuzuweisen.

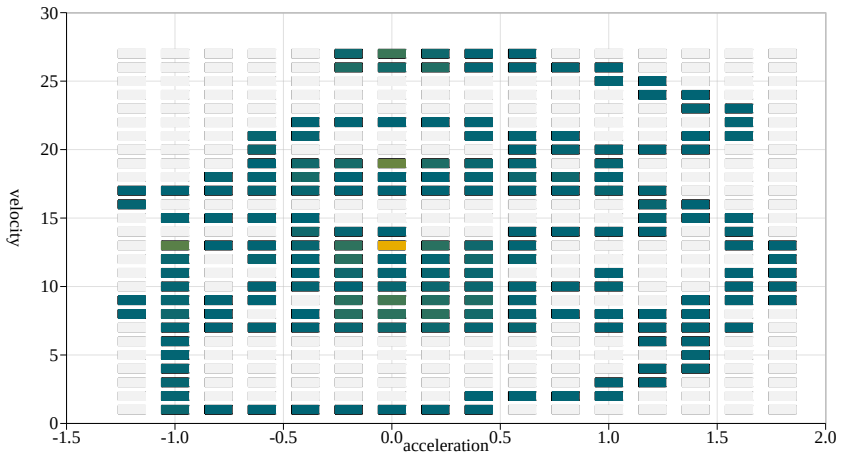


Abbildung 3.13: Histogramm basierte Analyse der Parameterüberdeckung der Systemeingänge Geschwindigkeit (velocity) und Beschleunigung (acceleration) eines Simulationsszenarios

Der internationale Standard für die funktionale Sicherheit von Straßenfahrzeugen ISO 26262 [59] empfiehlt die Erfüllung der Anweisungsüberdeckung besonders für

ASIL-A und ASIL-B klassifizierte Elemente. Die Anwendung der Zweigüberdeckung wird besonders für die Stufen ASIL-B und ASIL-C empfohlen. Für die kritischste Einstufung ASIL-D wird die Erfüllung der MCDC-Überdeckung besonders empfohlen.

Wittmann [163] beschreibt einen Ansatz zur Definition und Identifikation der Systemgrenzen für hochautomatisierende Fahrfunktionen. Der Spezifikationsraum wird als Kombination der funktional unabhängigen Komponenten Umgebung, Verkehrsdynamik, Umweltbedingungen, Zustand des Egofahrzeug und Insassen beschrieben, die wiederum in beliebig detaillierte Unterräume aufgespalten werden. Diese Dekomposition ermöglicht einen der Anwendung angepassten Detaillierungsgrad.

Überträgt man die durch Wittmann vorgeschlagene Dekomposition des Spezifikationsraums auf die Verifizierung und Validierung, ermöglicht sie eine Messung der Parameterüberdeckung beim Test hochautomatisierender Fahrfunktionen. Dies kann einen entscheidenden Beitrag zur Bestimmung des notwendigen Testumfangs für „Die Freigabe des autonomen Fahrens“ [7] liefern. Abbildung 3.13 skizziert eine Histogramm basierte Analyse der Parameterüberdeckung der Systemeingänge Geschwindigkeit und Beschleunigung eines Simulationsszenarios.

3.8 Fazit

Die eingesetzten Methoden für die Entwicklung, Verifikation und Validierung von Softwaresystemen für Fahrzeuge sind auf die in Kapitel 2 vorgestellten Prozesse und Prozessphasen abgestimmt. Für die im Verlauf der Dekomposition und Integration durchlaufenen Hierarchieebenen werden auf die Ziele der jeweiligen Prozessphase ausgelegte Entwicklungs- und Testmethoden angewandt. Abbildung 3.14 skizziert eine qualitative Einordnung der in diesem Kapitel vorgestellten Methoden für Entwicklung, Verifikation und Erprobung im schematischen Vorgehen der Produktentwicklung.

In den frühen Phasen der Anforderungsanalyse und des Architekturentwurfs von System und Software kommen simulationsbasierte Ansätze (MIL) und Rapid RP-Techniken (A-Muster) für Test und Erprobung auf Systemebene zum Einsatz. Manuelle Prüfungen tragen zur Verifikation der erstellten Spezifikationen bei. Während der Implementierung auf Einheitenebene stellen Werkzeuge zur statischen Codeanalyse automatisiert die Einhaltung von Programmierrichtlinien und

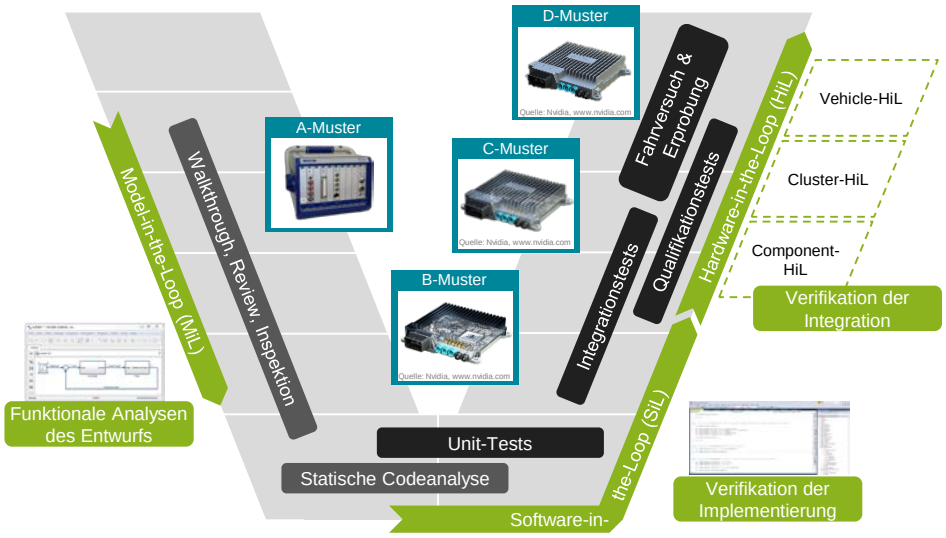


Abbildung 3.14: Qualitative Einordnung der vorgestellten Methoden für Entwicklung, Verifikation und Erprobung im schematischen Vorgehen der Produktentwicklung

Standards sicher und Unit-Test prüfen die korrekte Umsetzung des spezifizierten Entwurfs. Die Durchführung von Code-Reviews trägt zu einer gesteigerten Qualität bei und unterstützt den Wissenstransfer zwischen beteiligten Entwicklern. Die Durchführung von Tests und Erprobungen auf System- und Subsystemebene wird durch SIL-Simulation und die Nutzung des B-Musterstands ermöglicht. SIL- und HIL-Umgebungen ermöglichen die notwendigen Integrations- und Qualifikationstests der Komponenten und Subsysteme. Fahrversuche und Erprobungen mit C- und D-Musterständen weisen die zur Freigabe notwendige Qualität des Systems nach.

Die beschriebenen Methoden decken damit alle Phasen der Produktentwicklung ab. Mit zunehmender Komplexität wächst die Bedeutung der frühen Entwicklungsphasen und die Analyse von Entwurfsalternativen mit Hilfe von A-Mustern und der MIL-Simulation gewinnt an Wichtigkeit, um Fehlentscheidungen im Umgang mit diesem großen Entwurfsraum zu minimieren. Um eine ausreichende Qualität dieser prototypischen Umsetzungen zu gewährleisten, nehmen die Aufwände für grundlegende regelmäßige Tests bereits in diesen frühen Phasen zu. Wird ein Konzept verworfen, sind die Aufwände für die spezifizierten Testfälle und -szenarien jedoch häufig verschwendet. Eine ressourcenschonende Alternative können hier Verfahren bieten, die Testszenarien automatisiert generieren oder aus in der Erprobung aufgezeichneten Daten ableiten.

4 Einordnung aktueller Serienanwendungen und Forschungskonzepte des Automobilsektors

Die Fahrzeughersteller bieten ihren Kunden eine große Vielfalt an individuell wählbaren zusätzlichen Ausstattungsmerkmalen (Features). Ein wesentlicher Anteil dieser Features basiert dabei auf E/E-Funktionalität. Die Komplexität der einzelnen Features und damit die Quervernetzung der beteiligten Funktionselemente nimmt kontinuierlich zu. Der in diesem Kapitel präsentierte Überblick über die aktuellen und kommenden Features mit wesentlichen E/E-Anteilen ermöglicht eine für diese Dissertation notwendige Analyse der in Kapitel 2 und 3 vorgestellten Prozesse und Methoden in Bezug auf die neuen Anforderungen an die Automobilentwicklung durch die Digitalisierung.

Im Folgenden werden Kriterien zur Charakterisierung und Strukturierung E/E-basierter Features eingeführt. Die Klassifikation erlaubt eine verallgemeinerte Betrachtung der featurespezifischen Anforderungen an die Entwicklungsprozesse und -methoden. Daraus lässt sich die Eignung einzelner Vorgehensweisen und Ansätze für die jeweilige Featureklasse ableiten. Die verwendete Charakterisierung betrachtet die bestehenden organisatorischen Strukturen und Fachgebiete der Fahrzeughersteller sowie die systemtheoretische Komplexität und den Automatisierungsgrad der Features. Für die Taxonomie wurden Serienfeatures ausgewählt, die einen Querschnitt über die angebotenen Ausstattungsmerkmale führender Fahrzeughersteller repräsentieren. Sie entstammen den Webauftritten von BMW[1], Daimler[2], Ford[3], Peugeot[4], Toyota[5] und VW[6]. Aktuelle Forschungskonzepte im Bereich des assistierten und automatisierten Fahrens vervollständigen die Taxonomie.

[1] BMW Technology Guide, Bayerische Motoren Werke Aktiengesellschaft, http://www.bmw.com/com/en/insights/technology/technology_guide/index.html

[2] Welcome to the Mercedes-Benz TechCenter, Daimler AG, https://techcenter.mercedes-benz.com/en/index.html

[3] Advanced technology at your fingertips, Ford Motor Company, http://www.ford.com/cars/focus/features/#page=FeatureCategory4

[4] Technologies & Innovations, Automobiles Peugeot, http://www.peugeot.com/en/technology

[5] Toyota Technology, Toyota Motor Sales, U.S.A., Inc., http://www.toyota.com/technology/

[6] Technik auf den Punkt gebracht., Volkswagen AG, http://www.volkswagen.de/de/technologie/technik-lexikon.html

Darauf folgend werden die individuellen Funktionsblöcke der in der Taxonomie klassifizierten Features abstrahiert und in einem konzeptuellen Modell der übergreifenden und umfassenden logischen Systemarchitektur für intelligente Fahrzeuge zusammengefasst. Diese abstrahierte Betrachtung ermöglicht eine Hierarchisierung der Funktionsblöcke gemäß des Charakters der durch sie verarbeiteten Informationen. Sie bietet eine übersichtliche Darstellung der gesamtheitlichen Zusammenhänge und funktionalen Schnittstellen des Fahrzeugs. Die Einordnung in Hierarchieebenen fördert die verallgemeinerte Betrachtung der Anforderungen an die Entwicklungsprozesse und -methoden und deren Auswahl. Teile dieses Kapitels wurden bereits im Rahmen eines Konferenzbeitrags veröffentlicht [JB2].

4.1 Charakterisierung von E/E- und Software-Features

Die in Abschnitt 2.2 eingeführte Aufteilung in Fahrzeugdomänen stellt ein wichtiges Kriterium zur Klassifizierung von eng mit der Mechanik verzahnten E/E-Features, wie Motorsteuerung und Stabilitätskontrolle, dar. In den langjährig bewährten Fahrzeugdomänen werden inhaltlich ähnliche Features und das spezifische Fachwissen gebündelt. Die Domänen bilden die erste Grundlage zur Charakterisierung und Einordnung der Features in der entwickelten Taxonomie.

Die Aufteilung in Fahrzeugdomänen ist jedoch nicht ausreichend zur Klassifizierung von Fahrerassistenzsystemen geeignet, da diese zunehmend komplexe und domänenübergreifende Funktionswirkketten und -netzwerke bilden. Eine weitergehende Charakterisierung der Features ermöglicht die in Abschnitt 2.1 vorgestellte Spezifikation der Aspekte und Gründe für die Komplexität in der Automobilentwicklung. Sie ermöglicht eine objektive Ermittlung der individuellen Komplexität eines Features. Dabei stellen die Vielfalt und die Konnektivität geeignete Merkmale zur objektiven Klassifizierung der Komplexität eines Features dar. Die Vielfalt wird durch die Anzahl an verschiedenen zur Funktionserbringung beteiligten Funktionsblöcken der logischen Funktionsarchitektur erfasst. Die Konnektivität ermittelt sich aus den Relationen zu Funktionsblöcken paralleler Features, externer Infrastruktur und kommunikationsbasierten Diensten.

Neben der Zuordnung zu einer spezifischen Fahrzeugdomäne und der Komplexitätsbewertung stellt der Automatisierungsgrad (s. Abschnitt 2.2.3) des jeweiligen Features ein geeignetes Klassifikationskriterium dar. Die Unterscheidung der Features in Automatisierungsstufen ist für alle die Fahraufgabe übernehmenden Features

geeignet. Für einen großen Teil der verfügbaren Fahrerassistenzfeatures bietet sie jedoch nur eine ungenügende Detaillierung, da diese keine direkte Übernahme der Fahraufgabe enthalten und damit alle in die Automatisierungsstufe 0 eingeordnet werden. Sie wirken durch Hinweise auf den Fahrer oder beeinflussen den Fahrzeugzustand (z. B. Start-Stop) oder die Fahrzeugumgebung (z. B. adaptives Fernlicht). In der Taxonomie werden die Features, wie in Abbildung 4.1 dargestellt, in die Assistenzklassen hinweisend (Stufe 0), unterstützend (Stufe 0), assistierend (Stufe 1 & 2) und automatisierend (Stufe 3+) eingeteilt.

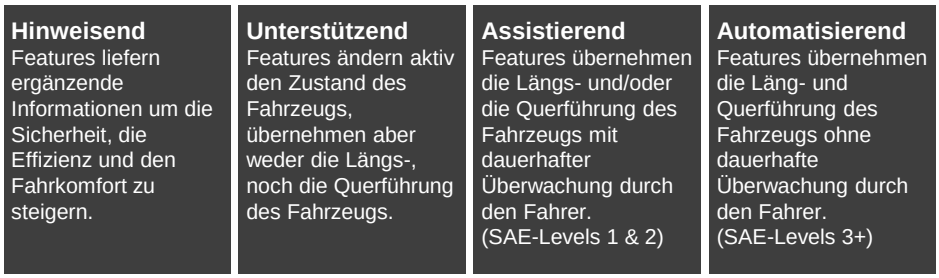


Abbildung 4.1: Im Rahmen der Taxonomie genutzte Assistenzklassen zur Charakterisierung der Features. Die Klassen hinweisend und unterstützend ermöglichen die Unterscheidung von Features der Automatisierungsstufe 0.

Eine weitere wichtige Eigenschaft ist die Sicherheitseinstufung ASIL eines Features. Basierend auf dieser Einstufung werden in der ISO Norm für funktionale Sicherheit von Straßenfahrzeugen [59] anzuwendende Methoden und Testumfänge empfohlen. Die Sicherstellung und Bewertung der funktionalen Sicherheit wird im Rahmen dieser Arbeit nicht betrachtet (s. dazu [164, 18]).

4.2 Taxonomie aktueller und zukünftiger Features mit wesentlichen E/E- und Softwareanteilen

Im Folgenden werden etablierte Funktionen verschiedener Fahrzeughersteller und aktuelle Konzepte aus der Forschung in einer umfassenden Taxonomie eingeordnet. Die bewerteten Features werden in der Taxonomie in die drei Hierarchieebenen integrierte Features, verteilte Features und quervernetzte Features eingeordnet. Die Kategorisierung der Features basiert dabei auf den drei maßgeblichen Kriterien Hardwarenähe, Variabilität und Konnektivität. Jedes Feature wird bezüglich dieser Kriterien durch die fünf Abstraktionsstufen in Tabelle 4.1 bewertet. Eine Grup-

pierung in Fahrzeugdomänen, bzw. in Assistenzklassen innerhalb der jeweiligen Hierarchieebene verbessert die Übersichtlichkeit.

--	-	o	+	++
sehr schwach	schwach	mittelmäßig	stark	sehr stark

Tabelle 4.1: Angewandte Abstraktionsstufen zur Bewertung der Hardwarenähe, Vielfalt und Konnektivität

Integrierte Features sind eng mit einer bestimmten mechanischen Domäne des Fahrzeugs verknüpft. Sie repräsentieren den E/E-Inhalt, der für den gezielten Betrieb der physikalischen Komponenten des Fahrzeugs notwendig ist. Dies bringt eine unmittelbare Nähe zu spezifischen mechanischen Einheiten mit sich und erfordert häufig die Ausführung auf einem dedizierten ECU. Die meisten Sensoren und Aktuatoren, die für die zugewiesene Aufgabe des Merkmals erforderlich sind, sind direkt an die dedizierte ECU angeschlossen. Integrierte Funktionen basieren hauptsächlich auf propriozeptiven[7] Sensoren der Eigenwahrnehmung. Sie erfassen Informationen über den internen Zustand des Fahrzeugs [165]. Ihre Funktionalität stützt sich stark auf die direkte Beeinflussung eines Aktuators und beinhaltet eine schwache Variabilität und Konnektivität.

Verteilte Features kombinieren einzelne Komponenten unterschiedlicher Domänen, um neue Funktionalitäten zu ermöglichen. Diese Features erfordern nicht notwendigerweise zusätzliche mechanische Hardwarekomponenten. Sie generieren einen Mehrwert durch die sequentielle Kombination von verfügbaren Informationen und ansteuerbaren Aktuatoren. Ihr Funktionsverhalten lässt sich durch Wirkketten beschreiben und basiert auf der Verbindung verschiedener Domänen. Verteilte Features basieren häufig auf exterozeptiven[8] Reizen und sie führen dedizierte Sensoren zur Umfeldwahrnehmung in die Fahrzeugarchitektur ein [165]. Ihre durchschnittliche Hardwarenähe, Variabilität und Konnektivität ist mittelmäßig.

Quervernetzte Features verbinden mehrere Funktionselemente miteinander und ihr funktionales Verhalten hängt von der koordinierten Beeinflussung der unabhängigen Komponenten getrennter Domänen ab. Für eine vollständige Repräsentation des Fahrzeugzustandes und der Fahrzeugumgebung nutzen sie das Sensornetzwerk des gesamten Fahrzeugs und fusionieren die propriozeptiven und exterozeptiven Informationsquellen. Mehrschichtige Algorithmen verarbeiten diese Informationen

[7] „Wahrnehmungen aus dem eigenen Körper vermittelnd (z. B. aus Muskeln, Sehnen, Gelenken)“, Duden, http://www.duden.de/rechtschreibung/propriozeptiv

[8] „Reize wahrnehmend, die von außerhalb des Organismus kommen (z. B. mittels Augen, Ohren)“, Duden, http://www.duden.de/rechtschreibung/exterozeptiv

und leiten daraus Fahrempfehlungen ab oder beeinflussen mehrere Aktuatoren. Im Unterschied zu den sequentiellen Funktionsketten der verteilten Funktionen bilden sie funktionale Netzwerke. Die Ebene der quervernetzten Features beinhaltet hochentwickelte kognitive und prädiktive Features, einschließlich hoher Automatisierungsstufen. Sie sind durch eine schwach ausgeprägte Hardwarenähe und eine starke Variabilität und Konnektivität geprägt.

Abbildung 4.2 fasst die Taxonomie in einer Übersicht zusammen. Die Bewertung der Features und ihre resultierende Einordnung wird im folgenden im Detail diskutiert.

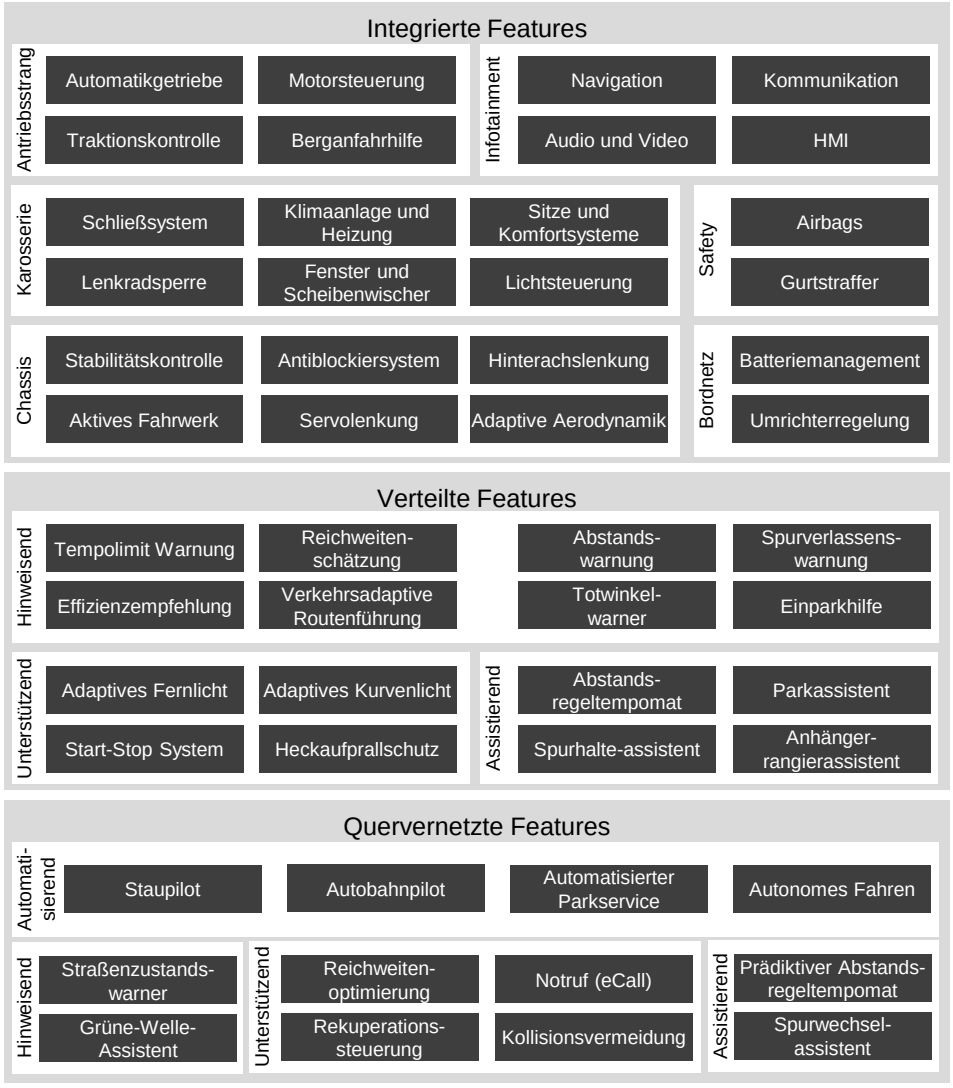


Abbildung 4.2: Taxonomie aktueller und kommender E/E-Features. Integrierte Features sind nach Fahrzeugdomänen gruppiert, verteilte und quervernetzt Features nach dem Grad ihrer Einflussnahme.

4.2.1 Integrierte Features

Die integrierten Features sind geprägt durch eine starke Hardwarenähe und eine schwache Variabilität und Konnektivität. Im Rahmen dieser Taxonomie wurden 24 integrierte Features analysiert. Ihre Einstufung zu Hardwarenähe, Variabilität und Konnektivität sowie ihre Gruppierung innerhalb der Hierarchie und die analysierten Quellen sind in Tabelle 4.2 zusammengefasst.

Feature	**H**	**V**	**K**	**Gruppe**	**Quelle**
Traktionskontrolle	+	-	o	AS	[166, 3]
Motorsteuerung	++	+	--	AS	[16, 3]
Automatikgetriebe	++	-	-	AS	[167, 168]
Berganfahrhilfe	o	--	-	AS	[169, 170]
Airbags	++	o	-	ST	[16, 3]
Gurtstraffer	++	-	-	ST	[16, 3]
Stabilitätskontrolle	+	o	-	CS	[171, 172, 173]
Antiblockiersystem	++	--	--	CS	[173]
Servolenkung	++	-	--	CS	[174, 175]
Aktives Fahrwerk	++	-	o	CS	[176, 177]
Hinterachslenkung	++	-	-	CS	[178]
Adaptive Aerodynamik	++	--	-	CS	[179, 180]
Schließsystem	++	-	o	KR	[181, 182]
Lenkradsperre	++	--	--	KR	[183]
Klimaanlage und Heizung	++	-	-	KR	[3]
Fenster und Scheibenwischer	++	-	--	KR	[184, 3]
Sitze und Komfortsysteme	++	--	-	KR	[3]
Lichtsteuerung	++	--	--	KR	[16]
Navigation	+	-	-	IT	[3]
Audio und Video	++	--	o	IT	[16]
Kommunikation	++	--	o	IT	[185]
HMI	+	-	o	IT	[186]
Batteriemanagement	++	-	--	BN	[187]
Umrichterregelung	++	-	-	BN	[3]

Tabelle 4.2: Ergebnisse der Featureanalyse für Hardwarenähe (H), Vielfalt (V) und Konnektivität(K) sowie die abgeleitete Gruppierung (Antriebsstrang (AS), Safety (ST), Chassis (CS), Karosserie (KR), Infotainment (IT), Bordnetz (BN)) und analysierte Quellen der integrierten Features

Die vorgenommene Bewertung wird im folgenden anhand einiger Beispiele erläutert. Der E/E Anteil eines Automatikgetriebes ist durch eine sehr starke Hardwarenähe charakterisiert, da die Gänge und Kupplungen direkt über elektrohydraulische Aktuatoren gesteuert werden [167]. Als Eingangssignale dienen die in Motor und Getriebe gemessenen Drehzahlen, Momente und hydraulischen Drücke, die eine Funktionslogik verarbeitet [168]. Daraus ergibt sich für das Feature eine schwache Vielfalt und Konnektivität. Die Stabilitätskontrolle unterstützt den Fahrer in kritischen Fahrsituationen durch das gezielte Abbremsen einzelner Räder. Der Bewertung starke Hardwarenähe liegt die funktionale Erweiterung des Bremssystems und die zusätzliche Sensorik zur Messung der Drehraten und Beschleunigungen des Fahrzeugs zugrunde. Die Verknüpfung verschiedener Reglerstufen, wie in [173] beschrieben, begründet die Bewertung mittelmäßiger Vielfalt und die Verknüpfung von Informationen der Domänen Chassis und Antriebsstrang führt zu einer Bewertung schwacher Konnektivität. Das Batteriemanagment überwacht und regelt den Ladezustand des Batteriesystems inklusive der einzelnen Zellen und Module mithilfe von Strom-, Spannungs- und Temperatursensoren [187]. Die Bewertung sehr starke Hardwarenähe des Batteriemanagements ist durch die tiefe Integration des Systems mit der Elektrischen Hardware begründet. Die notwendige Logik und Regelung ist überschaubar, daher wird die Vielfalt als schwach bewertet. Da das System seine Funktion weitgehend autark erfüllen kann, wird die Konnektivität mit sehr schwach bewertet.

Die Taxonomie differenziert die integrierten Merkmale in die etablierten Fahrzeugdomänen der E/E-Systemarchitektur, wie Sie in Abbildung 2.2 eingeführt wurden. Sie umfasst die Features Automatikgetriebe, Motorsteuerung, Traktionskontrolle und Berganfahrhilfe als repräsentative Featureauswahl der Antriebsstrang-Domäne.

Die Safety-Domäne der Taxonomie umfasst die passive Sicherheitsausstattung Airbagsteuerung und Gurtstraffer. Aufwändigere aktive Sicherheitsmerkmale sind in die höheren Hierachieebenen der verteilten und quervernetzten Features eingeordnet.

Die Chassis-Domäne verfügt über Funktionen zur Steuerung der Fahrdynamik und sorgt für ein sicheres und attraktives Fahrerlebnis. ESC, ABS und Elektrisch Angetriebene Servolenkung (Electric Power Assisted Steering, EPS) beschreiben Features, die vor allem sicheres und komfortables Fahren ermöglichen. Hinterachslenkung, aktives Fahrwerk und adaptive Aerodynamik unterstützen besonders ein agiles Fahrverhalten. Da die adaptive Aerodynamik großen Einfluss auf die Fahrdynamik nimmt, wird sie als Chassis-Feature geführt und nicht der Karosserie-Domäne

zugeordnet.

Die Karosserie-Domäne umfasst alle in die Fahrzeugkarosserie integrierten Features und deren Komponenten, wie die Lichtsteuerung, Aktuatoren für Fenster und Scheibenwischer, Sitz- und Komfortsysteme, die Innenraumkonditionierung durch Klimaanlage und Heizung, sowie das Schließsystem und die Lenkradsperre.

Die Infotainment-Domäne fasst alle Informations- und Unterhaltungssysteme des Fahrzeugs zusammen. Sie werden in der Taxonomie durch die Features Navigation, Kommunikation, Audio, Video und HMI repräsentiert. Diese sind eng vernetzt und häufig innerhalb einer Hardwarekomponente integriert. Der VW-Konzern fasst sie beispielsweise im Modularen Infotainment Baukasten[9] zusammen. Die Bordnetz-Domäne wird durch die Features Batterie-Management und Wandler-Steuerung repräsentiert, die Teil der 48 Volt-Netze von Hybrid-Elektrofahrzeugen sowie der Hochspannungsnetze von vollelektrischen Fahrzeugen sind.

4.2.2 Verteilte Features

Die Hierarchieebene der verteilten Features beinhaltet hinweisende, unterstützende und assistierende Fahrerassistenzanwendungen. Die 16 für die Taxonomie ausgewählten Features repräsentieren in heutigen Serienfahrzeugen verfügbare Systeme. Ihre Analyseergebnisse, Gruppenzuordnung und die analysierten Quellen sind in Tabelle 4.3 zusammengefasst.

Die verteilten Features der hinweisgebenden Gruppe liefern zusätzliche Informationen für sicheres und komfortables Fahren und beeinflussen das Fahrverhalten. Sie bereiten mit Sensoren erfasste Umfeldinformationen und von integrierten Features bereitgestellte Daten zu fahrzeuginternen Zuständen auf und erzeugen mit Hilfe der Infotainment-Features optische, akustische und haptische Reize. Die Features Abstandswarnung, Spurverlassenswarner, Totwinkelwarner und Einparkhilfe sorgen für zusätzliche Sicherheit bei der Fahrzeugführung. Die Tempolimitwarnung fördert die Einhaltung der Straßenverkehrsordnung und die Effizienzempfehlung beeinflusst das Fahrverhalten, um einen nachhaltigen Fahrstil zu erzielen. Reichweitenschätzung und adaptive Routenführung unterstützen die Entscheidungen des Fahrers bezüglich der ausgewählten Route und Zwischenziele. Die adaptive Routenführung

[9] MIB II: Volkswagen Infotainment System unterstützt Android Auto, Apple Carplay und Mirrorlink, CNET, http://www.cnet.de/88154583/mib-ii-volkswagens-infotainment-system-unterstuetzt-android-auto-und-apple-car-play/

ergänzt die statischen Verkehrsnetzdaten des Navigationssystems mittels Traffic Message Channel (TMC) oder internetbasierte Dienste um Echtzeitinformationen zur Verkehrslage.

Feature	H	V	K	Gruppe	Quelle
Tempolimit Warnung	-	+	o	HW	[188]
Effizienzempfehlung	-	+	o	HW	[189, 190, 191]
Reichweitenschätzung	-	o	o	HW	[192, 193]
Adaptive Routenführung	-	o	+	HW	[194, 195, 196]
Abstandsswarnung	o	+	-	HW	[197, 198]
Totwinkelwarner	o	o	-	HW	[199]
Spurverlassenswarner	o	o	-	HW	[200, 3]
Einparkhilfe	o	-	-	HW	[201]
Adaptives Fernlicht	o	o	-	US	[202]
Adaptives Kurvenlicht	+	o	o	US	[3]
Start-Stop-System	o	+	o	US	[203, 204]
Heckaufprallschutz	o	o	+	US	[205]
Abstandsregeltempomat	o	+	+	AS	[206]
Spurhalteassistent	o	+	+	AS	[207, 208, 209]
Parkassistent	o	+	o	AS	[201, 3]
Anhängerrangier-assistent	o	o	o	AS	[210]

Tabelle 4.3: Ergebnisse der Featureanalyse für Hardwarenähe (H), Vielfalt (V) und Konnektivität(K) sowie die abgeleitete Gruppierung (Hinweisend (HW), Unterstützend (US), Assistierend (AS)) und analysierte Quellen der verteilten Features

Die Gruppe der unterstützenden Features beinhaltet alle Ausstattungsmerkmale die den Fahrzeugzustand aktiv beeinflussen, jedoch keine Längs- oder Querführungsaufgaben übernehmen. Sie umfasst die Anwendungen adaptives Fernlicht und adaptives Kurvenlicht sowie das automatisierte Start-Stop-System des Motors. Der Heckaufprallschutz stellt ein Beispiel für ein aktives Sicherheitsmerkmal zum Insassenschutz dar. Bogenrieder [205] beschreibt einen Ansatz, der einen rückwärts gerichteten Radarsensor verwendet, um eine bevorstehende Heckkollision zu erkennen. Kurz vor dem Aufprall wird das stillstehende Fahrzeug durch eine automatische Bremsdruckerhöhung festgebremst, der Gurtstraffer wird aktiviert und die aktive Nackenstütze wird ausgefahren.

Die assistierenden Features gruppieren Systeme der Automatisierungsstufe 1 und 2 nach SAE und VDA Einstufung. Der Parkassistent und der Anhängerrangierassistent übernehmen den Querführungsanteil der Fahraufgabe, während die Längsführung beim Fahrer verbleibt. Es handelt sich um assistierende Systeme der

Automatisierungsstufe 1. Diese Auswahl wird mit dem Abstandsregeltempomat (Adaptive Cruise Control, ACC), der die Längsführung übernimmt, und dem Spurhalteassistent (Lane Keep Assist, LKA), der die Querführung unterstützt, ergänzt. Ist ein Parallelbetrieb dieser Features möglich, wird das Fahrzeug nach SAE als teilautomatisiertes System der Automatisierungsstufe 2 eingestuft.

4.2.3 Quervernetzte Features

Die quervernetzten Features repräsentieren die Ebene der höchsten Systemkomplexität und Hardwareabstraktion. Die 12 für die Taxonomie ausgewählten Anwendungen vertreten kürzlich durch Fahrzeughersteller eingeführte Features und aktuelle Forschungskonzepte. Tabelle 4.4 fasst ihre Analyseergebnisse, Gruppenzuordnung und die verwendeten Quellen zusammen.

Feature	**H**	**V**	**K**	**Gruppe**	**Quelle**
Straßenzustandswarner	-	+	++	HW	[211]
Grüne-Welle-Assistent	--	+	++	HW	[211]
Reichweitenoptimierung	-	+	++	US	[212, 213]
Rekuperationssteuerung	-	+	+	US	[214]
Norutf (eCall)	-	o	++	US	[215]
Kollisionsvermeidung	-	+	+	US	[216, 217]
Prädiktives ACC	--	++	+	AS	[218, 219]
Spurwechselassistent	--	++	+	AS	[220, 221]
Staupilot	--	++	++	AT	[222, 223, 224]
Autobahnpilot	--	++	++	AT	[225]
Automatisierter Parkservice	--	++	++	AT	[226, 227]
Autonomes Fahren	--	++	++	AT	[85, 228]

Tabelle 4.4: Ergebnisse der Featureanalyse für Hardwarenähe (H), Vielfalt (V) und Konnektivität(K) sowie die abgeleitete Gruppierung (Hinweisend (HW), Unterstützend (US), Assistierend (AS), Automatisierend (AT)) und analysierte Quellen der quervernetzten Features

Die quervernetzten Features der hinweisgebenden Gruppe sind im Manifest des Car2Car-Communication-Konsortiums [211] beschrieben und stellen Forschungskonzepte dar. Der Straßenzustandswarner informiert den Fahrer über gefährliche Straßenverhältnisse und schlägt ggf. eine Routenänderung vor. Die notwendige Information zur Ermittlung des Straßenzustands liefern optische Sensoren und integrierte Features, wie Traktions- und Stabilitätskontrolle. Sie werden inklusive der aktuellen Fahrzeugposition mit Hilfe der Car2Car-Kommunikation oder

eines Back-End-Services zwischen den Verkehrsteilnehmern verbreitet. Der Grüne-Welle-Assistent interagiert mit der Straßenverkehrsinfrasturktur und empfiehlt eine optimale Geschwindigkeit, die andernfalls notwendige Rotlichtstopps vermeidet. Beide Features erfordern eine spurgenaue Lokalisierung, den Zugriff auf verschiedene interne Zustände und die Kommunikationsplattform des Fahrzeugs.

Die Reichweitenoptimierung und die Rekuperationssteuerung repräsentieren zwei Energiemanagementanwendungen aus dem Bereich der Elektromobilität. Die Reichweitenoptimierung berechnet die verbleibende Energie des Fahrzeugs und prognostiziert die benötigte Energie, um das gewünschte Ziel zu erreichen. Die Steuerung energieintensiver Komfortsysteme, wie Heizung und Klimaanlage, sowie eine Begrenzung der Antriebsleistung unterstützt den Fahrer und schafft notwendiges Vertrauen. Die Rekuperationssteuerung prognostiziert die Energieflüsse des Fahrzeugs. Sie sorgt vor langen Rekuperationsphasen für einen niedrigen Ladezustand und begrenzt die Energieentnahme, um Energieverschwendung aufgrund einer Überhitzung der Batterie zu vermeiden [214]. Da diese Features verschiedene Aktuatoren beeinflussen und prädiktive Kartendaten, Verkehrsflussinformationen und interne Zustände für ein optimales Ergebnis erfordern, werden sie in die quervernetzte Ebene eingeordnet. Notruf und Kollisionsminderung runden die unterstützende Feature-Gruppe ab. Dabei handelt es sich um Anwendungen der aktiven Sicherheit, die vor einer drohenden Kollision eingreifen oder nach dem Unfall automatisch Hilfe rufen.

Analog zu den verteilten Features beinhaltet die Assistenzgruppe der quervernetzten Features Anwendungen der Automatisierungsstufe 1 und 2. Der prädiktive Abstandsregeltempomat übernimmt die Längsführungsaufgabe des Fahrzeugs [218]. Auf der Grundlage prädiktiver Kartendaten, propriozeptiver Informationen und exterozeptiver Sensoren wird eine energieoptimale Geschwindigkeitstrajektorie berechnet und eingeregelt. Der Spurwechselassistent führt den Fahrer während eines Fahrspurwechsels auf einer sicheren und komfortablen Trajektorie [220]. Die Berechnung dieser Trajektorie erfordert ein umfangreiches Abbild der Fahrzeugumgebung und eine Bewegungsprädiktion der Verkehrsobjekte [221]. Beide Features basieren auf der Fusion mehrerer Sensorelemente, einer vielschichtigen Informationsverarbeitung und der Ansteuerung unterschiedlicher Aktuatoren.

Alle Funktionen ab der SAE Automatisierungsstufe 3 aufwärts gehören zur Automatisierungsgruppe der quervernetzten Features. Die Beispielfeatures Staupilot, Autobahnpilot, automatisierter Parkservice und autonomes Fahren übernehmen die vollständige DDT. Während die ersten drei Merkmale auf eine spezifische ODD

begrenzt sind, repräsentiert das Feature autonomes Fahren die Automatisierungsstufe 5. Abhängig von ihren Eigenschaften und ihrer Implementierung nutzen alle Automatisierungsfunktionen eine mehr oder weniger umfassende Umweltwahrnehmung und interpretieren die erfasste Szene. Features der Automatisierungsgruppe beinhalten die höchste Vernetzungsdichte innerhalb der Taxonomie.

4.3 Logische Systemarchitektur für intelligente Fahrzeuge

Das in Abschnitt 3.3.1 vorgestellte und in der Automobilbranche etablierte Vorgehen zur Modellierung der Systemarchitektur modelliert das gesamte Funktionsverhalten innerhalb der logischen Systemarchitektur auf einer Ebene. Dabei wird die unterschiedliche Charakteristik und Integrationstiefe der einzelnen Funktionselemente nicht berücksichtigt. Die Darstellung löst die Komplexität der zugrunde liegenden funktionalen Abhängigkeiten und die mehrfache Nutzung einzelner Funktionsblöcke in verschiedenen Wirkketten nur bedingt auf. Die systemtechnischen Prinzipien der Modularisierung, Abstraktion und Hierarchisierung finden nicht in vollem Umfang Anwendung. Um der steigenden Konnektivität und Vielfalt zukünftiger Features, insbesondere des automatisierten Fahrens, gerecht zu werden, ist eine Weiterentwicklung der gesamtheitlichen Systemarchitekturbeschreibung notwendig.

4.3.1 Existierende Ansätze

Die Forschung auf dem Gebiet des automatisierten Fahrens bietet verschiedene Ansätze, die Systemarchitektur der Forschungskonzepte zu beschreiben. Stiller [229] bietet einen kognitiv orientierten Ansatz von Wahrnehmungs-, Planungs- und Handlungsaufgaben. Hierarchisierte Schichten klassifizieren die abstrakte Darstellung der Funktionsblöcke in eine Ebene der Physik, der Systemdynamik, der Situationsanalyse und Verhaltensgenerierung sowie eine Ebene zur Repräsentation von Wissen. Das Architekturkonzept von Bauer [230] wird in eine Missionsschicht, eine Koordinierungsschicht und eine Verhaltensschicht unterteilt. Jede Schicht beinhaltet Elemente zur Umgebungsmodellierung, zur Planung und zur Mensch-Maschine-Interaktion. Höll [231] beschreibt die bisher in der Fahrdynamik vorherrschende dezentrale Reglerarchitektur und die steigenden Herausforderungen aufgrund der Komplexitätszunahme durch neu eingeführte Chassis-Features. Als Lösung wird ein

Konzept zur integrativen Fahrdynamikregelung vorgeschlagen, das die dezentrale Reglerstruktur um zentrale, hierarchische Ansätze erweitert.

Donges [232] unterteilt die Fahraufgabe in die drei Ebenen Navigation, Führung und Stabilisierung. Flemisch [233] nutzt diese Kategorisierung zur Untersuchung des Einflusses von Mensch-Maschine-Interaktionen auf die Systemarchitektur und stellt mit dem Conduct-by-Wire Ansatz ein kooperatives Regelungskonzept zur Verfügung. Dieses bietet dem Fahrer auf jeder Ebene der Fahraufgabe Schnittstellen zur Interaktion mit dem Automatisierungssystem. Auch die von Matthaei [234] vorgeschlagene funktionale Systemarchitektur für ein autonomes Straßenfahrzeug basiert auf der in drei Ebenen unterteilten DDT. Matthaei ergänzt die horizontale Struktur um eine vertikale Differenzierung in die Spalten Lokalisierung, externe Daten, Wahrnehmung und Auftragserfüllung.

Aeberhard [225] liefert einen Überblick über die Forschungsaktivitäten bei BMW Forschung und Technik. Die verwendete logische Systemarchitektur zur Beschreibung des Automatisierungssystems verknüpft die im Serienfahrzeug verbauten Sensoren mit zusätzlichen, für das hochautomatisierte Fahren notwendigen Sensoren. Die Automatisierungsaufgabe ist in einen Teil zur Umgebungserfassung und einen Teil zur Ausführung unterteilt. Die Aktuatoren werden über den Kommunikationsbus des Serienfahrzeugs angesteuert. Buechel [227] stellt den Prototyp eines automatisierten Elektrofahrzeugs vor. Die vorgestellte Softwarearchitektur des Fahrzeugs besteht aus den drei Komponenten Datenfusion, Trajektorienplanung und Trajektorienregelung. Die zentrale Reglerarchitektur geht mit einer Zentralisierung der technischen Systemarchitektur [235, 236] und den bereits in Abschnitt 2.1 eingeführten Domaincontroller-Ansätzen einher.

Elektrobit präsentiert mit *open robinos* [237] eine skalierbare logische Systemarchitektur und Schnittstellenspezifikation für das automatisierte Fahren. Die Architektur ist in Abbildung 4.3 dargestellt und beinhaltet die fünf Schichten:

- Sensordatenfusion (Sensor Data Fusion)
- situationsabhängige Verhaltensentscheidung (Situative Behavior Arbitration)
- Bewegungsmanagement (Motion Management)
- HMI-Management

- Safety- und Fehlermanagement (Safety and Error Management)

Hilfsebenen abstrahieren die enthaltenen Sensoren und Aktuatoren und ermöglichen eine funktionsspezifische Informationsreduktion.

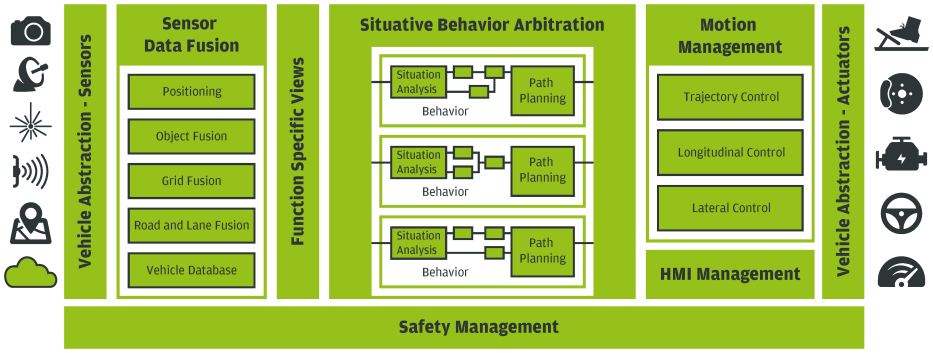


Abbildung 4.3: Funktionale Architektur für assistierende und automatisierende Fahrfunktionen der *open robinos* Spezifikation [237]. Quelle: Elektrobit, www.elektrobit.com

4.3.2 Hierarchisierung nach Information und Integrationstiefe

Die vorgestellten Konzepte zur strukturierten Beschreibung der logischen Systemarchitektur basieren auf einer kognitiven Abstraktion wie in [229] vorgestellt, nutzen die drei Ebenen der Fahraufgabe [233] oder kombinieren die beiden Ansätze [234]. Dies ermöglicht eine Charakterisierung einzelner Funktionsblöcke von an der Fahraufgabe beteiligten Features, bietet aber nicht zwangsläufig eine geeignete Unterstützung des gesamten Systems Engineering Prozesses. Das funktionale Verhalten vorhandener und etablierter E/E-Features, die die Fahraufgabe nicht direkt beeinflussen, wird durch die beschriebenen Architekturdarstellungen größtenteils vernachlässigt. Ein ganzheitlicher Ansatz sollte diese inklusive ihrer Einordnung in spezifische Fahrzeugdomänen berücksichtigen und gleichzeitig eine Abstraktion des funktionalen Systemverhaltens bieten, die eine Zuordnung von angepassten Systems Engineering Maßnahmen ermöglicht.

Abbildung 4.4 zeigt ein in dieser Dissertation entwickeltes neues Konzept zur Erweiterung der logischen Systemarchitektur durch eine ergänzende Hierarchisierung. Die enthaltenen Funktionsblöcke dienen als Beispiel und repräsentieren die in

grundlegende Funktionselemente aufgeteilten Features der in Abschnitt 4.2 eingeführten Taxonomie. Die logische Systemarchitektur erweitert den mehrschichtigen Ansatz von Stiller [229]. Die maßgeblichen Kriterien zur Charakterisierung der Funktionsblöcke sind die jeweilige Hardwareintegrationstiefe und der Charakter der verarbeiteten Informationen. Entsprechend werden sie als Bindeglieder vier Ebenen zugeordnet:

- Physikebene
- Rohinformationen
- gefilterte Informationen
- abstrahierte Informationen

Ihre Anordnung ergibt einen im Uhrzeigersinn orientierten Informationsaustausch. Die Abstraktion ermöglicht eine flexible Abbildung der funktionalen Wirkketten und -netze, der Interaktionen zwischen den Funktionsblöcken und unterstützt in Verbindung mit Ansätzen wie der *open robinos* Spezifikation [237] eine präzise Schnittstellenbeschreibung.

Auf der linken Seite der logischen Systemarchitektur findet eine nach oben gerichtete Abstraktion der realen Systemumgebung statt. Diese beinhaltet die physikalischen Prinzipien und messtechnischen Verfahren zur Erfassung der Systemumgebung sowie die Algorithmen zur Aufbereitung der Rohinformationen durch Datenfusion und Umfeldmodellierung. Dabei handelt es sich um Open-Loop Funktionen mit offenem Wirkungsweg. Auf der höchsten Ebene fließen die Informationen in die Prädiktion der Systemumgebung und das Verständnis der Szene ein, auf deren Basis die Fahrzeugbewegung und Fahrzeugkonditionierung geplant wird. Diese Planung wird auf der rechten Seite der Architekturdarstellung stufenweise konkretisiert und realisiert. Diese Funktionen beinhalten Closed-Loop Verhalten. Analog zu den Ansätzen von Höll [231] und Flemisch [233] befinden sich grundlegende für den Betrieb des Fahrzeugs notwendige Regelschleifen, z. B. zur Stabilisierung der Fahrzeugbewegung, vollständig in den unteren Hierarchieebenen. Im Folgenden werden die Ebenen und ihre zugeordneten Funktionsblöcke, ihre Eigenschaften und ihre Einordnung in das gesamtheitliche Systems Engineering im Detail diskutiert.

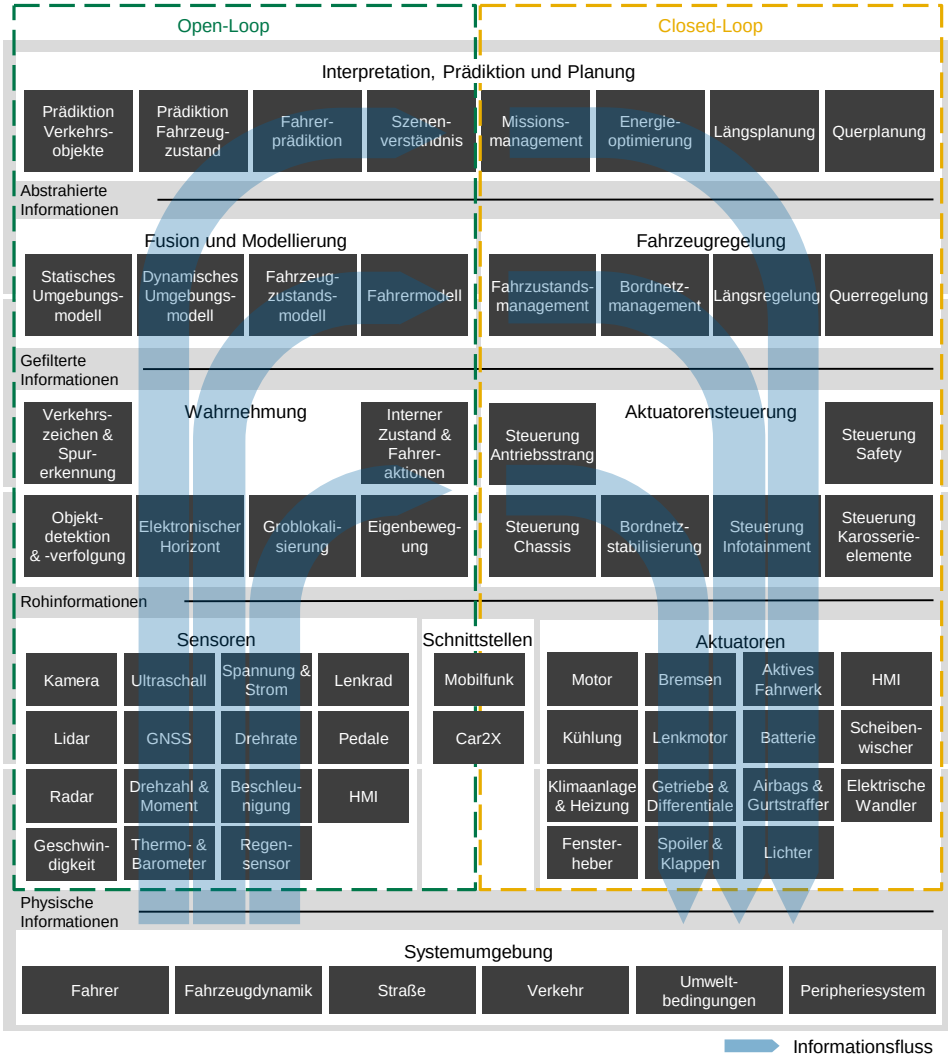


Abbildung 4.4: Ganzheitlicher Ansatz zur hierarchischen Beschreibung der logischen Systemarchitektur aller E/E-Features eines Fahrzeugs

Ebene 1 - Sensoren, Schnittstellen und Aktuatoren

Die erste Ebene abstrahiert das funktionale Verhalten der Sensoren, Kommunikationsschnittstellen und Aktuatoren, die über physikalische Prinzipien Rohinformationen gewinnen und umgekehrt entsprechend der Rohinformationen auf die Physik wirken. Sie beinhaltet die bidirektionale Umwandlung zwischen analogen Signalen und digitalen Datenströmen.

Die Funktionsblöcke der Sensorgruppe nutzen physikalische Messprinzipien und stellen die grundlegende Eigen- und Umgebungswahrnehmung zur Verfügung. Die Eigenwahrnehmung besteht aus in der Regel diskret abgetasteten, physikalischen Größen, wie Drehzahlen, Drehmomenten, Spannungen, Strömen, Beschleunigungen und Geschwindigkeiten, sowie Empfängern für ein Globales Navigationssatellitensystem (Global Navigation Satellite System, GNSS) zur Eigenlokalisierung. Die in den Sensoren zur Umgebungswahrnehmung gewonnene Rohinformation reicht von den diskret abgetasteten Radar-, Lidar- und Ultraschallreflexionen zu in Kameras aufgenommenen Einzelbildern. Die Fahrerschnittstellen zur Steuerung und Bedienung des Fahrzeugs sind in der Sensorgruppe durch die Funktionsblöcke Lenkrad, Pedale und HMI repräsentiert.

Die Schnittstellengruppe beinhaltet die grundlegenden Funktionsblöcke mit bidirektionalem Informationsfluss zur Interaktion mit technischen Systemen der Umgebung. In der Beispielsystemarchitektur ist die Schnittstellengruppe durch die Kommunikationseinheiten für Mobilfunk und Car2X-Kommunikation vertreten.

Die Gruppe der Aktuatoren umfasst alle Funktionselemente, die den Fahrzeugzustand und seine Umgebung im Sinne eines physikalischen Systems beeinflussen. Die Funktionsblöcke Motor, Getriebe und Differentiale des Antriebsstrangs beeinflussen den Vortrieb und die Energieflüsse des Fahrzeugs. Der Lenkmotor legt an die Lenksäule ein Moment an und beeinflusst damit die Querbewegung des Fahrzeugs sowie die Lenkbewegung des Fahrers. Die Bremsen beeinflussen sowohl die Längs als auch die Querbewegung des Fahrzeugs. Das aktive Fahrwerk, Spoiler, Klappen und Flügel verändern die Eigenschaften der Aero- und Fahrdynamik. Die Funktionsblöcke Batterie und elektrische Wandler repräsentieren die physikalische Funktionalität des Bordnetzes und die Safety-Domäne wird durch die passiven Sicherheitselemente Airbags und Gurtstraffer vertreten. Die weiteren Funktionsblöcke erfüllen unterstützende Aufgaben (Motorkühlung, Licht und Scheibenwischer), dienen dem Fahrkomfort (Klimaanlage, Heizung und Fensterheber)

und der Fahrerbeeinflussung, -information und -unterhaltung (HMI).

Die Funktionsblöcke dieser Ebene repräsentieren die funktionalen Inhalte von mechanischen und elektronischen Komponenten. Diese Ebene beinhaltet keine Softwareanteile. Die Funktionsblöcke repräsentieren die funktionale Schnittstelle zur eingebetteten Software. Sie unterstützen die Zuordnung von Funktionsinhalten höherer Abstraktionsebenen auf die Softwarearchitektur und die technische Systemarchitektur.

Die Systems Engineering Prozesse und Methoden der ersten Ebene sind auf die jeweiligen spezialisierten Fachgebiete der Fahrzeug- und Elektrotechnik zugeschnitten. Die Integration und Synchronisation dieser Teilgebiete wird durch das in der Automotive SPICE (vergl. Abschnitt 2.7) enthaltene „Plug-in"-Konzept abgedeckt und hier nicht näher betrachtet.

Ebene 2 - Wahrnehmung und Aktuatorensteuerung

Diese Ebene enthält die funktionalen Elemente zur Filterung und Verarbeitung der digitalen Rohsignale und zur Ansteuerung der Aktuatoren. Die enthaltenen Funktionsblöcke repräsentieren die diskreten Algorithmen zur Aufbereitung der Information und zur Realisierung des Systemverhaltens.

Die Rohinformationen stellen ein Zwischenprodukt des verwendeten Messprinzips dar. Um interpretierbare, physikalische Größen zu extrahieren, müssen sie durch die Funktionsblöcke der Wahrnehmungsgruppe gefiltert und verarbeitet werden. Beispielsweise muss das mit einem Hall-Sensor gemessene Rechtecksignal eines Drehzahlmessers entsprechend der Anzahl der Polpaare in eine Winkelgeschwindigkeit umgesetzt werden [16]. Die ausgewählten Beispiele dieser Gruppe beinhalten Funktionsblöcke zur Messung der Eigenbewegung (Geschwindigkeiten, Beschleunigungen, Drehraten) und zur Erfassung von Fahreraktionen und interner Zustände. Die Groblokalisierung repräsentiert die Ermittlung der Fahrzeugposition auf Basis der gemessenen Pseudostrecken[10] des GNSS-Empfängers. Der Funktionsblock Elektronischer Horizont nutzt die Groblokalisierung und liefert Informationen über das bevorstehende Straßensegment [239, 240]. Diese stammen aus einem internen Speicher oder werden über die Kommunikationsschnittstellen von einem

[10] Aus der Signallaufzeit ermittelter Abstand zwischen Empfänger und Satellit, weiterführende Information siehe [238]

internetbasierten Dienst abgerufen. Die Funktionsblöcke zur Verkehrszeichenerkennung, Spur- und Objektverfolgung beinhalten die grundlegenden Radar-, Lidar- und Bildverarbeitungsalgorithmen zur Extraktion von Objekten und statistischen Belegungskarten (engl. occupancy grid) [228, 225, 241, 242, 243].

Die Gruppe Aktuatorensteuerung beinhaltet Funktionsblöcke zur Steuerung der mechatronischen Aktuatoren. Sie realisieren für den Betrieb der integrierten Fahrzeugfeatures wesentliche Funktionalität. Der Funktionsblock Steuerung Antriebsstrang schließt beispielsweise die Zündwinkelsteuerung des Motors und die Kupplungssteuerung eines Automatikgetriebes ein. Die Regelfunktionen der Fahrwerkdomäne, wie ESC und ABS, sind im Block Steuerung Chassis enthalten. Die übrigen E/E-Domänen sind analog dazu durch Funktionsblöcke repräsentiert. Alle Elemente dieser Ebene implizieren Schnittstellen zur Sollwertvorgabe durch übergeordnete Funktionen.

Während die Elemente der ersten Ebene den funktionalen Anteil mechanischer und elektronischer Komponenten repräsentieren, enthält die zweite Ebene den funktionalen Teil der eingebetteten Software, die diesen Komponenten zugeordnet ist. Grundlegende Regelschleifen für den Betrieb und die Stabilisierung des Fahrzeugs sind auf diese Ebene begrenzt. Die auf dieser Ebene gewonnene Information wird innerhalb der Domäne des Fahrzeug-Kommunikationsnetzes botschaftsbasiert geteilt.

Die in Kapitel 2 eingeführten Systems Engineering Prozesse und die in Kapitel 3 eingeführten Methoden sind auf die Funktionen dieser Ebene zugeschnitten und eng an die Prozesse der Fahrzeug- und Elektrotechnik gekoppelt. Diese enge Synchronisation ist notwendig, da die Funktionsblöcke der ersten und zweiten Ebene zusammengefasst eingebettete Systeme bilden und beidseitig voneinander abhängig sind. Aufgrund ihrer Hardwarenähe und der damit zusammenhängenden Kritikalität unterliegen die Software-Implementierungen dieser Funktionsblöcke harten Echtzeitbedingungen und werden häufig auf dedizierte ECUs partitioniert. Daher erfordern durchzuführende Änderungen dieser Funktionen nach SOP einen Werkstattaufenthalt zum Aufspielen (Flashen) neuer Firmware. Die Entwicklung auf dieser Ebene wird häufig durch Tier-1-Lieferanten durchgeführt. Die Verifikation und Validierung der einzelnen Funktionselemente kann unabhängig von benachbarten Funktionselementen erfolgen, da keine oder nur geringe Interaktion stattfindet.

Ebene 3 - Fusion, Modellierung und Fahrzeugregelung

Funktionen auf der dritten Ebene fusionieren und abstrahieren die verschiedenen, unabhängigen Informationsquellen der Wahrnehmungsgruppe und übernehmen die Regelung des Fahrzeugs. In der Funktionsgruppe Fusion und Modellierung werden die gefilterten Informationen der propriozeptiven und exterozeptiven Sensoren akkumuliert und mittels modellbildender Verfahren abstrahiert. Im Fahrzeugzustandsmodell werden die unabhängigen Zustandsgrößen zueinander in Relation gesetzt. Dies ermöglicht eine verbesserte Schätzung der Eigenbewegung und -position[238] und die Ermittlung nicht messbarer interner Zustände, wie den Fahrwiderständen [244]. Im statischen Umgebungsmodell wird die geschätzte Pose[11] mit den Informationen des elektronischen Horizonts und den detektierten Spur-, Objekt- und Verkehrszeichendaten kombiniert. Dies ermöglicht eine fahrspurgenaue Lokalisierung des Fahrzeugs[245, 188]. Das dynamische Umgebungsmodell ergänzt die statische Umgebung mit den am Verkehrsgeschehen beteiligten Objekten [246, 242]. Auf diese Weise liefern die Funktionsblöcke für das statische und das dynamische Umgebungsmodell eine kompakte und in sich konsistente Darstellung der Fahrzeugumgebung. Das Fahrermodell überwacht das Verhalten des Fahrers und stellt abstrahierte Merkmale, wie Aufmerksamkeit und Fahrstil [247, 248], zur Verfügung.

Die Funktionsblöcke zur Längs- und Querregelung stellen die wichtigsten Elemente der Fahrzeugregelungsgruppe dar. Sie regeln innerhalb der spezifizierten Betriebsgrenzen die Geschwindigkeit und Pose des Fahrzeugs entsprechend vorgegebener Sollwerte und sind Bestandteil assistierender und automatisierender Features. Das Fahrzustandsmanagement koordiniert die Einzelfunktionen und ermöglicht die Auswahl unterschiedlicher Betriebsmodi (z. B. Eco, Komfort, Sport), um ein gut abgestimmtes Fahrerlebnis zu erreichen. Der Funktionsblock Bordnetzmanagment überwacht und begrenzt den Energieverbrauch der verschiedenen Komponenten und koordiniert die Energierückgewinnung elektrifizierter Fahrzeuge. Die Stellgrößen dieser Ebene stellen die Sollwertvorgaben der Steuerungs-Gruppe dar.

Die Funktionen dieser Ebene haben keine wesentliche Bedeutung für die grundlegende Funktionsfähigkeit des Fahrzeugs. Sie repräsentieren in erster Linie die funktionalen Umfänge verteilter Features. Die vorgestellten Entwicklungsprozesse und -methoden der Automobilentwicklung sind für diese Ebene geeignet, die funktionale Abhängigkeit von tieferliegenden Funktionen stellt jedoch eine Herausforderung

[11] Pose (Technik) - Bewegung und -position, Wikipedia, https://de.wikipedia.org/wiki/Pose_(Technik)

dar. So können gesamtheitlich synchronisierte Release-Meilensteine gegebenenfalls nicht geschlossen erreicht werden, wenn Basisfunktionalitäten nicht mit genügend Vorlauf zur Verfügung stehen. Dies stellt insbesondere dann ein Risiko dar, wenn diese durch Zulieferer bereitgestellt werden. Die Verifikation und Validierung dieser Funktionen erfolgt auf Schnittstellenebene. Simulationsbasierte Techniken erfordern nicht nur die Modellierung der Fahrzeugphysik und der Umgebung, sondern auch die Modellierung aller tieferliegenden funktionalen Elemente. Stehen entsprechende Funktionsmodelle nicht zur Verfügung, können Integrationstests zur Verifikation vollständiger Features erst sehr spät im Entwicklungsprozess durchgeführt werden. Die Funktionen auf dieser Ebene unterliegen nur noch weichen Echtzeitbedingungen, da sie keinen direkten Hardwarekontakt beinhalten. Die Echtzeitbedingungen leiten sich aus der durchzuführenden Regelungsaufgabe und insbesondere aus der Trägheit der Regelstrecke ab.

Ebene 4 - Interpretation, Prädiktion und Planung

Diese am stärksten von der Physik und der Hardware abstrahierte Ebene enthält die kognitiven Funktionen zur Interpretation, Prädiktion und Planung. Stochastische Prozesse ermöglichen basierend auf den Modellen des Fahrzeugzustands, der Umgebung und des Fahrers deren Vorhersage [249]. So können neben der Position und Geschwindigkeit die Energieflüsse des Fahrzeugs vorhergesagt werden. Die Prädiktion des Verhaltens von Verkehrsobjekten und des Fahrers ermöglicht eine antizipatorische Auslegung des Systemverhaltens [218]. Der Funktionsblock Szenenverständnis repräsentiert die Interpretation der aggregierten Informationen [250].

Basierend auf der interpretierten Umgebung koordiniert das Missionsmanagement die Optimierungs- und Planungsziele des Systems und übernimmt strategische und taktische Entscheidungen, wie eine Routenänderung oder die Durchführung eines Spurwechsels [251]. Der Funktionsblock Energieoptimierung stellt ein dediziertes Element zur energetischen Betrachtung und Beeinflussung des Fahrzeugs dar. Das Element abstrahiert funktionale Aspekte zur Reichweitenoptimierung und Rekuperationssteuerung und deren Auswirkung auf die Bewegungsplanung [218]. Entsprechend der abstrahierten und interpretierten Informationen und der Vorgaben durch Missionsmanagement und Energieoptiomierung wird die Fahrzeugbewegung geplant [252, 253, 254]. Der Funktionsblock Längsplanung berechnet eine Geschwindigkeitstrajektorie, die der Längsregelung als Sollwertvorgabe dient. Die Querplanung errechnet die Bahn des Fahrzeugs als Sollwertvorgabe der Quer-

regelung.

Die etablierten Kommunikationsnetze sind nicht für den Informationsumfang auf dieser Ebene ausgelegt. Sie bieten eine hohe Zuverlässigkeit und Robustheit, jedoch nur eine begrenzte Kapazität. Aus diesem Grund eignen sich die Funktionsblöcke dieser Ebene insbesondere für eine Implementierung auf einer zentralen, leistungsfähigen Steuereinheit [255, 235, 236].

Die weitere Komplexitätssteigerung und Zunahme an Abhängigkeiten verdeutlicht die auf Ebene der gefilterten Informationen eingeführten Herausforderungen (Release Synchronisierung & komplexe Modellbildung funktionaler Elemente) für etablierte Systems Engineering Ansätze.

Eine schrittweise Integration der Funktionen würde gestaffelte Release-Konzepte erfordern und die Gesamtentwicklungszeit verlängern oder eine verkürzte Entwicklungszeit der Basisfunktionen voraussetzen. Zentralisierte Ansätze in Verbindung mit Over-the-Air-Updates [256] bieten hier eine vielversprechende Alternative. Eine Etablierung von Continuous Delivery Prozessen (s. Abschitt 3.3.5) für Funktionen dieser Ebene stellt eine Chance dar, die Integration und Freigabe nach SOP sowie eine kontinuierliche Verbesserung des Fahrerlebnisses zu ermöglichen.

Auf der vierten Ebene eignen sich aufgrund der starken Hardwareabstraktion in der Softwaretechnik etablierte Methoden zur Verifikation und Validierung besser als die Hardware-fokussierten Methoden eingebetteter Systeme. Anstelle detaillierter physikalischer Mehrkörpermodelle der Fahrzeugmechanik und der physikalischen Grundlagen der Sensoren ist eine vereinfachte Modellierung des funktionalen Verhaltens für die Anwendung simulationsbasierter Ansätze ausreichend. Dies beinhaltet insbesondere die Nachbildung der Umgebungsmodellierung und des Funktionsverhaltens der tieferliegenden Schichten.

4.3.3 Darstellung ausgewählter Features innerhalb der vorgeschlagenen logischen Systemarchitektur

Die Funktionselemente der exemplarischen logischen Systemarchitektur in Abbildung 4.4 wurden aus der Analyse der ausgewählten und charakterisierten Fahrzeug-Features abgeleitet. Im folgenden wird durch eine konzeptuelle Modellierung des Funktionsverhaltens ausgewählter Merkmale aller drei Hauptkategorien der Ta-

xonomie die Anwendbarkeit der Hierarchisierung demonstriert. Die Modellierung von etablierten Features verdeutlicht die Fähigkeit des Ansatzes, existierende Features und ihre Struktur abzubilden. Gleichzeitig wird mit der Modellierung von Forschungskonzepten bewiesen, dass der Ansatz auch zukünftigen Anforderungen gewachsen ist.

Um sowohl Features der Längs als auch der Querführung darzustellen werden aus der Klasse der integrierten Features die Elektronische Stabilitätskontrolle (Electronic Stability Control, ESC) und die Elektrisch Angetriebene Servolenkung (Electric Power Assisted Steering, EPS) in der hierarchisierten logischen Architektur abgebildet. Auf diesen bauen die verteilten Features zur Längsregelung mittels Abstandsregeltempomat (Adaptive Cruise Control, ACC) und zur Querregelung mittels Spurhalteassistent (Lane Keep Assist, LKA) auf. Auch sie werden in der hierarchisierten Architektur modelliert. Aus der Gruppe der quervernetzten Features wird die logische Systemarchitektur des Staupilot beispielhaft abgebildet, der die Längs- und Querregelung kombiniert und als weltweit erstes Serienfeature für hochautomatisiertes Fahren angekündigt ist [224].

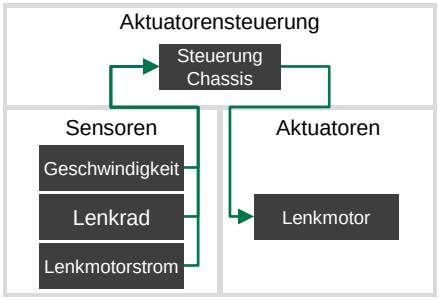
Modellierung EPS

Die EPS stellt einen Aktuator zur Beeinflussung der Querbewegung des Fahrzeugs dar. Lenkachse und Lenkrad werden mittels eines Elektromotors mit einem Drehmoment beaufschlagt, um die Lenkbewegung des Fahrers zu unterstützen oder um eine vorgegebene Sollposition der Lenkung zu erreichen. Kim [174] beschreibt eine Servolenkung zur Unterstützung der beabsichtigten Lenkbewegung des Fahrers. Die Veröffentlichung liefert eine detaillierte Beschreibung der Systemarchitektur einer EPS. Das vom Fahrer ausgeübte Lenkmoment wird gemessen. In Abhängigkeit von der Fahrzeuggeschwindigkeit berechnet sich daraus der Sollwert des elektrischen Lenkmoments. Mittels pulsweitenmoduliertem Motorstrom wird entsprechend der Sollwertvorgabe durch den Fahrer das unterstützende Moment eingeregelt.

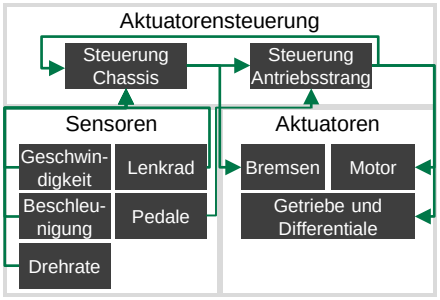
Naranjo [175] beschreibt den Einsatz einer EPS als innerer Regelkreis einer kaskadierten Reglerstruktur für die automatisierte Querführung eines Fahrzeugs. Anstelle der Lenkbewegung des Fahrers dient der Stellgrößenausgang der umgebenden Regelschleife als Sollwertvorgabe der EPS. In der äußeren Schleife regelt ein Fuzzy-Regler das Fahrzeug auf einer vorgegebenen Zieltrajektorie. Während für den Betrieb der äußeren Regelschleife eine Frequenz von 10 Hz ausreichend ist, setzt die EPS in der inneren Schleife eine Betriebsfrequenz von 100 Hz voraus. Sie stellt damit

strengere Echtzeitanforderungen an die eingesetzte Steuerungselektronik als der sie umschließende Regelkreis zur Trajektorienregelung.

Die aus diesen beiden Arbeiten abgeleitete logische Systemarchitektur des EPS-Features ist in Abbildung 4.5a dargestellt. Die Regelung des Lenkmotors wird dem Funktionsblock Steuerung Chassis zugeordnet. Die Fahrzeuggeschwindigkeit und das am Lenkrad gemessene Lenkmoment dienen der Regelung als Führungsgröße. In Verbindung mit dem gemessenen Motorstrom als Rückführungsgröße wird das Drehmoment des Lenkmotors geregelt.



(a) Elektrisch Angetriebene Servolenkung (Electric Power Assisted Steering, EPS)



(b) Elektronische Stabilitätskontrolle (Electronic Stability Control, ESC)

Abbildung 4.5: Modellierung integrierter Features mit Elementen der vorgeschlagenen logischen Systemarchitektur (Abb. 4.4)

Modellierung ESC

Das ESC-Feature stellt eine aktive Sicherheitstechnologie des Fahrwerks dar. Der Fahrer wird darin unterstützt, das Fahrzeug auf dem beabsichtigten Weg zu halten und folglich Unfälle abzuwenden [172]. Das Grundprinzip der ESC ist die Stabilisierung der Gierbewegung des Fahrzeugs durch die individuelle Steuerung des Reifenschlupfs jeden Rads. Um eine entgegen der Fahrzeugführung des Fahrers gerichtete Wirkung zu verhindern, muss die elektronische Stabilitätskontrolle eine präzise Erfassung der Fahrerabsichten beinhalten [171]. Deren zutreffende Interpretation ermöglicht erweiterte fahrdynamische Fähigkeiten zur Richtungssteuerung.

Tseng [171] beschreibt die Entwicklung und Verbesserung eines ESC-Systems bei Ford. Ziel des Systems ist die Regelung der Gierrate entsprechend der Fahrervorgabe sowie die Minimierung des Fahrzeugschwimmwinkels durch eine Anpassung

des Antriebsmoments, des Bremsdrucks und der Bremsdruckverteilung. Die beschriebene Funktion berechnet die durch den Fahrer beabsichtigte Gierrate mittels Einspurmodell[12] aus dem erfassten Lenkwinkel. Während die reale Gierrate mittels Sensoren direkt erfasst wird, schätzt das System den aktuellen Schwimmwinkel aus den gemessenen Beschleunigungen, Drehraten und der Fahrzeuggeschwindigkeit. Um dem Fahrer bei Erreichen der fahrdynamischen Grenzen des Fahrzeugs eine haptische Rückmeldung zu liefern, werden die Gierrate und der Schwimmwinkel lediglich an die erwarteten Sollwerte angenähert. Diese Abweichung ist im durch Tseng beschriebenen System so gewählt, dass sie die Fähigkeiten eines durchschnittlichen Fahrers zur fahrdynamischen Regelung nicht übersteigt. Liebermann [172] stellt die Integration eines ESC-Systems in einen Allrad getriebenen Antriebsstrang vor. Die Stellgrößen der ESC werden durch die elektronisch steuerbaren Kupplungen des zentralen Differentialgetriebes ergänzt. Sie ermöglichen eine variable Lastverteilung zwischen Vorder- und Hinterachse.

Eine auf den vorgestellten Systemen basierende beispielhafte logische Systemarchitektur des ESC-Features ist in Abbildung 4.5b dargestellt. Sie enthält die Sensoren zur Erfassung des Fahrzeugzustandes und der Fahrerabsicht. Die Regelung des Features wird über den Block „Steuerung Chassis“ abgebildet. Die Ansteuerung der Bremsen erfolgt direkt, der Motor, das Getriebe und die Differenziale werden über die Steuerung des Antriebsstrangs beeinflusst.

Modellierung ACC

Der ACC übernimmt als verteiltes Feature der Assistenzgruppe die Regelung der Fahrzeuglängsgeschwindigkeit. Als Führungsgröße der Regelung dient eine durch den Fahrer vorgegebene Zielgeschwindigkeit, die an die Geschwindigkeit vorausfahrender Verkehrsobjekte angepasst wird. Einen umfassenden Überblick über die Entwicklung und die Funktionsweise von ACC-Systemen verschiedener OEMs bietet Winner [206].

Grundlage der adaptiven Geschwindigkeitsregelung ist die Erfassung des relevanten Zielfahrzeugs vor dem Fahrzeug. Die Objekte im Frontbereich des Fahrzeugs werden mittels Radarsensor erfasst. Als Zielfahrzeug wird das nächstgelegene Objekt im prädizierten Fahrschlauch des Fahrzeugs ausgewählt. Statische und entgegenkommende Objekte werden bei der Zielauswahl ignoriert. Die Prädiktion des Fahrschlauchs

[12] Modell zur vereinfachten Abbildung der Fahrdynamik, weiterführende Informationen in [257]

erfolgt gemäß der Lenkbewegung des Fahrers. Sind zusätzliche Informationen über den Verlauf der Fahrspur vorhanden, können diese die Präzision und Robustheit der Zielauswahl verbessern. Die Regelung der Fahrzeuggeschwindigkeit in Folgefahrt basiert auf der um die Sollzeitlücke verzögerten Geschwindigkeit und Beschleunigung des Zielfahrzeugs. Die Sollzeitlücke definiert dabei den geschwindigkeitsabhängigen Abstand zum Vorderfahrzeug. Als Aktuatoren dienen der Regelung der Antriebsstrang und das Bremssystem des Fahrzeugs. Moon [258] beschreibt einen zweistufigen Kaskadenregler, dessen äußere Regelschleife die Fahrzeuggeschwindigkeit mittels Beschleunigungsvorgaben regelt. Diese werden in der inneren Regelschleife über eine Regulierung des Bremsdrucks und des Antriebsmoments umgesetzt.

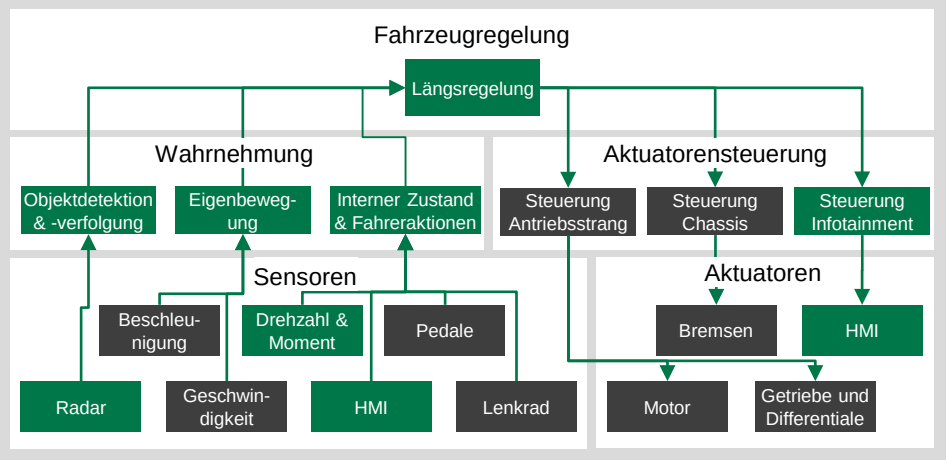


Abbildung 4.6: Modellierung eines Abstandsregeltempomats auf Ebene der logischen Systemarchitektur. Grüne Funktionsblöcke wurden neu hinzugefügt, graue Funktionsblöcke sind bereits in der Modellierung des ESC (Abb. 4.5b) vorhanden.

Abbildung 4.6 stellt eine konzeptuelle Modellierung der logischen Systemarchitektur eines ACC-Features dar. Die Rohinformationen der beteiligten Sensorelemente werden zunächst in der Wahrnehmungsgruppe aufbereitet. Basierend auf Informationen aus internen Sensoren wird die Eigenbewegung des Fahrzeugs geschätzt. Die gemessenen Radarreflexionen werden durch Algorithmen zur Objektverfolgung verarbeitet, um die für die Adaption der Geschwindigkeit benötigte Distanz, Relativgeschwindigkeit und -beschleunigung zu vorausfahrenden Verkehrsobjekten zu erhalten. Entsprechend dieser Informationen und der eingestellten Zeitlücke berechnet die Längsregelung für den jeweiligen Ausführungszyklus eine Zielbeschleunigung. Diese wird über die Elemente der Aktuatorensteuerung realisiert.

Modellierung Spurhalteassistent

Der Spurhalteassistent (Lane Keep Assist, LKA) assistiert dem Fahrer bei der Querführung des Fahrzeugs, ohne eine vollständige Übernahme der Fahraufgabe durchzuführen. Ishida [208] beschreibt einen Spurhalteassistenten, der aus einem Kameramodul zur Spurerkennung, einem Steuergerät für die Querregelung und der EPS zur Stellung des Lenkmoments besteht. Aus dem aufgenommenen Kamerabild extrahiert der Funktionsblock zur Spurerkennung die vorhandenen Spurmarkierungen inklusive deren Abstand, Orientierung, Krümmung und der Änderung der Krümmung. Entsprechend der gefilterten Information generiert die Querregelung ein Lenkmoment als Stellgröße. Dieses Moment wird mit Hilfe der EPS an der Lenkachse angelegt und das Fahrzeug in der Spur gehalten.

Modellierung Staupilot

Der Staupilot stellt ein Feature ab Automatisierungsstufe drei dar. Er übernimmt die Längs- und Querführung innerhalb des spezifischen Anwendungsfalls Stau und stockender Verkehr bis 60 km/h. Der Pilot ist in der Lage die eigenen Systemgrenzen zu überwachen und gegebenenfalls die Fahrzeugführung an den Fahrer zu übergeben.

Abbildung 4.7 stellt die konzueptuelle Modellierung der logischen Systemarchitektur eines Staupilot-Features dar. Zur vollständigen Erfassung des Fahrzeugumfelds wird eine große Zahl an Sensoren eingesetzt. Kamerasysteme dienen in erster Linie zunächst der Erfassung von Fahrspurmarkierungen und Verkehrsschildern. Weiter unterstützen sie die Radar-, Lidar- und Ultraschallsensor basierte Objektdetektion und -verfolgung. Die Fahrereingaben an Lenkrad, Pedalen und HMI dienen zur Überprüfung dessen Übernahmebereitschaft und werden im Fahrerzustand gesammelt. Mit Hilfe der gemessenen Drehrate, den Beschleunigungen und der Geschwindigkeit wird die Eigenbewegung des Fahrzeugs geschätzt.

Für eine sichere Führung des Fahrzeugs benötigt der Staupilot eine umfassende und konsistente Repräsentation des Fahrers, des Fahrzeugzustands und der Fahrzeugumgebung. Die aus den Signalen der unterschiedlichen Sensoren extrahierten Objekte werden im Modell der statischen Umgebung mit den erkannten Spuren und Verkehrszeichen und dem Modell der dynamischen Umgebung untereinander fusioniert. Auf Basis des statischen Umgebungsmodells, des internen Zustands und

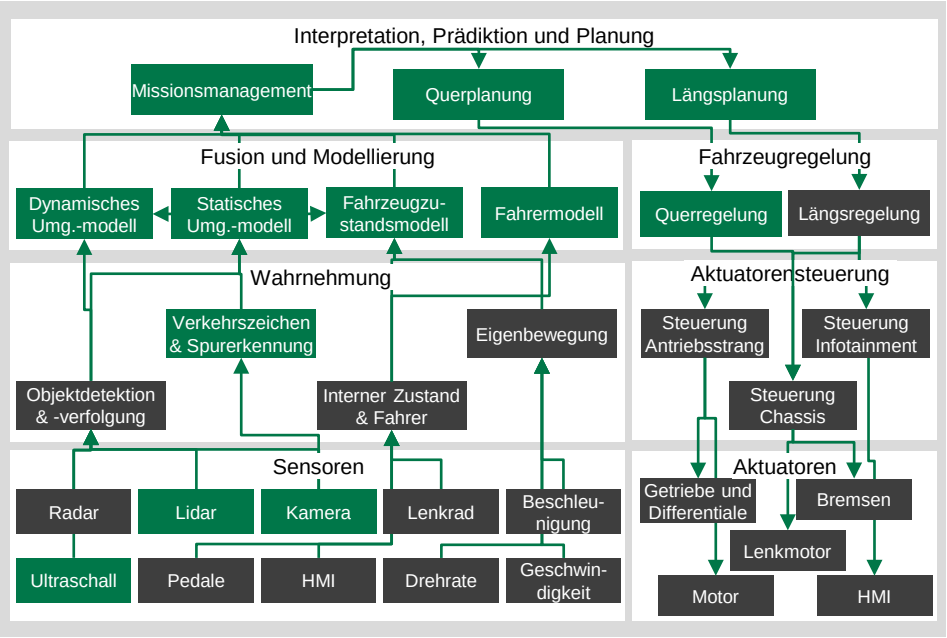


Abbildung 4.7: Modellierung eines Staupiloten auf Ebene der logischen Systemarchitektur. Grüne Funktionsblöcke wurden neu hinzugefügt, graue Funktionsblöcke sind bereits in vorhergehenden Modellierungen (Abb. 4.5a, 4.5b und 4.6) vorhanden.

der Eigenbewegung wird der Fahrzeugzustand abgebildet. Die über den Fahrer gesammelten Informationen fließen im Fahrermodell zusammen.

Das Funktionselement Missionsmanagement entscheidet auf Grundlage der Modelle über die durchzuführenden Fahrmanöver. Diese werden in den Blöcken zur Quer- und Längsplanung in eine entsprechende Bahn und Geschwindigkeitstrajektorie umgesetzt. Die Funktionselemente Quer- und Längsregelung realisieren diese geplante Bahn und Trajektorie mit Hilfe der Steuerungselemente und der Aktuatoren. Über eine derartige Wirkkette kann ein Staupilot-Feature das Fahrzeug sicher im Verkehr bewegen.

4.4 Fazit

Der in diesem Kapitel vorgestellte Überblick über aktuelle und zukünftige Kundenfunktionen ermöglicht eine ganzheitliche Betrachtung der Anforderungen an die Prozesse, Methoden und Werkzeuge des Automotive Systems Engineering (ASE). Aus der Analyse der 52 Features geht eine klare Zunahme der internen Vernetzung und Abhängigkeiten hervor. Damit gewinnt die ganzheitliche Betrachtung der Wirkketten und -netze an Bedeutung. Eine vom Gesamtsystem losgelöste Betrachtung der einzelnen Funktionselemente führt zwangsläufig zu suboptimalen Ergebnissen.

Die Abstraktion und Reduktion übertragener Informationen sollte eng mit den zuständigen Teams nachfolgender Funktionselemente abgestimmt werden. Beispielsweise reduzieren gängige GPS-Receiver die Positionsmessung entsprechend des NMEA Protokolls auf die jeweilige beste Annahme. Im Fall von Mehrwegeausbreitung oder Nicht-Sichtverbindung werden dadurch wertvolle Informationen zu zweit- oder drittbesten Positionsschätzungen reduziert. Werden diese Informationen weitergegeben, kann in Verbindung mit Sensorik zur Umfelderfassung und hochgenauen Fahrzeugmodellen eine deutlich präzisere und robustere Lokalisierung erreicht werden[259].

Für eine erfolgreiche Produktentwicklung ist insbesondere in frühen Entwurfsphasen ein detailliertes Verständnis der umgebenden Elemente notwendig. Idealerweise erhalten die Entwickler einen vollständigen Überblick bezüglich der innerhalb eines angrenzenden Funktionselements verfolgten Konzepte und eingesetzten Algorithmen. Aufgrund der zunehmenden Komplexität der Funktionselemente höherer Ebenen werden zur ausreichenden Kommunikation die sprachliche Beschreibung ergänzende Mittel benötigt. Aufgrund der in der Produktentwicklung und -produktion involvierten Lieferantenbeziehungen und dem Schutz geistigen Eigentums ist der vollumfängliche Austausch dieser Informationen in der Realität häufig nicht möglich. Auch hier kann eine intensivere Nutzung von in der Erprobung aufgezeichneten Daten einen Mehrwert bieten.

5 Neues Konzept für die Entwicklung und den Test prädiktiver Fahrzeugregelungsfunktionen

Die im vorhergehenden Kapitel analysierten Kundenfunktionen und das daraus abgeleitete konzeptuelle Modell der logischen Systemarchitektur für assistiertes und automatisiertes Fahren dient im Folgenden der Bestimmung der aktuellen und zukünftigen Herausforderungen an das Automotive Systems Engineering. Am Beispiel des in Abschnitt 5.1 vorgestellten Prädiktiver Abstandsregeltempomat (Predictive Cruise Control, PCC) werden die Anforderungen der Systementwicklung für Kraftfahrzeuge analysiert, die bestehenden Vorgehensweisen und Methoden für die Entwicklung, Verifikation und Validierung rekapituliert und offene Fragestellungen identifiziert. Ein maßgebliches Ziel ist die Integration neuer Ansätze in das bestehende und abgestimmte Entwicklungsvorgehen in der Automobilindustrie. Eine Ergänzung der bestehenden Prozesse und Methoden wird deren Austausch vorgezogen. Der Fokus der Analyse liegt auf der Entwicklung des Features. Querschnittthemen, wie der Entwurf der übergeordneten Systemarchitektur (vergleiche Abb. 4.4) oder der Vernetzung, stehen nicht im Fokus und werden primär bezüglich der Schnittstellen zur Entwicklung des PCC-Features betrachtet.

5.1 Der prädiktive Abstandsregeltempomat - Entwicklungsprojekt ACC InnoDrive

Radtke [260] und Wahl [218] beschreiben das im Entwicklungsprojekt ACC InnoDrive entwickelte PCC-Feature als ein Fahrerassistenzsystem für die energie- und zeitoptimale Längsregelung des Fahrzeugs. Neben der Adaption der Geschwindigkeit an führende Verkehrsobjekte, wird die Geschwindigkeit bei der um die Prädiktion erweiterten Variante des ACC auch an die Straßentopologie und die geltenden Geschwindigkeitsbeschränkungen angepasst.

Der durch Wahl beschriebene Ansatz implementiert eine optimale Modellprädiktive Regelung (Model Predictive Control, MPC). Für die Regelung der Fahrzeuglängsgeschwindigkeit steht ein von der Fahrstrecke abhängiger Geschwindigkeitsbereich zur Verfügung. Dieser Bereich wird durch die geltenden gesetzlichen Geschwindigkeitslimits und eine maximal zulässige Querbeschleunigung beim Durchfahren von Kurven nach oben begrenzt. Basierend auf einem Modell des dynamischen Verhaltens des Fahrzeugs und Informationen über die Geschwindigkeitsbeschränkungen und Topologie der vorausliegenden Fahrstrecke, sogenannte prädiktive Streckendaten, wird in diesem Geschwindigkeitsbereich mittels eines definierten Gütemaßes eine optimale Trajektorie geplant und geregelt. Abbildung 5.1 stellt die Geschwindigkeitsbeschränkung aufgrund gesetzlicher Tempolimits und Kurvenradien sowie die durch das PCC-Feature realisierte Geschwindigkeit auf einem drei Kilometer langen Streckenabschnitt dar.

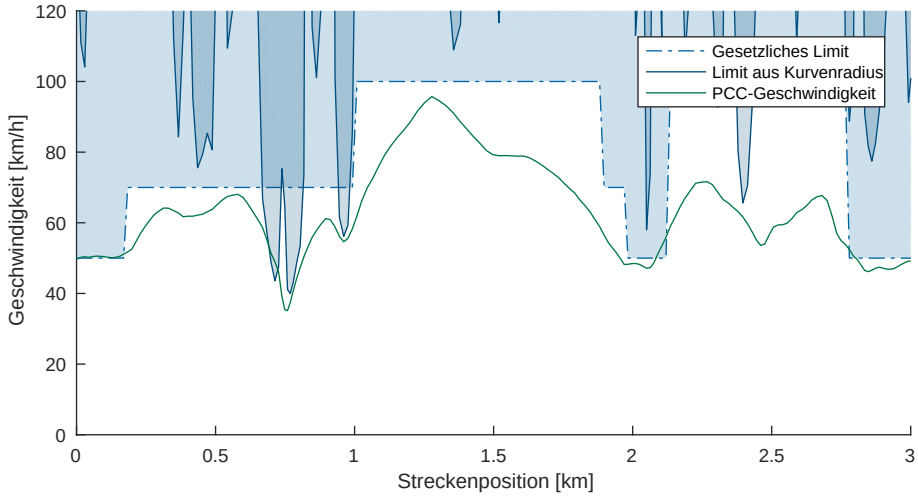


Abbildung 5.1: Begrenzung des möglichen Geschwindigkeitsbereichs durch gesetzliche Geschwindigkeitslimits und Kurvenradien sowie die durch das PCC-Feature realisierte Fahrzeuggeschwindigkeit.

Das in Abbildung 5.2 dargestellte logische Architekturkonzept stellt die hierarchisierte Abstraktion des beschriebenen Ansatzes dar. Die Umfelderfassung des ACC-Features (vergl. Abschnitt 4.3.3) wird für das PCC-Feature um ein Kamerasystem und die zugehörige Bildverarbeitung zur Spurerkennung und Verkehrszeichenerfassung erweitert. Der im Fahrzeug verbaute GNSS Empfänger liefert eine Groblokalisierung, auf deren Basis das Funktionselement Elektronischer Horizont die Topologie und die zulässige Höchstgeschwindigkeit des vorausliegenden

Streckenabschnitts bereitstellt. Aus den mit fahrzeuginternen Sensoren ermittelten Werten für die Geschwindigkeit und Beschleunigungen wird die Eigenbewegung des Fahrzeugs geschätzt. Aktionen des Fahrers an Lenkrad, Pedalen und dem HMI sowie gemessene Drehzahlen und Momente definieren den internen Zustand des Systems. Sie werden auf der Interpretationsebene im Fahrzeugzustandsmodell mit der geschätzten Eigenbewegung des Fahrzeugs zusammengefasst. Das Fahrzeugzustandsmodell stellt im Zusammenspiel mit den Modellen der statischen und dynamischen Umgebung eine konsistente Repräsentation des Systems und der Systemumgebung dar.

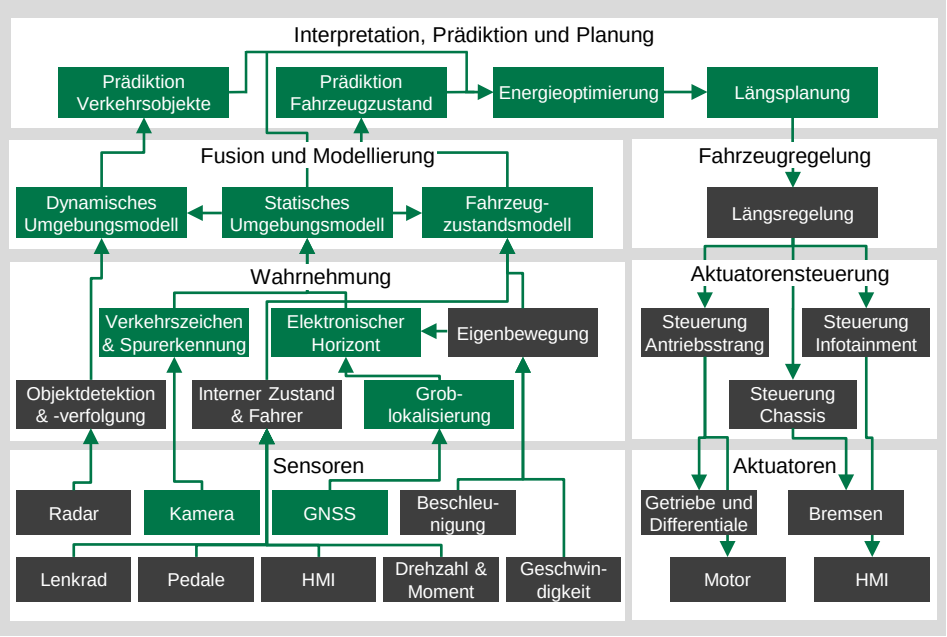


Abbildung 5.2: Modell der logischen Systemarchitektur der Funktionswirkkette eines prädiktiven Abstandsregeltempomats. Die grünen Funktionsblöcke ergänzen die grauen Funktionsblöcke, die bereits in der Modellierung des ACC (Abb. 4.6) vorhanden sind.

Bauer [219] beschreibt einen zweischichtigen Ansatz für die MPC der Fahrzeuglängsbewegung. Die in Abbildung 5.2 gewählte konzeptuelle Darstellung fasst die Ansätze von Wahl und Bauer in funktional abstrahierten Blöcken zusammen. Basierend auf der System- und Umgebungsmodellierung werden innerhalb der MPC das System beeinflussende Verkehrsobjekte und mögliche Zustandsvektoren des Egofahrzeugs prädiziert. Die möglichen prädizierten Zustandsvektoren werden mit einer Kostenfunktion entsprechend des erwünschten Fahrzeugverhaltens bezüg-

lich Energieverbrauch und Fahrkomfort bewertet. Die Längsplanung repräsentiert die Auswahl der energie-, zeit-, und komfortoptimalen Realisierung als geplante Längsführungsstrategie. In der von Bauer beschriebenen zweischichtigen Regelungsarchitektur dient diese Strategie einer zweiten nachgeschalteten MPC-Schicht als Führungsgröße. Der Vorteil der beschriebenen Aufteilung in zwei Schichten liegt in der ressourcenschonenden Realisierung der MPC ohne die Regelungsgüte durch Vereinfachungen oder eine Erhöhung der Zykluszeit negativ zu beeinflussen [219].

Als Stellglieder der MPC dienen die nachgelagerten Elemente der Aktuatorensteuerung und die zugehörigen Aktuatoren. Die Steuerung des Antriebsstrangs setzt angeforderte Beschleunigungen in ein entsprechendes Motormoment und im Getriebe die vorgegebene Schaltstrategie um. Verzögerungen, die über das Schleppmoment des Motors hinausgehen, werden über die im ESC-Steuergerät implementierte Chassissteuerung als Bremsmoment realisiert.

5.2 Abgeleitete Herausforderungen

Die in dieser Arbeit für die Entwicklung prädiktiver Fahrzeugregelungsfunktionen adressierten Herausforderungen an die Methoden und Prozesse des ASE sind in Abbildung 5.3 abstrahiert dargestellt. Die Einführung hochentwickelter Sensorsysteme erweitert den Wahrnehmungsbereich der Fahrzeugfunktionen und verschiebt damit die Systemgrenzen aktueller Entwicklungen zunehmend über die physische Grenze der Karosserie hinaus. Funktionen stellen Informationen digitaler Karten in Form eines elektronischen Horizonts zur Verfügung und erweitern damit die Umfelderfassung über das Sichtfeld des Fahrers hinaus und ermöglichen ein vorausschauendes Regelungsverhalten. Die Verfügbarkeit zuverlässiger Datenverbindungen verstärkt diesen Trend durch die Nutzung serviceorientierter Funktionen (vergl. Abschnitt 3.3.2), die beispielsweise Informationen zum aktuellen Verkehrsfluss auf der geplanten Route bereitstellen. Diese Technologien ermöglichen einerseits die Einführung innovativer Anwendungen, erhöhen andererseits aber die im Fahrzeug verfügbaren und zu verarbeitenden Informationstypen und -umfänge. Diese Zunahme bedingt eine erhebliche Steigerung der benötigten Aufwände für Spezifikation, Modellbildung, Test und Simulation.

Neben der Zunahme an Signalen und Sensoren steigt die Gesamtzahl verschiedener Kundenfunktionen im Fahrzeug. Viele der neu eingeführten Features kommunizieren untereinander. Die in Abschnitt 4.3 vorgestellte logische Systemarchitektur

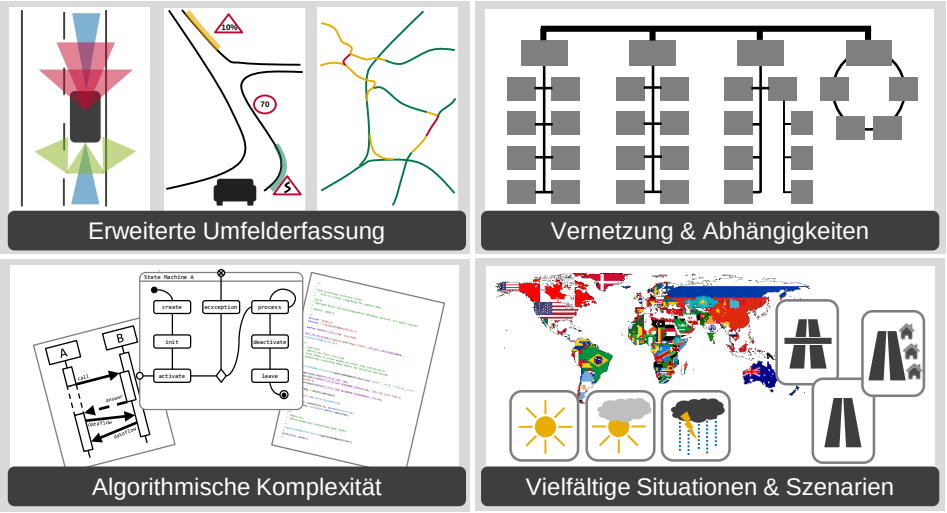


Abbildung 5.3: Aktuelle Herausforderungen an die Prozesse und Methoden des Automotive Systems Engineering

verdeutlicht die zunehmende Vernetzung einzelner E/E-Funktionen im Fahrzeug und die daraus resultierenden Abhängigkeiten. Die innovativen Funktionselemente des beschriebenen PCC liegen in den höheren Abstraktionsebenen und sind von den sensorischen und ausführenden Fähigkeiten der unteren Ebenen abhängig. Diese enge Vernetzung und hohe Abhängigkeit einzelner Funktionselemente erfordert eine aufwändige Koordination der unterschiedlichen Tätigkeiten und zuständigen Entwicklungsgruppen, -abteilungen und -unternehmen. Die Komplexität der Funktionen und ihrer zugehörigen Schnittstellen erfordert eine Kooperation und Kommunikation, die über den Austausch von Spezifikationen und konzeptuellen Modellen hinausgeht. Dem entgegen steht das Interesse aller Beteiligten ihr geistiges Eigentum angemessen zu schützen.

Der in Abschnitt 5.1 beschriebene zweischichtige MPC-Ansatz für dir prädiktive Längsregelung steht beispielhaft für die dritte große Herausforderung, der steigenden algorithmischen Komplexität der entwickelten Funktionen. Eine hohe Verfügbarkeit performanter Hardware ermöglicht die Realisierung zunehmend komplexer, aber auch leistungsstärkerer Funktionskonzepte. Sie umfassen nicht nur neue Anwendungen, sondern auch gesteigerte Fähigkeiten bestehender Anwendungen. Diese reagieren mit einer erhöhten Sensibilität auf detaillierte Änderungen ihrer Eingangsgrößen und decken damit ein größeres Funktionsspektrum ab. Die Komplexitäts- und Leistungszunahme erschwert die präzise Spezifikation der Zielsetzung und

Anforderungen der Funktion, da ein höherer Detailgrad erforderlich wird. Der Einsatz disruptiver Technologien, wie beispielsweise die prädiktiven Streckendaten des elektronischen Horizonts, verstärkt diesen Effekt. Vorhandene Erfahrung im Umgang mit diesen Technologien ist begrenzt und muss häufig zunächst in der praktischen Anwendung aufgebaut werden, bevor eine Spezifikation des erwarteten Verhaltens möglich ist.

Die erweiterte Umfelderfassung (Abb. 5.3 links oben) und zunehmende Vernetzung (Abb. 5.3 rechts oben) führen zu expandierenden Systemgrenzen und erweitern damit den Zustandsraum des Fahrzeugs im Sinne eines Systems. Zusätzlich steigt die Menge an zu verarbeitender Information aufgrund der steigenden algorithmischen Komplexität (Abb. 5.3 links unten) und der einhergehenden feineren und detaillierteren Abtastung des Zustandsraums. Daraus lässt sich als vierte große Herausforderung eine Zunahme der für die Verifikation und Validierung abzudeckenden Situationen und Szenarien folgern (Abb. 5.3 rechts unten). Eine Verfügbarkeit der Funktionen in weltweit verteilten Märkten verschärft diese Herausforderung aufgrund unterschiedlicher externer Rahmenbedingungen und Regularien und eine möglichst dauerhafte Betriebsbereitschaft erfordert eine Prüfung unter unterschiedlichsten Wetter und Klimabedingungen. Bei der Entwicklung des PCC müssen neben den unterschiedlichen Verkehrsregeln der Zielmärkte beispielsweise auch regional unterschiedliche Verfahren zur Trassierung des Straßenverlaufs sowie abweichende Kategorisierungen der Straßenklassen berücksichtigt werden.

5.3 Anforderungen und Rahmenbedingungen der Entwicklung

Für eine erfolgreiche Entwicklung müssen die eingesetzten Prozesse und Methoden befähigt sein, in einem begrenzten Zeitraum und mit begrenztem Ressourcenaufwand die in Abschnitt 3.1 vorgestellten Qualitätskriterien im entwickelten Produkt umzusetzen und deren Erfüllung nachzuweisen. Aus diesen Voraussetzungen können eine Reihe von Anforderungen an das Automotive Systems Engineering abgeleitet werden. Im Folgenden werden allgemeingültige und anwendungsspezifische Anforderungen und Rahmenbedingungen in Bezug auf die Entwicklung des in Abschnitt 5.1 vorgestellten PCC-Features aufgestellt. Der Fokus dieser Arbeit liegt auf der Untersuchung der Qualitätskriterien Leistungsfähigkeit und Zuverlässigkeit sowie der Nutzungsqualität (s. Abschnitt 3.1) in Bezug auf das funktionale Verhalten

des Features.

5.3.1 Allgemeingültige Anforderungen

Nachverfolgbarkeit - Zu Beginn eines Entwicklungsprojekts werden entsprechend Automotive SPICE (Abschnitt 2.7) die Anforderungen an das Produkt erhoben. Um die Umsetzung und Erfüllung der spezifizierten Anforderungen systematisch und vollständig nachvollziehbar und überprüfbar zu gestalten, muss für alle Arbeitsprodukte die Rückverfolgbarkeit auf spezifizierte Anforderungen gewährleistet sein.

Reproduzierbarkeit - Alle Ergebnisse der durchgeführten Maßnahmen zur Verifikation und Validierung müssen reproduzierbar sein. Die Forderung nach Reproduzierbarkeit schließt einerseits die Verwendung zufälliger Ergebnisse aus, andererseits erfordert jede Änderung eine wiederholte Qualitätsprüfung aller von der Änderung betroffenen Arbeitsprodukte um deren Zuverlässigkeit nachzuweisen.

Regelmäßige Bewertung - Um während der Entwicklung auftretende Risiken zu minimieren und eine Steuerung des Entwicklungsprojekts zu gewährleisten, muss der jeweilige Reifegrad des Systems und die Erfüllung der Qualitätsziele in regelmäßigen Abständen erfasst werden.

Hinreichende Abdeckung - Eine statistisch signifikante Abdeckung dient dem Nachweis der Allgemeingültigkeit. Die durchgeführten Bewertungen müssen daher eine relevante Untermenge der in der Realität auftretenden Situationen und Szenarien abdecken. Die angewandten Methoden sollen die für Verifikation und Validierung verantwortlichen Entwickler bei der Ermittlung der Abdeckung unterstützen. Diese Anforderung betrifft alle Aktivitäten zur Verifikation und Validierung der Leistungsfähigkeit und Zuverlässigkeit des Features.

Direktes Feedback - Eine Testumgebung soll Entwicklern eine direkte Prüfung ihrer Implementierungen ermöglichen. Das direkte Feedback unterstützt die Entwickler beim detaillierten Systementwurf und der Auswahl geeigneter Realisierungsvarianten. Insbesondere Entwickler mit geringem Expertenwissen in Bezug auf die Anwendung können von direktem Feedback profitieren. Bei der Entwicklung des PCC betrifft dies beispielsweise Softwareentwickler mit geringer Erfahrung im Umgang mit MPC-Algorithmen.

Analyse von Änderungen - Die verwendeten Methoden sollen eine Möglichkeit bereitstellen um die Wirksamkeit umgesetzter Änderungen nachzuweisen und unerwünschte Auswirkungen einer Änderung zu identifizieren. Mögliche Änderungen umfassen dabei sowohl die Implementierung als auch die Parametrierung des Systems.

Usability der Testumgebung - Die genutzte Testumgebung auf Softwareebene soll allen beteiligten Entwickler zur Verfügung stehen und einfach bedienbar sein. Sie soll flexibel gestaltet sein und spezifisch auf die Funktion und die zu beantwortende Fragestellung zugeschnittene Erweiterungen zulassen.

5.3.2 Anwendungsspezifische Anforderungen

Geschlossene Regelschleife - Der PCC stellt ein System für die Fahrzeuglängsregelung dar. Als solches wird, wie in Abschnitt 3.3.4 beschrieben, für die Bewertung der Funktionalität eine geschlossene Regelschleife benötigt, beispielsweise die reale Umgebung oder ein Modell der Regelstrecke.

Validierung am realen Objekt - Die Nutzeranforderungen disruptiver Technologien (s. Definition 2.19) und innovativer Anwendungen sind zu Beginn der Entwicklung aufgrund mangelnder Erfahrung nicht vollständig bekannt und können daher nicht ausreichend detailliert spezifiziert werden. In frühen Phasen der Entwicklung sollen explizit und implizit formulierte Anforderungen am realen Objekt durch Prototypen und Musterstände (s. Abschnitt 3.3.3) erkundet und untersucht werden, um notwendige Anpassungen und Ergänzungen zu identifizieren und eine geeignete Nutzungsqualität nachzuweisen. Neben der Erkundung von Nutzeranforderungen steigert die Validierung am realen Objekt die Erfahrung im Umgang mit der disruptiven Technologie und technischen Rahmenbedingungen. Im Kontext des PCC steigt insbesondere die Erfahrung im Umgang mit den prädiktiven Informationen des elektronischen Horizonts und den Nutzererwartungen an eine an die Straßentopologie angepasste Geschwindigkeitsregelung.

Bewertung der Nutzungsqualität - Im Laufe der Entwicklung sind wiederholt Managemententscheidungen bezüglich der Nutzungsqualität und der markenspezifischen Abstimmung [261] der Fahrzeugregelungsfunktion zu treffen. Dies betrifft im Speziellen das längsdynamische Verhalten des PCC, welches im realen und öffentlichen Straßenverkehr zu evaluieren ist.

Detaillierte Funktionsanalyse - Die Eingänge der MPC bilden einen mehrdimensionalen Parameterraum. Die diesen Raum aufspannenden Eingangssignale sind in unterschiedlichem Maße mit Rauschen behaftet und untereinander korreliert. Die Komplexität dieser Abhängigkeiten erschwert die detaillierte Spezifikation des erwarteten funktionalen Verhaltens erheblich. Analysemethoden, die diese komplexen Zusammenhänge auflösen, sollen die detaillierte Spezifikation des erwarteten funktionalen Verhaltens unterstützen.

Analyse sekundärer Funktionen - Der PCC ist in großem Maße abhängig von sekundären Funktionen, die nicht im Aufgabenbereich der PCC-Entwickler liegen. Beispiele für diese sekundäre Funktionen sind die Funktionselemente elektronischer Horizont, Objektverfolgung und Verkehrszeichen- & Spurerkennung (s. Abbildung 5.2). Eine Möglichkeit zur detaillierten Analyse des funktionalen Verhaltens dieser Funktionen unterstützt das Verständnis der vorgegebenen Rahmenbedingungen.

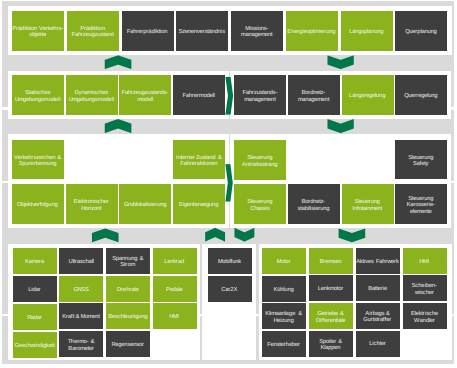
Analyse der Funktionspartitionierung - Die Komplexität der im PCC genutzten modellprädiktiven Regelung, sowie der Elemente zur Vorverarbeitung und Aufbereitung der benötigten Eingangsdaten erschwert die geeignete Partitionierung des Features auf verfügbare Hardwareressourcen. Insbesondere die Interaktion der getrennt partitionierten Software-Items und von der Partitionierung abhängige Signallaufzeiten müssen untersucht werden, um die ausreichende Leistungsfähigkeit des Features nachzuweisen.

5.4 Einordnung der Entwicklungsmethoden

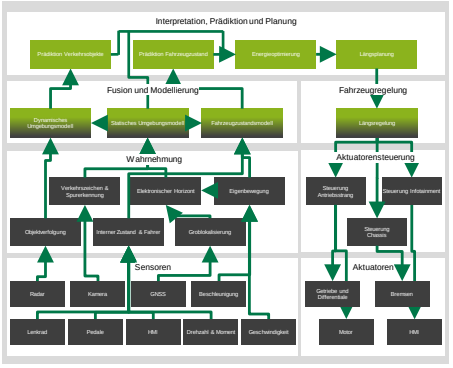
Im Folgenden werden die in Kapitel 3 eingeführten Entwicklungsmethoden auf die im vorhergehenden Abschnitt abgeleiteten Anforderungen und die Entwicklung des PCC-Features (Abschnitt 5.1) abgebildet. Für die frühe Phase der Forschung und Vorentwicklung (Abschnitt 2.9.1) und die Phasen der Serienentwicklung (Abschnitt 2.7 und 2.9) erfolgt eine getrennte Einordnung. Abbildung 5.4 skizziert die Anwendung der in Kapitel 2.2.2 eingeführten Systemabstraktionsebenen auf das PCC-Feature.

Die Systemebene (Abb. 5.4a) beinhaltet alle funktionalen Elemente der logischen Systemarchitektur des Gesamtfahrzeugs und wird durch die in Abschnitt 4.3 vorgestellte Beispielarchitektur repräsentiert. Die funktionale Wirkkette des PCC (Abschnitt 5.1) bildet ein Subsystem (Abb. 5.4b). Auf dieser Ebene werden während

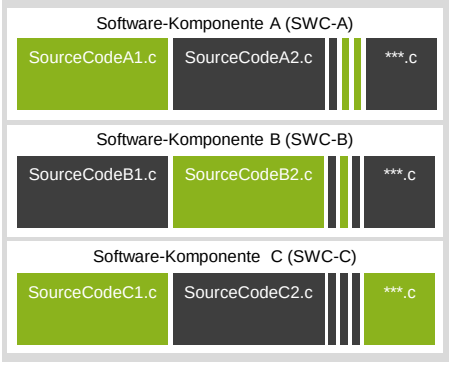
des Systemarchitekturentwurfs (SYS.3, Abschnitt 2.7) alle für die prädiktive Längsregelung relevanten Funktionsblöcke zusammengefasst. Dies beinhaltet auch die für Wahrnehmung und Ausführung benötigten Funktionen, die nicht im expliziten Aufgabenbereich der PCC-Entwicklung liegen.



(a) Darstellung des PCC-Features auf Systemebene. Zugehörige Funktionsblöcke sind grün markiert.



(b) Darstellung des PCC-Features auf Subsystemebene (vergl. Abb. 5.2). PCC-spezifische Funktionsblöcke sind grün markiert.



(c) Beispielhafte Aufteilung der Funktionsblöcke des PCC-Features auf verschiedene Softwarekomponenten. Grün markierte Quelltexte stellen PCC-Anteile dar

SourceCodeA1.c

```
/**
* Umsetzung zusammenhängender Funktionalität
*/

/** \brief Initialisierung */
void functionUnitA1_Init(INOUT int *data) {
   data = INITVALUE;
}

/** \brief Berechnung der euklidischen Distanz */
float calcDist(IN const float &dx, IN const float &dy)
{
   float dist = 0.0f;
   dist = sqrt(pow(dx, 2.0f) + pow(dy, 2.0f));
   return dist;
}
```

(d) Beispiel für in einer Softwareeinheit umgesetzte zusammenhängende Funktionalität des PCC-Features

Abbildung 5.4: Schematisches Anwendungsbeispiel der in Kapitel 2.2.2 eingeführten Systemabstraktionsebenen auf den prädiktiven Abstandsregeltempomat

Auf Komponentenebene werden die Funktionselemente im Rahmen der Partitionierung des Softwarearchitekturentwurfs (SWE.2, Abschnitt 2.7) den Software-Komponenten zugeordnet (Abb. 5.4c). Die in der Subsystemebene hellgrün hinterlegten Funktionsblöcke werden den Software-Komponenten des PCC-Features zugeord-

net, wobei nicht alle Blöcke eindeutig zuordenbar sind. Die Software-Komponenten fassen die jeweiligen Quelltextdateien der implementierten Komponente zusammen. Diese beinhalten die in C-Funktionen implementierten Software-Einheiten (Abb. 5.4d).

5.4.1 Forschung und Vorentwicklung

Die Entwicklung einer innovativen Kundenfunktion oder einer Featureerweiterung startet mit der Ideenfindung und der Definition eines ersten abstrakten Umsetzungskonzepts (s. Abschnitt 2.9.1). Zur Analyse und Validierung dieses Umsetzungskonzepts wird eine Machbarkeitsstudie durchgeführt, deren Ziel die prototypische Realisierung der Kundenfunktion ist.

In den verschiedenen Stadien der Forschung und Vorentwicklung des PCC-Features wurden und werden dazu durchgängig die in Abschnitt 3.3.3 eingeführten RP-Systeme eingesetzt. Sie ermöglichen die geforderte Validierung am realen Objekt. Radtke [260] und Wahl [218] beschreiben den Einsatz einer dSpace MicroAutobox und eines Fahrzeug PCs mit der Entwicklungssoftware ADTF in den Erprobungsfahrzeugen. Für die Durchführung der ersten Machbarkeitsstudie wird die Systemumgebung zur Durchführung der Versuche stark eingeschränkt. Beispielsweise beschränkte sich die genutzte Versuchsstrecke für die Entwicklung des PCC-Features zu Beginn auf einen Rundkurs von insgesamt knapp 23 km [260].

Nach dem ersten erfolgreichen Machbarkeitsbeweis und der Entscheidung zur Fortführung der Entwicklung ändert sich deren Zielsetzung. Es gilt die eingesetzten Algorithmen und das Funktionskonzept zu konkretisieren. Neben der Darstellung der eigentlichen Funktionalität rücken Fragestellungen zur Laufzeit der Algorithmen und zu deren Speicherbedarf sowie eine generalisierte Darstellung ohne Einschränkung des Anwendungsbereichs in den Fokus. Mit der Ausweitung des Anwendungsbereichs werden neue und vielfältige Situationen und Szenarien erschlossen. Diese decken neue Bereiche des Zustandsraums der Längsregelungsfunktion ab und steigern die notwendigen Aufwände für die Analyse des Systemverhaltens.

Die detaillierte Analyse des Systemverhaltens erfordert eine das RP ergänzende Alternative, die eine von der Echtzeit entkoppelte Ausführung des Systems ermöglicht. Wie in Abschnitt 3.3.4 eingeführt wurde, deckt der Einsatz einer Gesamtfahrzeugsimulation die Anforderungen nach einer geschlossenen Regelschleife

und Reproduzierbarkeit ab. Die Nutzung von Simulationsmodellen ermöglicht genau definierte Umgebungsbedingungen und wendet Vereinfachungen an, ist jedoch gleichzeitig mit hohen Aufwänden verbunden. Die in Abbildung 3.7 eingeführten Bestandteile einer Gesamtfahrzeugsimulation verdeutlichen den notwendigen Ressourcenaufwand für die Modellbildung und Parametrierung einer hinreichend realistischen Darstellung der Systemumgebung. Aktuell verfügbare Simulationswerkzeuge erfüllen zwar die identifizierte Anforderung der Reproduzierbarkeit, sind aber aufgrund des hohen Aufwands zur Spezifikation einzelner Testszenarien nur bedingt geeignet eine hinreichende Abdeckung der notwendigen Varianten zu liefern. Der Umfang und die Komplexität allgemein anwendbarer Gesamtfahrzeugsimulationen und die üblicherweise umgesetzten Ansätze zur Co-Simulation erschweren darüber hinaus die Nutzbarkeit und die Fähigkeit der Werkzeuge direktes Feedback zu liefern.

Die Entwicklung der MPC setzt eine proprietär entwickelte Simulationsumgebung ein [218]. Sie ist Teil des PCC-Entwicklungsprojekts, steht allen Beteiligten zur Verfügung und ermöglicht die Durchführung aller Anpassungen innerhalb der gewohnten Entwicklungsumgebung. Das zugrundeliegende Längsdynamikmodell der Simulation stimmt mit dem Modell der MPC überein. Bauer [219] beschreibt wie das Aufschalten von Störgrößen zur Analyse der Regelungsgüte und Robustheit genutzt wird. Einen umfassenden Überblick über die Fahrzeugdynamik bieten Mitschke und Wallentowitz [257].

In fortgeschrittenen Phasen der Vorentwicklung soll der initial geführte Machbarkeitsbeweis auf die Allgemeinheit ausgedehnt werden und als Entscheidungsgrundlage für eine Serieneinführung dienen. Für einen überzeugenden Fähigkeitsnachweis in den zuständigen Entscheidungsgremien muss der Prototyp eine ausgereifte Nutzungsqualität aufweisen und innerhalb der eingeschränkten Systemumgebung zuverlässig funktionieren. Um diese Qualität zu erreichen bieten Werkzeuge zur statischen Codeanalyse (siehe Abschnitt 3.4.2) einen hohen Mehrwert, ohne mit hohen Aufwänden verbunden zu sein. Aufwändige Spezifikationen der detaillierten Anforderungen und entsprechender Unit- und Integrationstests werden vermieden. Die Hauptaufwände werden in die zentrale Zielsetzung der Vorentwicklung, den Nachweis der Funktionalität und Nutzungsqualität, investiert. Der Nachweis der Qualitätsmerkmale Leistungsfähigkeit, Kompatibilität und Zuverlässigkeit innerhalb einer uneingeschränkten Systemumgebung ist in der Serienentwicklung verortet.

5.4.2 Anforderungsspezifikation und Systementwurf

Für die während der Anforderungserhebung (Kapitel 2.7) spezifizierten Anforderungen und deren Verfolgung werden in der Automobilindustrie spezifische Werkzeuge eingesetzt. Für das Anforderungsmanagement in der Entwicklung des PCC-Features wird das Werkzeug Rational DOORS[1] genutzt. Zur Gewährleistung der Nachverfolgbarkeit werden den spezifizierten Anforderungen eindeutige Referenzen zugewiesen, die mit den jeweiligen Elementen des Systementwurfs (SYS.3), den implementierten Items und den zugehörigen Testfällen verknüpft werden. Auf der Einheitenebene (Kapitel 2.2.2) werden beispielsweise die entsprechenden Anforderungen einer Software-Einheit in den zugehörigen Kommentaren im Quelltext referenziert. Analog dazu wird mit den spezifizierten Testfällen verfahren. Durch dieses Vorgehen ist die Nachverfolgbarkeit im Rahmen der PCC-Entwicklung gewährleistet.

Basierend auf den während der Vorentwicklung gewonnenen Erkenntnissen werden die Anforderungen des PCC entsprechend des organisationsspezifischen Prozesses in Textform erfasst. Durch den in der Vorentwicklung gewonnenen technischen Einblick besteht die Gefahr, dass die Entwicklungsphasen zur Anforderungserhebung und -analyse (SYS.1-2 & SWE.1, Kapitel 2.7) mit den Entwurfsphasen (SYS.3 & SWE.2-3, Kapitel 2.7) vermischt werden und Details zu Entwurf und Umsetzung bereits in den Anforderungen formuliert werden. Eine solche Vermischung erschwert in den darauffolgenden Phasen (SWE.4-6 & SYS.4-5, Kapitel 2.7) die Spezifikation von Unit-Tests (Kapitel 3.5.3), Integrationstests (Kapitel 3.5.4) und Qualifikationstests (Kapitel 3.5.5). Die Durchführung manueller Prüfungen, wie in Abschnitt 3.4.1 beschrieben, wirkt dem entgegen, da Richtlinien und Vorgaben zur Anforderungsspezifikation durchgesetzt werden.

Die im Rahmen des System- und Softwareentwurfs festgelegte Systemarchitektur und Softwarepartitionierung liegt im übergeordneten Aufgabenbereich der Baureihe (Kapitel 2.9) und zugehörigen Fahrzeugdomänen. Die Möglichkeiten und Freiheitsgrade zur Einflussnahme durch eine Feature-Entwicklung sind dabei sehr begrenzt. Die wichtigsten Aufgaben auf Seiten der Feature-Entwicklung sind die Abstimmung des Bedienkonzepts, der zugesicherten Prozessorlaufzeit, des Speicherbedarfs, der Schnittstellen und der Steuergerätediagnose. Spezifische modellbasierte Methoden und Werkzeuge, wie in Abschnitt 3.3.1 vorgestellt, sind Teil der übergeordneten Querschnittaktivitäten und werden im Rahmen der PCC-Entwicklung nicht weiter betrachtet.

[1] IBM Rational DOORS(Dynamic Object Oriented Requirements System), IBM Deutschland GmbH, http://www-03.ibm.com/software/products/de/ratidoor

5.4.3 Detaillierter Entwurf und Implementierung

Auf Basis der in den Entwurfsphasen (SYS.3 & SWE.2, Kapitel 2.7) abgestimmten groben Softwarearchitektur und der in ihr enthaltenen Software-Komponenten wird der detaillierte Entwurf festgelegt und implementiert (SWE.3, Kapitel 2.7). Die in der Vorentwicklung erstellte prototypische Implementierung bildet die Grundlage. Sie wird entsprechend des spezifizierten Feature-Umfangs der Serienentwicklung überarbeitet und refaktoriert. Für den Test der durchgeführten Refaktorierung der Software-Einheiten können die in Abschnitt 3.5.1 beschriebenen testvektorbasierten Back-To-Back Tests genutzt werden.

Der Einsatz von Werkzeugen zur statischen Codeanalyse prüft die Umsetzung des in Abschnitt 3.4.2 vorgestellten Standards MISRA-C und der HIS Source Code Metrics. Diese Prüfung dient der Validierung der Softwarezuverlässigkeit und -wartbarkeit durch den Nachweis der Abwesenheit kritischer Konstrukte und einer ausreichend detaillierten Dokumentation des Quelltextes. Code-Reviews ermöglichen eine zusätzliche manuelle Prüfung während der Implementierung. Da die Durchführung von Code-Reviews mit hohen Aufwänden verbunden ist, werden nur algorithmisch aufwändige Software-Einheiten einer manuellen Prüfung unterzogen.

Die Nutzung der im vorhergehenden Abschnitt 5.4.1 beschriebenen Simulationsumgebung ermöglicht in der Phase des detailliertern Entwurfs und der Implementierung (SWE.3, Kapitel 2.7) die closed-loop Ausführung der Software und damit die Analyse des funktionalen Regelungsverhaltens auf Subsystemebene. Im Fokus dieser Analysen steht die umgesetzte Funktionalität, die Leistungsfähigkeit in Bezug auf Prozessorlaufzeit und Speicherbedarf sowie die Zuverlässigkeit der in der Software umgesetzten MPC. Die Simulation wird während des detaillierten Entwurfs und der Implementierung durch die Validierung am realen Objekt auf Basis des A-Musters unterstützt. Die spezifizierten Simulationsszenarien werden durch die Varianten realer Situationen und Szenarien ergänzt. Für das A-Muster wird weiterhin spezielle leistungsfähige RP-Hardware eingesetzt. Die Softwarearchitektur entspricht in großen Teilen bereits der Zielarchitektur, mit Ausnahme der Basissoftware, die durch das RP-System ersetzt wird.

Damit in der Phase SWE.3 ohne einen Musterstand der Zielhardware zur Verfügung zu haben detailliertere Aussagen über den Ressourcenbedarf getroffen werden können, kommt die PIL-Methode (siehe Kap. 3.5.2) zum Einsatz. Testvektoren stellen die benötigten Stimuli zur Verfügung, die zur Ausführung der Software auf einer

Entwicklungshardware mit einer zur Zielhardware identischer Prozessorarchitektur benötigt werden. Sie können manuell spezifiziert werden, aus der Simulationsumgebung abgeleitet werden oder aus Aufzeichnungen von Erprobungen mit dem A-Muster stammen. Um eine funktionale Evaluation der Regelungsgüte auf der Prozessorarchitektur des Zielsystems zu ermöglichen, kann die PIL-Umgebung mit der Simulationsumgebung des Gesamtsystems gekoppelt werden.

5.4.4 Integration, Verifikation und Validierung

Die Prozessschritte auf der rechten Seite des V-Modells in Automotive SPICE (Kap. 2.7) dienen der Integration, Verifikation und Validierung der implementierten Items. Die Ziele und zu ihrer Erreichung eingesetzten Methoden und Werkzeuge ändern sich mit jedem Prozesschritt. Zur Verifikation der Implementierung auf Einheitenebene (SWE.4) werden Unit-Tests umgesetzt. Der Umfang der spezifizierten Tests wird mithilfe der Metriken zur Ermittlung der Testabdeckung aus Abschnitt 3.7.2 bestimmt. Die anzuwendende Metrik wird vom Projektverantwortlichen in Abstimmung mit dem oder den Qualitätsverantwortlichen entsprechend der übergeordneten Teststrategie des Unternehmens festgelegt. Der Nachweis vollständiger Anforderungs-, Anweisungs- und Zweigüberdeckung stellt einen branchenüblichen (Mindest-)Standard dar.

Nach Automotive SPICE (Abschnitt 2.7) folgt auf die Verifikation der Softwareeinheiten der Prozessschritt SWE.5 Softwareintegration und Integrationstests. Die Zielsetzung dieses Schritts ist der Übergang von der Einheitenebene auf die Komponentenebene durch die Integration der einzelnen Software-Einheiten in einer vollständigen Softwarekomponente (Software Component, SWC). Während der sukzessiven Integration der Software-Einheiten werden die Software-Integrationstests durchgeführt. Dazu werden basierend auf den in Prozesschritt SYS.1 definierten Anforderungen Testfälle spezifiziert, die die teilintegrierte Software innerhalb einer SIL-Umgebung prüfen. Testvektoren dienen als Stimuli an den jeweiligen Schnittstellen der Einheiten, um einen Nachweis über die spezifikationskonforme Interaktion der integrierten Einheiten zu erbringen. Für die Verifikation der Schnittstellen werden Spezifikationsmethoden wie die in Abschnitt 3.7.1 vorgestellte Äquivalenzklassenbildung oder Grenzwertanalyse eingesetzt. Die Integrationstests weisen die Kompatibilität und Zuverlässigkeit der einzelnen Software-Einheiten nach.

Die Softwarequalifikation erbringt einen Nachweis über die korrekte Umsetzung

der spezifizierten Anforderungen. Die vollständig integrierte SWC wird in der SIL-Umgebung mit explizit für den Nachweis der Anforderungen definierten Testfällen geprüft. Für den Nachweis der Funktionalität mit komplexeren Testszenarien wird eine geschlossene Regelschleife benötigt. Die SIL-Testumgebung wird mit der Simulationsumgebung gekoppelt und die Regelung wird innerhalb der virtuellen Wirkkette anhand der spezifizierten Testszenarien geprüft. Die Tests zur Softwarequalifikation erbringen einen Nachweis über die Qualität in Bezug auf Funktionalität und Zuverlässigkeit.

Der Einsatz einer Werkzeugkette zur Anwendung von Continuous Integration (CI) und Continuous Verification, wie in Abschnitt 3.3.5 beschrieben, ermöglicht eine kontinuierliche Überwachung der Softwarequalität und Testabdeckung in den Phasen der Integration, Verifikation und Validierung. Die Werkzeuge zur Durchführung der zuvor beschriebenen Unit-Tests, Integrationstests und Qualifikationstests werden dazu mit der CI-Werkzeugkette gekoppelt und automatisiert gestartet und ausgewertet. Damit ist jederzeit der aktuelle Entwicklungsfortschritt verfügbar.

Auf die Softwarequalifikation folgt mit der Systemintegration der Übergang auf die Subsystemebene und letztendlich die Systemebene. In einem ersten Schritt werden die SWCs mit der Basissoftware auf einem B-Musterstand der ECU integriert. Diese erste Integration erfolgt bereits vor dem vollständigen Abschluss der Softwareimplementierung mit einem in der Funktionalität reduzierten Softwarestand. Auf Basis dieser ersten Integration werden am Komponenten-HIL die Hardware-Schnittstellen des Steuergeräts geprüft und Laufzeitanalysen der kritischen Software-Einheiten durchgeführt. Die für diese Tests benötigten Schnittstellen werden durch den Komponenten-HIL emuliert. Die Spezifikation der Testfälle erfolgt analog zu den Softwareintegrationstests. Sie erbringen einen Nachweis über die Leistungsfähigkeit und Effizienz sowie die Kompatibilität der einzelnen SWCs in Interaktion mit der gesamten Software der ECU.

Neben der Integration am Komponenten-HIL dient der B-Musterstand dem Aufbau der ersten Prototypenfahrzeuge der entwickelten Baureihengeneration. Ziel dieser Prototypen ist die Analyse und der Nachweis der Funktionalität, Kompatibilität und Zuverlässigkeit der grundlegenden Elemente innerhalb eines Gesamtsystems, das bereits in großen Teilen der finalen Realisierung entspricht. Im Fokus stehen die hardwarenahen Funktionselemente der unteren Ebenen der in Abschnitt 4.3 vorgestellten logischen Systemarchitektur. Daher spielt die Erprobung auf dem B-Musterstand für die Entwicklung des PCC-Features nur eine untergeordnete Rolle.

Mit Erreichen des C-Musterstands beginnt die Integration des vollständigen Subsystems am Cluster-HIL oder Wirkkettenprüfplatz. Integrations- und Qualifikationstests am Cluster-HIL erbringen einen Nachweis über die korrekte Interaktion und Funktionalität auf Subsystemebene. Nicht verfügbare Teile des Systems und notwendige Elemente der Systemumgebung werden mit Hilfe von Werkzeugen zur Gesamtfahrzeugsimulation ersetzt. Am Cluster-HIL werden verschiedene Feature-Entwicklungen zusammengefasst, daher sind die möglichen Testumfänge begrenzt. Die Tests am Cluster-HIL weisen die Funktionalität und Zuverlässigkeit des Subsystems und die Kompatibilität der einzelnen ECUs nach.

Parallel zu den Integrations- und Qualifikationstests am Cluster-HIL werden mit dem C- und D-Musterstand Erprobungen des Gesamtsystems in verschiedenen Ländern und unter unterschiedlichen klimatischen Bedingungen durchgeführt. Diese Validierung am realen Objekt ermöglicht die Überprüfung der Funktionalität, Zuverlässigkeit und Usability. Für den formalen Nachweis werden Prüfkataloge mit textuell beschriebenen Testszenarien definiert, die systematisch abgearbeitet werden.

5.5 Offene Herausforderungen und mögliche Lösungen

Die eingesetzten Methoden zur dynamischen Verifikation und Validierung basieren zu einem großen Teil auf manuell, entsprechend den Anforderungen spezifizierten Testfällen. Das PCC-Feature wird als Ganzes und in seinen Einzelelementen in der Realität und unter Laborbedingungen geprüft. Die Prüfungen und Tests erbringen einen Nachweis über die Funktionalität, Leistungsfähigkeit, Kompatibilität, Usability und Zuverlässigkeit anhand ausgewählter Eckfälle des Zustandsraums, die die Funktions- und Systemgrenzen sowie Standardwerte abdecken. Der Nachweis der Leistungsfähigkeit und Kompatibilität erfolgt am Cluster-HIL, der Nachweis korrekter Funktionalität, Usability und Zuverlässigkeit basiert in großem Maße auf den Gesamtsystemerprobungen mit realen Fahrzeugen, die den größten Anteil zur statistischen Signifikanz beitragen. Die beschriebene Zunahme an abzusichernden Features und die steigende Vernetzung und algorithmische Komplexität führen zu einem starken Anstieg der Test- und Prüfaufwände in allen Phasen der Vor- und Serienentwicklung und können mit den etablierten Methoden zukünftig nicht mehr ausreichend abgesichert werden [7].

Um in zukünftigen Entwicklungen bei der in Abschnitt 5.2 beschriebenen steigenden

Anzahl unterschiedlicher Situationen und Szenarien eine ausreichende Testabdeckung zu gewährleisten, werden die etablierten Methoden in zunehmendem Maße durch skalierbare virtuelle Testumgebungen ergänzt. Abbildung 5.5 skizziert die Unterteilung der PCC-Wirkkette in ein zu untersuchendes Testobjekt und die benötigte skalierbare Testumgebung anhand der logischen Systemarchitektur.

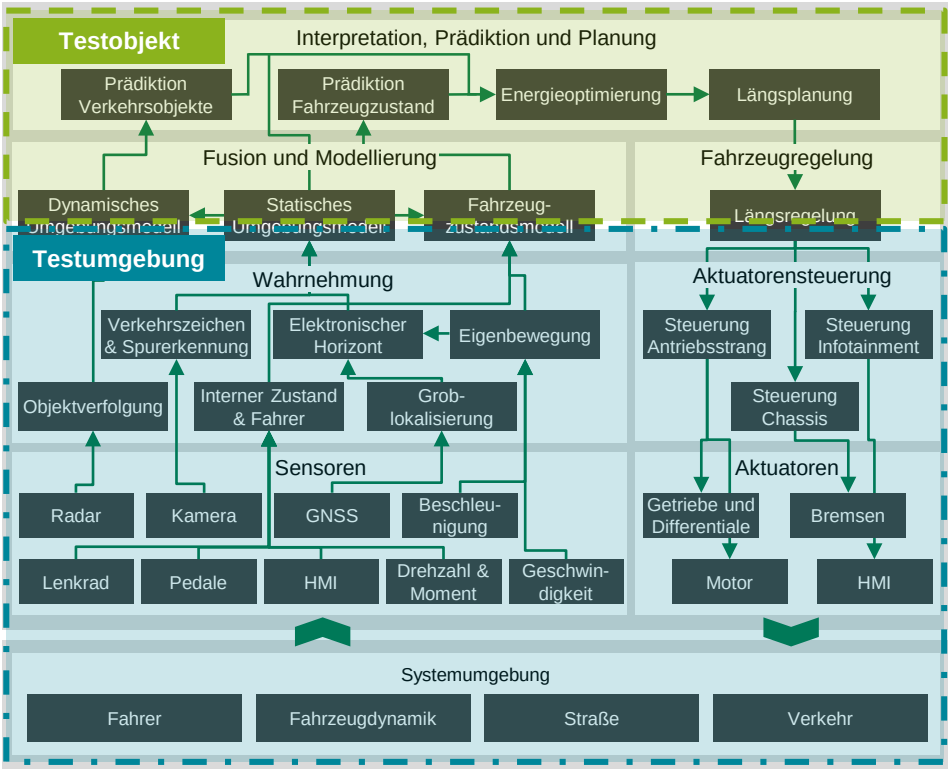


Abbildung 5.5: Logische Systemarchitektur der PCC-Wirkkette (s. Abb. 5.2) unterteilt in Testobjekt und Testumgebung.

Eine große Herausforderung stellt dabei die Definition geeigneter Testszenarien dar, die eine breite Abdeckung des Zustandsraums der Testobjekte gewährleisten.

Die in Abschnitt 3.3.4 erwähnten Ansätze zur automatisierten Generierung der Testszenarien erfordern hohe initiale Aufwände für die Erstellung und Parametrierung der genutzten Modelle. Neben den Aufwänden für Modellierung und Parametrierung erfordert eine detaillierte Nachbildung physikalischer und optischer Effekte, wie Radarreflexionen oder Überblendungen, leistungsstarke Hardware. Dies erschwert die Nutzung der Modelle für die tägliche Arbeit der Entwickler.

Im Bereich der Bildverarbeitung ist die Nutzung aufgezeichneter Videodaten für die Entwicklung und den Test von Algorithmen weit verbreitet [262, 131]. Zum einen können dadurch die Aufwände für eine akkurate Modellierung der vielfältigen optischen Effekte in jedem einzelnen Bild vermieden werden, zum anderen können Videodaten ohne große Aufwände während ohnehin durchgeführten Erprobungen aufgezeichnet werden.

Wie in Abschnitt 5.4 beschrieben wurde, werden zur Evaluation von Prototypen und Musterständen. sowie zu Freigaben Erprobungen durchgeführt. Bislang werden die dabei aufgezeichneten Datenreihen (vergleiche Abbildung 5.6) der internen Fahrzeugkommunikation hauptsächlich zur Analyse des funktionalen Verhaltens einzelner Kundenfunktionen und Wirkketten genutzt. Die in diesen Daten enthaltene Information kann in einer zur Bildverarbeitung ähnlichen Weise genutzt werden, um während der Entwicklung von Funktionen höherer Ebenen die vorherrschenden spezifikationsbasierten Ansätze zu ergänzen und eine größere Bandbreite an Varianten abzudecken.

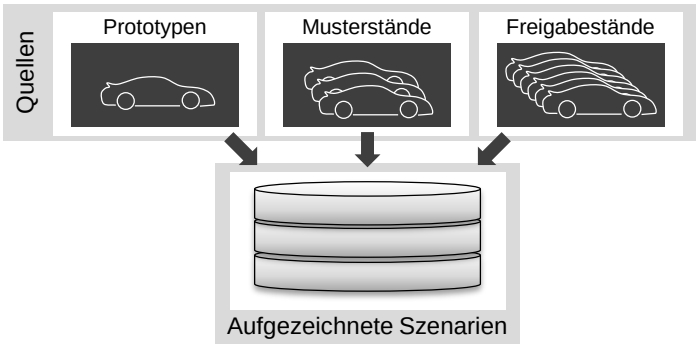


Abbildung 5.6: Verschiedene, während der Entwicklung nutzbare Datenquellen zur Ableitung von Szenarien.

Neben einer größeren Anzahl verfügbarer Varianten, ermöglicht eine Zusammenführung der Funktionserprobungen im Fahrversuch mit den Tests in der simulierten Umgebung eine verbesserte Analyse und Bewertung des Funktionsverhaltens in spezifischen Situationen. Solche für die Entwicklung besonders interessanten Situationen treten bei der Erprobung mit dem prototypischen System auf und erweitern die Erfahrung der Entwickler im Umgang mit dem System. Neben auffälligen Situationen mit unerwünschtem Regelungsverhalten sind dies beispielsweise Situationen, die den Mehrwert der gewählten Umsetzung gegenüber anderen Ansätzen besonders verdeutlichen. Eine direkte und automatisierte Überführung dieser Situationen und Szenarien in die Simulation ermöglicht es, die Wirksamkeit umgesetzter Ände-

rungen schnell und effizient zu prüfen und die gesammelte Erfahrung nachhaltig erlebbar zu halten.

Um diese Aufzeichnungen für die Entwicklung und den Test von Fahrzeugregelungsfunktionen nutzen zu können wird im folgenden Kapitel eine Methode vorgestellt, die die Nutzung aufgezeichneter, aus der Fahrzeugkommunikationsstruktur (ergl. Kap. 2.2.1) stammender Eingangsdaten des PCC-Features innerhalb einer Simulation mit geschlossener Regelschleife ermöglicht.

6 Reactive-Replay - Messdatenbasierte Closed-Loop Simulation

Die Nutzung der aufgezeichneten fahrzeuginternen Kommunikation für die Entwicklung und den Test von Funktionen mit Open-Loop Charakter stellt eine leistungsfähige Möglichkeit für deren simulierte Ausführung und Bewertung dar. Eine Voraussetzung für die Nutzung der Daten ist eine ausreichend präzise Zeitauflösung der Aufzeichnung. Die assistierenden und automatisierenden Funktionen höherer Abstraktionsebenen nutzen, wie in Abschnitt 4.3 beschrieben wurde, die über das interne Kommunikationsnetzwerk des Fahrzeugs verfügbaren Informationen der Funktionen unterer Ebenen. Abbildung 6.1 stellt beispielhaft einen Ausschnitt von zwei der im PCC-Feature genutzten Signale dar.

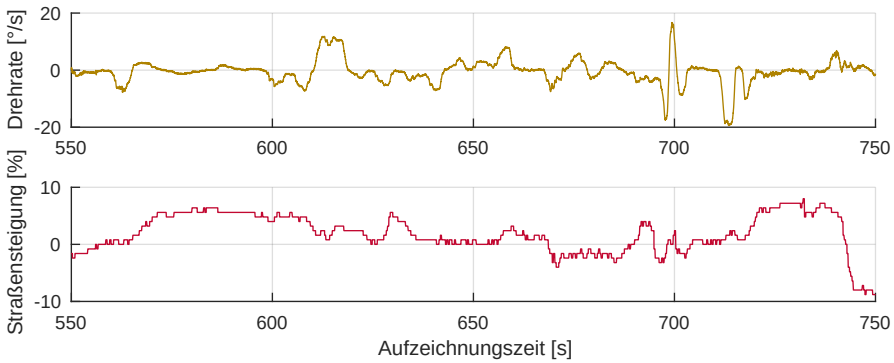


Abbildung 6.1: Darstellung eines Ausschnitts aufgezeichneter Eingangsdaten der fahrzeuginternen Kommunikation am Beispiel von Signalen für die Drehrate und Straßensteigung.

Die in den Kommunikationsnetzen üblichen Zykluszeiten kontinuierlich bereitgestellter Signale von 10-20 ms stellen für aktuelle Datenlogger keine Herausforderung dar[1]. Eine zeitsynchrone Wiedergabe dieser Daten ermöglicht in der virtuellen

[1] Z.B. XORAYA Datenlogger: 100 ns, XORAYA Datenlogger 6810 Quad V5, X2E GmbH, http://www.x2e.de/de/produkte/xoraya_datenlogger_v5.php

Testumgebung die wiederholte Ausführung einer Open-Loop Funktion innerhalb des realistischen Testszenarios und eine detaillierte Analyse der enthaltenen Situationen.

Für die prädiktive Längsregelung des PCC-Features stellt die Nutzung einer zeitsynchronen Wiedergabe nur einen begrenzten Mehrwert dar. Die open-loop Simulation eignet sich zur detaillierten Analyse von Auffälligkeiten des während des Fahrversuchs genutzten Softwarestands. Die virtuelle Wiedergabe ergänzt die Erprobung durch eine von der Echtzeit entkoppelten Wiedergabe und unterstützt so die detaillierte Analyse des Systemverhaltens und der internen Zustände. Werden jedoch Änderungen an der Software oder ihrer Parametrierung durchgeführt, um ein angepasstes Regelungsverhalten zu erreichen, weicht das aufgezeichnete Verhalten vom erwarteten Regelungsverhalten ab. Folglich verschlechtert sich die Aussagekraft der open-loop Simulation mit zunehmender Änderung des Regelungsverhaltens bis zur Unbrauchbarkeit.

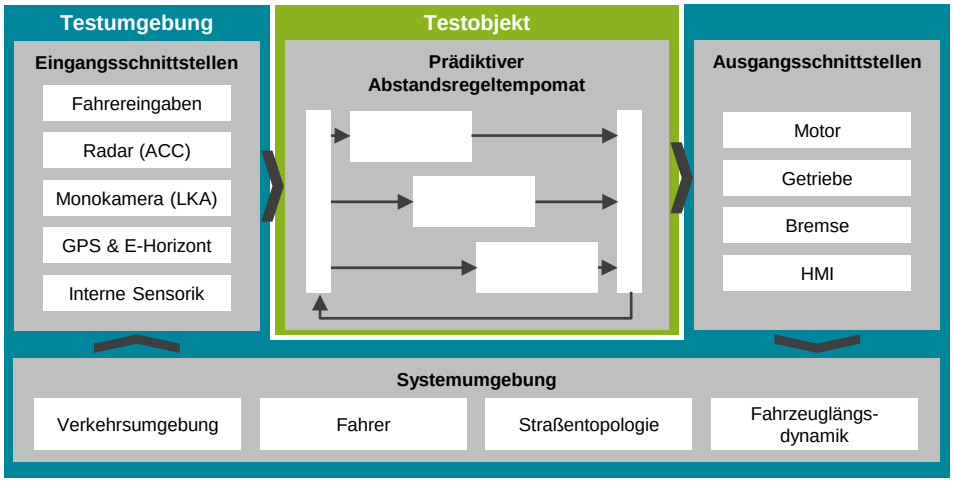


Abbildung 6.2: Abstrahierte Darstellung der PCC-Wirkkette in Anlehung an [218], ergänzt um die in Abbildung 5.5 eingeführten Aufteilung in Testobjekt & -umgebung.

Abbildung 6.2 bietet eine stärker abstrahierte Darstellung der PCC-Wirkkette, aufgeteilt in Testobjekt und Testumgebung. Die Ein- und Ausgangsschnittstellen des Testobjekts repräsentieren die über das Kommunikationsnetz des Fahrzeugs realisierte Verbindung mit verschiedenen im Fahrzeug verteilten Steuergeräten. Um die Analyse der Wirkkette zu vereinfachen, sind sie thematisch zusammengehörig gruppiert. Der Begriff Egofahrzeug ermöglicht im Folgenden eine eindeutige Unterscheidung zwischen dem eignen Fahrzeug und durch Sensorik erfassten Ver-

kehrsobjekten. Die konkreten Signale der in der Simulation zu substituierenden Eingangsschnittstellen des PCC sind in der folgenden Liste zusammengefasst:

- Fahrereingaben (Lenkradwinkel, Pedalpositionen, Bedienaktionen)
- Radar des ACC (Relative Position, Geschwindigkeit und Beschleunigung erfasster Verkehrsobjekte in Bezug zum Egofahrzeug)
- Monoskopische Frontkamera (Geschwindigkeitslimits sowie relative Position, Krümmung, Breite und Länge erkannter Fahrspurmarkierungen)
- GPS Lokalisierung des Egofahrzeugs und elektrischer Horizont(globale Fahrzeugposition, vorausliegende Kurvenkrümmung, Streckensteigung und Geschwindigkeitslimits)
- Interne Sensorik (Geschwindigkeit, Beschleunigung, Momente und Drehzahlen des Motors, Getriebes und der Räder, aktueller Getriebegang, geschätzte Streckensteigung)

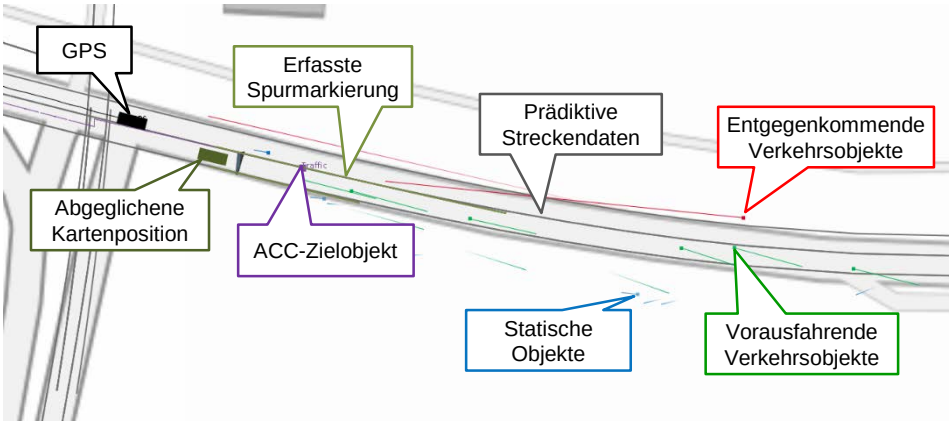


Abbildung 6.3: Visualisierung der Umfelderfassung des PCC-Features mit GPS-Position, prädiktiver Streckenvorschau des elektronischen Horizonts, erfassten Spurmarkierungen, erfasstem ACC-Zielobjekt und statischen und dynamischen Objekten (Unterlegte Karte: © OpenStreetMap contributors).

Abbildung 6.3 visualisiert die zur Umfelderfassung genutzten Eingangsschnittstellen des PCC. Die Sicht aus der Vogelperspektive vereint die Umfeldinformationen der beschriebenen Sensoren über einer geografischen Kartendarstellung. Es ist eine Differenz zwischen der GPS-Position (schwarz) und der im elektronischen Horizont

abgeglichenen Kartenposition (olive) des Egofahrzeugs zu erkennen. Die optisch erfassten Spurmarkierungen (ocker) decken sich größtenteils mit den prädiktiven Streckendaten. Die durch das Radar erfassten Objekte sind entsprechend ihrer Klassifizierung als Zielobjekt (violett), statische (blau), vorausfahrende (grün) und entgegenkommende Objekte (rot) dargestellt.

Diese Eingänge des Systems sind in unterschiedlicher Weise von der Fahrzeuglängsdynamik abhängig. Um einen größtmöglichen Nutzen aus den aufgezeichneten Daten des Fahrversuchs zu ziehen und die Aufwände für Modellierung und Parametrierung der Simulation möglichst gering zu halten, werden im folgenden Abschnitt die kausalen Zusammenhänge der Wirkkette analysiert und entsprechend ihren Abhängigkeiten kategorisiert. Aufbauend auf dieser Kategorisierung wird die Reactive-Replay Methode vorgestellt, die das in der bestehenden Simulation genutzte Regelstreckenmodell der Fahrzeuglängsdynamik mit den aufgezeichneten Erprobungsdaten kombiniert und damit eine an das Regelungsverhalten angepasste Wiedergabe der Daten ermöglicht. Teile dieses Kapitels wurden bereits im Rahmen eines Konferenzbeitrags veröffentlicht [JB3].

6.1 Analyse der kausalen Zusammenhänge

Die Betrachtung kausaler, d.h. auf Ursache und Wirkung bezogener Zusammenhänge, ermöglicht eine detaillierte Analyse der komplexen Abhängigkeiten innerhalb eines technischen Systems. Beispielsweise beschreibt Wiese [263] eine Methode zur Analyse und Identifikation von Fehlerwirkungen und Fehlerfortpflanzungen basierend auf der Modellierung System inhärenter Ursache-Wirkungs-Pfade in der logischen Systemarchitektur. Der Reactive-Replay Ansatz basiert auf einer Ursache-Wirkungsanalyse des PCC zur Unterscheidung zwischen Systemeingängen, die eine direkte Rückkopplung auf eine Stellgröße des Regelungsalgorithmus darstellen, und Systemeingängen, die indirekt von einer Stellgröße und/oder einer externen Abhängigkeit beeinflusst werden. Die zentrale Idee des Ansatzes ist es, Signale mit direkter Rückkopplung durch ein Simulationsmodell zu ersetzen und die übrigen Signale aus den aufgezeichneten Daten zu extrahieren und entsprechend der Ursache-Wirkungs-Beziehung an das Ergebnis des Simulationsmodells anzupassen.

6.1.1 Mathematische Herleitung

Die kausalen Zusammenhänge des PCC-Regelungssystems können mit drei Eingangssignalen $x(t)$, $y(t)$ und $z(t)$ sowie dem Ausgang $u(t)$ vereinfacht durch

$$u\left(t\right) = f\left(x\left(t\right), y\left(t\right), z\left(t\right)\right) \tag{6.1}$$

ausgedrückt werden. Dabei ist $x(t)$ eine Komponente der Rückkopplung und monoton steigend. Damit kann dieses Signal als eindeutige Referenzgröße genutzt werden. $y(t)$ ist direkt von $x(t)$ abhängig und kann damit direkt referenziert werden. $z(t)$ stellt ein Signal dar, das von der Zeit t und von $x(t)$ abhängt. Es kann durch einzelne zeitabhängige Sequenzen beschrieben werden, die durch Ereignisse ausgelöst werden, die von $x(t)$ abhängig sind.

Die Datenreihe einer zugehörigen Aufzeichnung von N Abtastungen und der Abtastzeit Δt kann als diskretes Signal

$$x_n = x\left(n\Delta t\right) = x\left(t_n\right) \tag{6.2}$$

mit dem positiv ganzzahligen Index $n \in \{0, 1, 2, ..., N-1\}$ und der entsprechenden Aufzeichnungszeit t_n ausgedrückt werden. Das Signal $x(t)$ kann mit $\hat{x}$ als zeitabhängiger Ausgang eines Simulationsmodells

$$\hat{x}_i = \hat{x}\left(i\Delta\hat{t}\right) = \hat{x}\left(\hat{t}_i\right) \tag{6.3}$$

mit der Zykluszeit der Simulation $\Delta\hat{t}$, einer Anzahl von I Simulationsschritten, dem positiv ganzzahligen Index $i \in \{0, 1, 2, ..., I-1\}$ und der Simulationszeit t_i beschrieben werden. Die Abtastzeit der Aufzeichnung Δt und die Zykluszeit der Simulation $\Delta\hat{t}$ sind über den Faktor α

$$\Delta t = \alpha\Delta\hat{t} \qquad mit \quad \alpha > 0 \tag{6.4}$$

miteinander verknüpft. Eine Verknüpfung des aufgezeichneten Signals y_n mit dem simulierten Eingang $\hat{x}_i$ ermöglicht dessen konsistente Wiederverwendung. Substituiert man t_n durch x_n erhält man

$$\begin{aligned} y_n &= y\left(n\Delta t\right) = y\left(t_n\right) \\ &= y\left(x^{-1}\left(x_n\right)\right). \end{aligned} \tag{6.5}$$

Der Bezug zwischen x_n und $\hat{x}_i$ erfolgt über eine Interpolation nicht äquidistanter Datenvektoren. Die Indizes i und n werden mit Hilfe der Funktion

$$g(a) = \begin{cases} 1 & for\ 0 \geq a \\ 0 & else \end{cases} \tag{6.6}$$

aufeinander abgebildet. Daraus folgt die Substitution des Simulationsindex i durch den Index der Aufzeichnung n mit

$$n = \sum_{j=0}^{N-1} g(\hat{x}_i - x_j). \tag{6.7}$$

Folglich können die aufgezeichneten Daten y_n auf das simulierte Signal x_i abgebildet werden und der Eingang $\hat{y}_i$ aus der Aufzeichnung entsprechend des Simulationsmodells abgetastet und in das Testobjekt eingespeist werden

$$\hat{y}_i = y_n \qquad mit \quad n = \sum_{j=0}^{N-1} g(\hat{x}_i - x_j). \tag{6.8}$$

Das Signal $\hat{y}_i$ verhält sich damit reaktiv auf den Ausgang der Regelung, bzw. des Testobjekts.

Die zeitabhängigen und ereignisgesteuerten Sequenzen des Signals z_n

Das Signal z_n besteht aus K aufeinanderfolgenden zeitabhängigen und ereignisgesteuerten Sequenzen $z_k(t_{jk})$, die jeweils J_k Datenelemente enthalten. Der Ablauf innerhalb einer Sequenz hängt von der Aufzeichnungszeit t_n ab. Der kausale Zusammenhang zwischen den verschiedenen Sequenzen und den übrigen Signalen basiert auf den jeweiligen Startzeiten t_{0k}, die wiederum auf die Referenz x_n bezogen werden können. Einzelne Sequenzen können durch

$$z_k(t_{jk}) = z(t_{jk} + t_{0k}) \cdot rect\left(\frac{t_{jk}}{J_k \Delta t}\right) \tag{6.9}$$

ausgedrückt werden. Die Rechteckfunktion

$$rect(a) = \begin{cases} 1 & for\ 0 \leq a < 1 \\ 0 & else \end{cases} \tag{6.10}$$

begrenzt dabei die Gültigkeit im Zeitbereich. Für das Beispielsystem ist $z = 0$ der Standardwert, wenn keine Sequenz verfügbar ist. Um die aufgezeichneten Sequenzen

in der Simulationsumgebung wiederzuverwenden müssen alle Abhängigkeiten zur Aufzeichnungszeit t_n durch die Simulationszeit $\hat{t}_i$ ersetzt werden. Zunächst muss die Startzeit der Sequenz t_{0k} auf das Signal $\hat{x}_i$ abgebildet werden. Mit

$$i_{0k} = \sum_{i=0}^{I-1} g\left(x_{0k} - \hat{x}_i\right) \tag{6.11}$$

ergibt sich für die Sequenz-Zeit t_{jk} innerhalb der Sequenzen

$$t_{jk} = \left\lfloor \frac{i - i_{0k}}{\alpha} \right\rfloor \Delta t. \tag{6.12}$$

Ersetzt man t_{jk} in Gleichung 6.9 mit dieser Beziehung und bildet über alle Sequenzen die Summe, kann das Eingangssignal $\hat{z}_i$ aus der Aufzeichnung wie folgt abgeleitet werden.

$$\hat{z}_i = \sum_{k=1}^{K} z\left(\left\lfloor \frac{i - i_{0k}}{\alpha} \right\rfloor \Delta t_r + t_{0k}\right) \cdot rect\left(\frac{1}{J_k}\left\lfloor \frac{i - i_{0k}}{\alpha} \right\rfloor\right) \tag{6.13}$$

Zur Vereinfachung wurde ein Term zur Limitierung auf eine einzige gültige Sequenz weggelassen. Bei der Umsetzung des Ansatzes müssen Vorkehrungen getroffen werden, die die gleichzeitige Existenz von zwei Sequenzen verhindern. Die Sequenzierung des Signals z_n ermöglicht eine Entkopplung der aufgezeichneten Daten vom Zeitstempel der Aufzeichnung. Im Folgenden wird die Wirkkette des PCC-Features entsprechend der beschriebenen kausalen Zusammenhänge unterteilt.

6.1.2 Kausale Zusammenhänge PCC

Die Kernfunktionalität des PCC ist die Regelung der Fahrzeuggeschwindigkeit. Folglich ist die direkte Rückkopplung der Regelstrecke ein Übertragungsglied, das die Stellgrößen der MPC in eine Geschwindigkeit überführt. Die einfache Integration der angeforderten Beschleunigungen stellt bereits ein stark vereinfachtes Modell dieses Übertragungsglieds dar. Um das in Abschnitt 5.1 beschriebene komfort- und energieoptimale Fahrverhalten zu erhalten, berücksichtigt die Regelung den vollständigen Antriebsstrang inklusive der Verbrauchs- und Momentenkennfelder des Motors und der Übersetzungen und Verlustmomente der Getriebe und Differentiale sowie die Längsdynamik des Fahrzeugs mit Reifenschlupf, Luft-, Roll-, Steigungs- und Beschleunigungswiderständen in unterschiedlichen Detaillierungen [218]. Diese

Details sind somit Teil der direkten Rückkopplung und müssen auch im Modell der Regelstrecke modelliert werden. Insbesondere müssen alle Schnittstellen des Antriebsstrangs, die der PCC als Stellgröße nutzt, emuliert werden, wie beispielsweise die Wahl einer Getriebestufe, des Antriebsmoments oder den Start einer Segelphase mit entkoppeltem Antrieb.

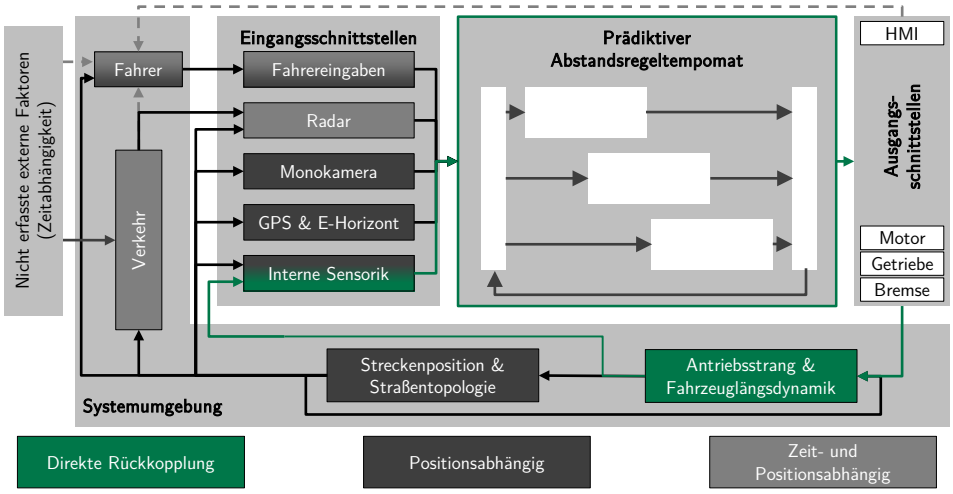


Abbildung 6.4: Die Analyse der kausalen Zusammenhänge in der PCC-Wirkkette führt zur Kategorisierung der Systemumgebung in Elemente mit direkter Rückkopplung, positionsabhängige Elemente sowie zeit- und positionsabhängige Elemente. Die grau gestrichelten Beziehungen sind auf externe Einflüsse und unbekannte Rahmenbedingungen zurück zu führen, die in den aufgezeichneten Daten nicht enthalten sind. Sie werden durch die Zeitabhängigkeit substituiert.

Abbildung 6.4 zeigt die kausale Analyse der PCC-Wirkkette und das Element Antriebsstrang & Fahrzuglängsdynamik innerhalb der direkten Rückkopplung. Die Beschleunigungskraft des Fahrzeugs ergibt sich aus der Differenz zwischen der Antriebskraft des Motors und den Fahrwiderstandskräften. Während der Luftwiderstand stark von der gefahrenen Geschwindigkeit abhängt, sind der Roll- und insbesondere der Steigungswiderstand vom Steigungswinkel der Straße abhängig. Dieser ändert sich mit der Streckenposition des Fahrzeugs. Daraus resultiert eine zweite Rückkopplung in der Wirkkette zwischen der Straßentopologie und der Fahrzeuglängsdynamik. Neben der Rückkopplung zur Fahrzeuglängsdynamik geht die Straßensteigung, die im Antriebsstrang ermittelt wird, über den Block interne Sensorik in den PCC ein.

Die über die Simulationszeit integrierte Fahrzeuggeschwindigkeit liefert mit der

absoluten Streckenposition eine eindeutige Referenz zwischen Längsdynamik und Straßentopologie. Diese Referenz ist in Abbildung 6.5 dargestellt und kann als eindeutige Bezugsgröße zwischen der Längsdynamik und den übrigen Systemeingängen genutzt werden.

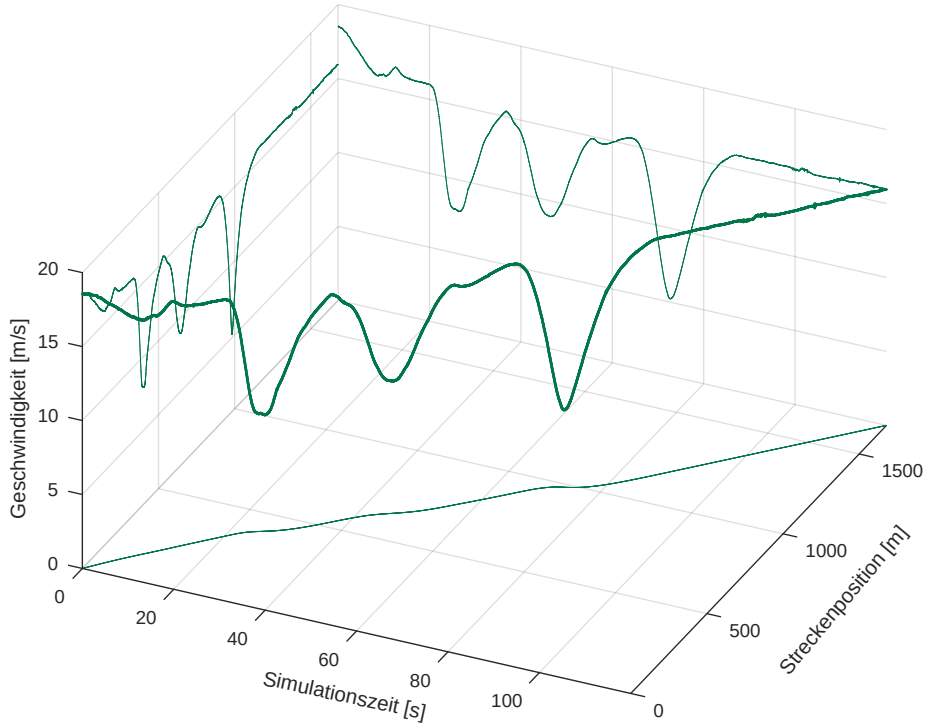


Abbildung 6.5: Die Integration der Fahrzeuggeschwindigkeit über der Simulationszeit liefert mit der absoluten Streckenposition eine eindeutige Referenz zwischen Längsdynamik und Straßentopologie.

Die Lokalisierung des Fahrzeugs in Bezug auf seine Position in der realen Welt basiert auf einem GPS-Empfänger und der digitalen Karte des elektronischen Horizonts. Die prädiktive Streckeninformation wird in Form einer komplexen Nachrichtenstruktur über das Kommunikationsnetz des Fahrzeugs empfangen. Beide Informationsströme sind von der Streckenposition des Fahrzeugs abhängig. Die GPS-Messungen stellen eine Abtastung des kontinuierlichen Verlaufs der Fahrzeugposition dar und die Daten des elektronischen Horizonts liefern den vorausliegenden Ausschnitt der globalen Karte. Dabei können sich aufeinanderfolgende Ausschnitte aufgrund von Abzweigungen oder Fehlern des Kartenabgleichs grundlegend unterscheiden. Darüber hinaus wird die komplexe Datenstruktur des elektronischen

Horizonts über mehrere Nachrichten verteilt übermittelt. Daher hat dieser Datenstrom einen auf die Streckenposition bezogenen, ereignisgesteuerten Charakter.

Das monoskopische Frontkamerasystem erfasst die Verkehrszeichen und Fahrspurmarkierungen des jeweiligen Bildausschnitts (s. Abb. 6.6). Die Verkehrszeichen und Fahrspurmarkierungen gehören zur statischen Umgebung des Fahrzeugs und können ortsfest in Bezug zur Streckenposition referenziert werden. Aus der Abtastung der statischen Umgebung durch das Kamerasystem folgt für den Datenstrom erkannter Verkehrszeichen zunächst ein spontaner, ereignisgesteuerter Charakter. Die erkannten Geschwindigkeitslimits werden im Navigationssystem mit den in der Karte referenzierten Geschwindigkeitslimits abgeglichen und für jedes Datenpaket wird das jeweils gültige Limit übermittelt. Damit stellen die Geschwindigkeitslimits und die erfassten Fahrspurmarkierungen analog zur GPS-Position die Abtastung eines kontinuierlichen Verlaufs dar. Werden keine Verkehrszeichen oder Fahrspurmarkierungen erfasst liegen an der jeweiligen Schnittstelle entsprechend ungültige Werte an. In Abbildung 6.4 wird die Monokamera entsprechend der Analyse als positionsabhängige Eingangsschnittstelle abgebildet.

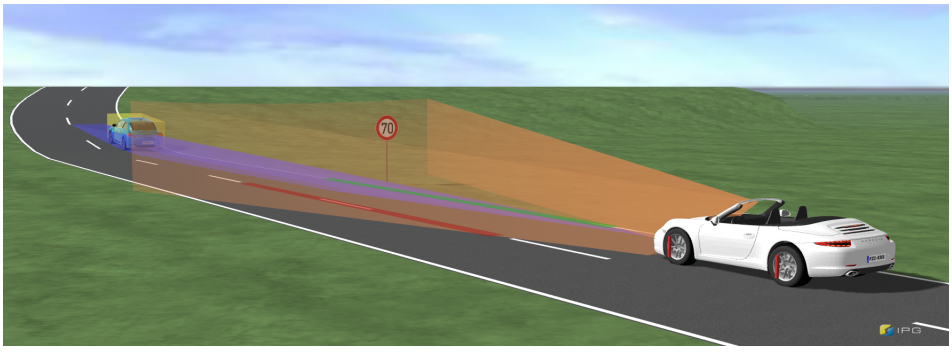


Abbildung 6.6: Das monoskopische Frontkamerasystem (orange) erfasst mit Verkehrszeichen und Fahrspurmarkierungen die statische Umgebung und der Radarsensor des ACC die Objekte der dynamischen Umgebung.

Die Wahrnehmung des dynamischen Umfelds basiert auf dem Radarsensor des ACC (s. Abb. 6.6). Dieser liefert eine Liste detektierter Objekte im Sichtfeld des Sensors. Diese sind den drei Kategorien vorausfahrende Verkehrsobjekte, entgegenkommende Verkehrsobjekte und statische Objekte zugeordnet. Ihre Positionen, Geschwindigkeiten und Beschleunigungen werden als zweidimensionale Referenz relativ zum Egofahrzeug ausgegeben. Das nächstliegende vorausfahrende Verkehrsobjekt, das der Spur des Egofahrzeugs zugeordnet werden kann, wird vom ACC als Zielobjekt ausgewählt und für die adaptive Geschwindigkeitsregelung genutzt. Dieses wird als

Zielobjekt mit eindimensionalen Werten für die relative Distanz, Geschwindigkeit und Beschleunigung ausgegeben.

Die Eingangssignale der dynamischen Objekte sind von der erfassten Position und Bewegung des jeweiligen Objektes und der Position und Bewegung des Egofahrzeugs abhängig. Das jeweilige Verhalten der Verkehrsobjekte in Referenz zu einem globalen, ortsfesten Koordinatensystem hängt von der Streckenposition des Objekts und nicht von in der Aufzeichnung erfassten externen Faktoren ab, wie der Umgebung des Verkehrsobjekts, dessen Fahrer und ggf. aktivierter Assistenzfunktionen. Eine mögliche Beeinflussung des Fahrverhaltens der Verkehrsobjekte durch das Verhalten des Egofahrzeugs wurde als vernachlässigbar bewertet. Da zu keinem der externen Faktoren ergänzende Informationen verfügbar sind, werden sie durch eine Zeitabhängigkeit ersetzt, wie in dem Auschnitt in Abbildung 6.7 zu sehen ist.

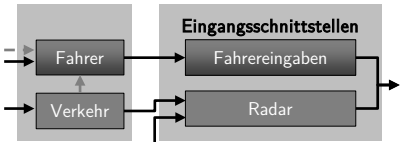


Abbildung 6.7: Ausschnitt der Positions- und Zeitabhängigen Eingänge aus Abbildung 6.4.

Einerseits hängt die Existenz der Objektdetektionen von der Position und dem Verhalten des Egofahrzeugs ab. Gleichzeitig muss aber das zeitliche, auf die absolute Position bezogene Verhalten der Objekte von der Bewegung des Egofahrzeugs entkoppelt werden, um in der Simulation ein reaktives Verhalten der Eingangssignale mit relativem Bezug zu erreichen. Abbildung 6.8 stellt ein abstraktes Überholszenario dar. Die Interpretation des detektierten Objekts als einzelne, zeitlich unabhängige Sequenz, deren Start von einer spezifischen Streckenposition des Egofahrzeugs abhängt ermöglicht diese Entkopplung. Als Startpunkt dient die jeweilige Position zu der ein Objekt zum ersten Mal erfasst wurde. Objekte, die zwischenzeitlich, z.B. aufgrund von Sensorabschattungen an Kurven oder Kuppen, nicht erfasst wurden, generieren mehrere derartige Sequenzen. Diese werden nur verknüpft, wenn diese Zuordnung bereits während der Aufzeichnung durch die Objektverfolgung des ACC getroffen wurde.

Das Verhalten des Fahrers und entsprechend die Fahrereingaben (s. Abb. 6.7) hängen von der Streckenposition, dem Verkehr, dem HMI des Fahrzeugs und nicht erfassten externen Einflussfaktoren ab. Die Untersuchung der Usability des PCC ist nicht Teil der Zielsetzung des vorgestellten Ansatzes, daher wird die Rückkopplung über das HMI größtenteils vernachlässigt und alle Bedienhandlungen

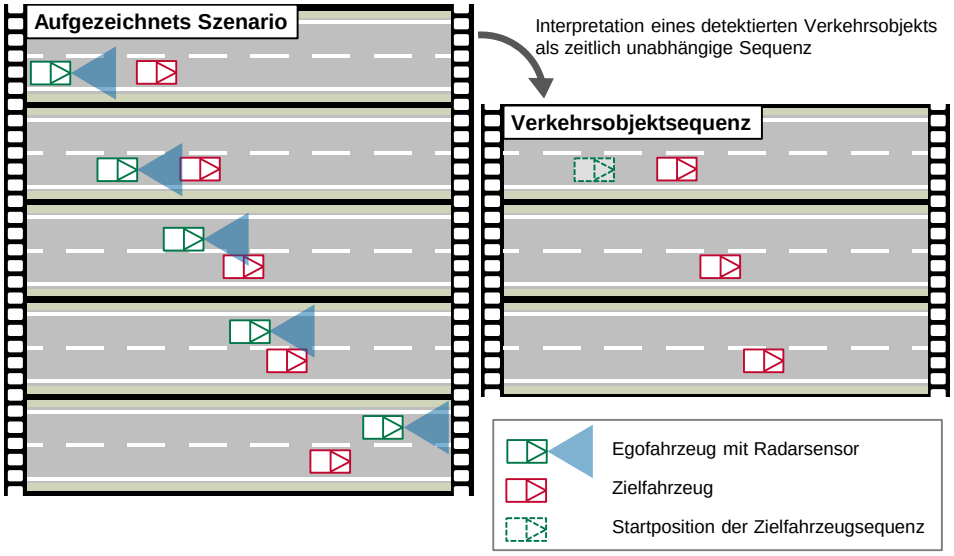


Abbildung 6.8: Abstrakte Darstellung eines Überholszenarios (linke Spalte) und die Interpretation des detektierten Objekts als einzelne, zeitlich unabhängige Sequenz (rechte Spalte).

am HMI des PCC werden auf die Streckenposition bezogen. Die Abhängigkeit von der Verkehrsumgebung hängt von der Persönlichkeit und Tagesverfassung des Fahrers ab. Da hierzu keine Informationen zur Verfügung stehen, wird diese Abhängigkeit über die nicht erfassten externen Faktoren abstrahiert und durch eine Zeitabhängigkeit ersetzt.

Der vom Fahrer über den Lenkradwinkel geregelte Lenkwinkel hängt makroskopisch von der Straßentopologie an der jeweiligen Streckenposition ab. Die konkrete Realisierung durch den Fahrer basiert auf mikroskopischen Details, die als Sensorrauschen interpretiert werden. Der gemessene Lenkradwinkel wird als kontinuierliches, positionsabhängiges Signal interpretiert. Abbildung 6.9 stellt den Bezug zwischen Aufzeichnungszeit und Streckenposition für die einzelnen Datenpunkte abstrakt dar. Neben dem Lenkradwinkel sind für das PCC-Feature die Fahrereingaben über die Pedalerie relevant. Die Betätigung der Bremse führt zur Deaktivierung der Regelung und das Gaspedal ermöglicht eine Übersteuerung durch den Fahrer. Bremse und Gaspedal stellen wie die Betätigung des Blinkers für die PCC-Wirkkette ereignisgesteuerte, positionsabhängige Signale dar.

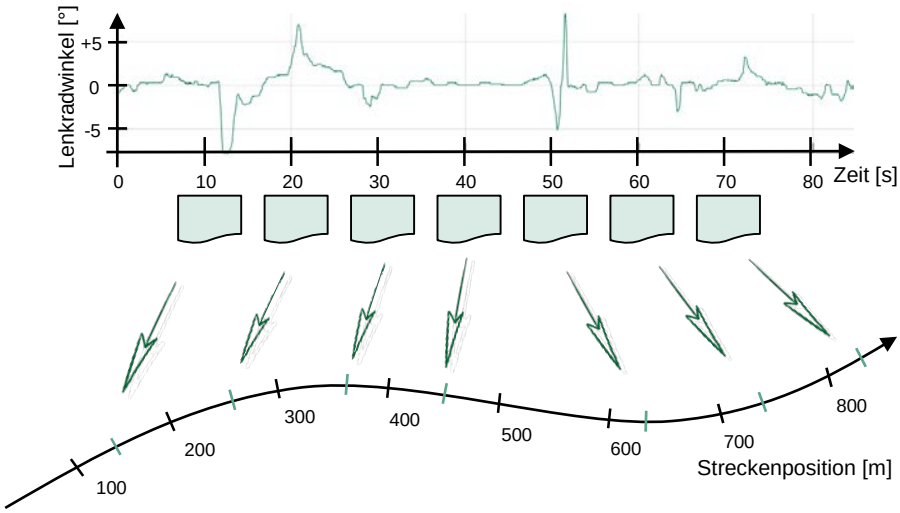


Abbildung 6.9: Abstrahierte Darstellung des Bezugs zwischen Aufzeichnungszeit und Streckenposition am Beispiel des vom Fahrer geregelten Lenkradwinkels.

6.2 Adaptive Wiedergabe der Messdaten

Durch die Analyse der kausalen Zusammenhänge kann das System in Elemente, die der grundlegenden Regelschleife zugehören, und Elemente, die von dieser abhängen, unterteilt werden. Letztere unterscheiden sich weiter in Elemente, deren Verhalten kontinuierlich oder ereignisgesteuert von der Position abhängt, und Elemente, deren Verhalten durch positionsabhängig ereignisgesteuerte Sequenzen mit zeitkontinuierlichem Verlauf beschreibbar ist.

Während Erprobungen werden die Botschaften und Signale der Fahrzeugkommunikation zeitsynchron aufgezeichnet. Änderungen in der Fahrzeuggeschwindigkeit führen in Bezug auf die Position zu variierenden Abtastintervallen der positionsabhängigen Signale. Die Position als Bezugsreferenz wird daher bei der Aufzeichnung dieser Signale durch die Aufzeichnungszeit als Referenz substituiert.

Im Folgenden wird für die drei identifizierten Abhängigkeiten dargestellt, wie der kausale Zusammenhang bei einer an unterschiedliche Fahrzeuggeschwindigkeiten adaptierten Wiedergabe der Messdaten rekonstruiert werden kann. Zur Veranschaulichung des Vorgehens wurde aus einer aufgezeichneten Testfahrt der in Abbildung 6.10 dargestellte Ausschnitt von ca. 120s Dauer und ca. 1700m Länge ausgewählt.

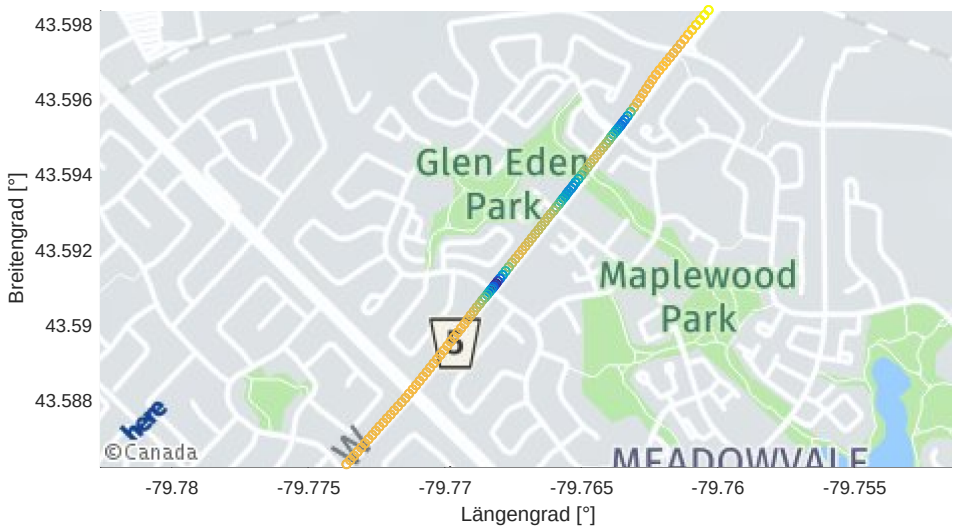


Abbildung 6.10: GPS-Position (rot) des Fahrzeugs für den Ausschnitt einer Testfahrt von 120s Dauer und 1700m Länge. Die Kreise sind entsprechend der Fahrzeuggeschwindigkeit eingefärbt (blau = 0m/s, gelb = 19m/s).

Bei einer Simulation des PCC-Features wird eine neue Fahrzeuggeschwindigkeit für die aufgezeichnete Strecke realisiert. Für die Annahme eines vollständigen und perfekten Simulationsmodells sowie einer identischen PCC-Software und Parametrisierung würde die simulierte Geschwindigkeit exakt der Geschwindigkeit der Aufzeichnung entsprechen. In der Realität wird die simulierte Geschwindigkeit aufgrund von Vereinfachung und Ungenauigkeiten der Modelle jedoch immer von der aufgezeichneten Geschwindigkeit abweichen.

Abbildung 6.11 stellt drei unterschiedlich skalierte Geschwindigkeitstrajektorien dar, an deren Beispiel zur Veranschaulichung der adaptiven Wiedergabe in den folgenden Abschnitten die Eingangssignale entsprechend der kausalen Abhängigkeiten aus den aufgezeichneten Messdaten rekonstruiert werden. Die Trajektorien wurden aus der Geschwindigkeit der Aufzeichnung abgeleitet. Trajektorie 1 entspricht der Aufzeichnungsgeschwindigkeit und die Trajektorien 2 und 3 wurden aus dieser durch Skalierung mit zwei aus überlagerten Sinuskurven bestehenden Vektoren abgeleitet, um den Verlauf deutlich, jedoch nicht unrealistisch zu verändern.

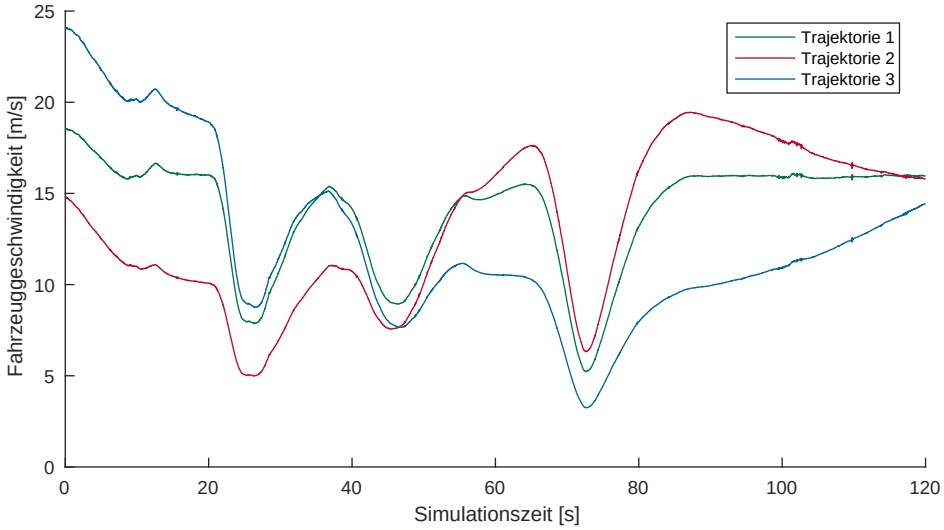


Abbildung 6.11: Drei simulierte Geschwindigkeitstrajektorien an deren Beispiel die adaptive Wiedergabe erläutert wird.

6.2.1 Positionsabhängige Signale

Der durch den Fahrer und ein möglicherweise aktiviertes Assistenzsystem eingesteuerte Lenkradwinkel stellt ein Beispiel für ein Signal mit kontinuierlichem, positionsabhängigem Verhalten dar. Um ein an die Fahrzeuggeschwindigkeit des Simulationsmodells adaptiertes Verhalten zu erzeugen, muss die während der Aufzeichnung ausgetauschte Bezugsreferenz wiederhergestellt werden. Aus der Integration der aufgezeichneten Fahrzeuggeschwindigkeit wird der Verlauf der Streckenposition über der Aufzeichnungszeit berechnet. Diese ergänzt die Zeitreferenz der Abtastungen um die benötigte Positionsreferenz.

Zur Rekonstruktion der kausalen Abhängigkeit werden die aufgezeichneten Daten entsprechend der simulierten Fahrzeugposition entlang der ergänzten Positionsreferenz abgetastet. Die Interpolation des Signals erfolgt im Sinne eines Abtast-Halte-Glieds, d.h. immer der zuletzt überstrichene Messpunkt wird genutzt. Das Vorgehen führt entsprechend der Fahrzeuggeschwindigkeit zu einer Dopplung oder dem Verlust von Informationen. Da die Zykluszeiten von 10-20 ms mit 50-100 Hz deutlich über die Signaldynamik der Längsregelung liegen, stellt dies kein Problem dar.

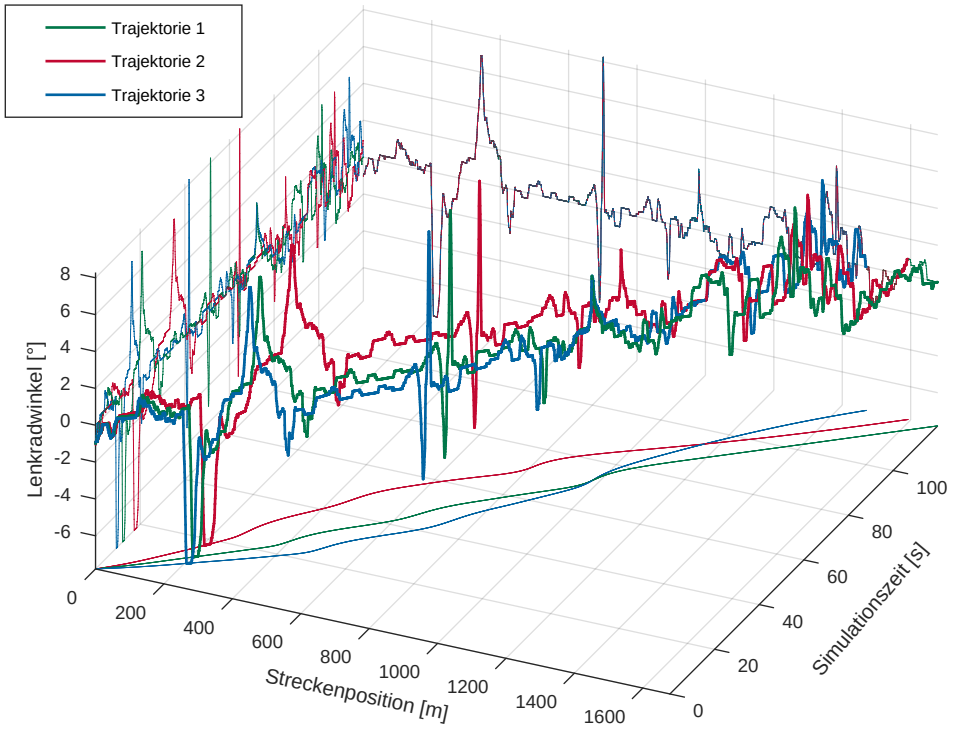


Abbildung 6.12: Darstellung des entsprechend der drei unterschiedlichen Geschwindigkeitstrajektorien abgetasteten Lenkradwinkels bezogen auf die Streckenposition und die Simulationszeit.

Abbildung 6.12 visualisiert den entsprechend drei unterschiedlichen Fahrzeuggeschwindigkeiten abgetasteten Lenkradwinkel bezogen auf die beiden Referenzen Streckenposition und Simulationszeit. Der Verlauf der Streckenposition über die Simulationszeit ist als zweidimensionale Projektion auf die Ebene Streckenposition-Simulationszeit dargestellt. Auf die dazu orthogonalen Flächen wird der Lenkradwinkel über die Streckenposition und über die Simulationszeit projiziert. In Bezug zur Streckenposition ist der Verlauf des Lenkradwinkels für die drei Geschwindigkeiten deckungsgleich, während er in Bezug zur Simulationszeit den gewünschten an die unterschiedlichen Geschwindigkeiten adaptierten Verlauf annimmt.

Ein Beispiel für ein Signal mit ereignisgesteuertem, positionsabhängigem Verhalten sind die durch die Kamera erfassten Geschwindigkeitsbeschränkungen. Das Vorgehen zur Rekonstruktion ereignisgesteuerter Signale unterscheidet sich vom kontinuierlichen Fall durch die Abtastmethode. Während der Rekonstruktion des

Signals mit einer gegenüber der aufgezeichneten Geschwindigkeit höheren Fahrzeuggeschwindigkeit der Simulation gehen Informationen verloren, da einzelne Messpunkte übersprungen werden. Während der Informationsverlust für kontinuierliche Signale aufgrund der sehr hohen Auflösung während der Aufzeichnung minimal ist, kann dies bei ereignisgesteuerten Signalen zum vollständigen Verlust einer Information führen. Die überstrichenen Messpunkte werden nach Daten gefiltert, die vom Null-Wert abweichen, d.h. eine Information enthalten. Diese Messpunkte werden in einen Pufferspeicher geschrieben aus dem für jeden Simulationsschritt ein Element entnommen wird. Enthält der Pufferspeicher keine Elemente gilt der Null-Wert. Damit ist sichergestellt, dass keine Information verloren geht.

6.2.2 Zeit- und Positionsabhängige Signalsequenzen

Aufgrund der direkten Abhängigkeit der positionsabhängigen Signale von der simulierten Streckenposition reicht der oben beschriebene Austausch der Bezugsreferenz für die Rekonstruktion der kausalen Zusammenhänge aus. Die korrelierte Abhängigkeit der mit dem Radar erfassten Verkehrsobjekte von der Streckenposition des Egofahrzeugs und der durch den zeitlichen Verlauf ersetzten externen Faktoren erfordert einen aufwändigeren Ansatz.

Der Zustand der Verkehrsobjekte setzt sich aus der erfassten Position, Geschwindigkeit und Beschleunigung in Relation zur Position, Geschwindigkeit und Beschleunigung des Egofahrzeugs zusammen. Pro Zeitschritt sendet der Radarsensor, bzw. das ACC-Steuergerät für eine feste Anzahl an Objekten ein Datenpaket mit dem erfassten Zustand, das aus mehreren Botschaften auf dem Kommunikationsbus besteht. Werden weniger Objekte erfasst als in der Schnittstellendefinition vorgesehen sind, wird ein Null-Frame gesendet. Die in der Schnittstellenspezifikation festgelegte Anzahl an Objekten stellt damit in den Messdaten einen kontinuierlichen, zeitsynchronen Datenstrom dar. Um die zeitliche Abhängigkeit und die Abhängigkeit von der Streckenposition des Egofahrzeugs zu entkoppeln, muss zunächst der relative Bezug der Verkehrsobjekte aufgelöst werden. Über den absoluten Bezug des Egofahrzeugs auf die aufgezeichnete Testfahrt wird der Zustand der Verkehrsobjekte in eine absolute Streckenposition, Geschwindigkeit und Beschleunigung überführt.

Abbildung 6.13 zeigt die aufgezeichnete Geschwindigkeit des Egofahrzeugs und der absoluten Geschwindigkeit des für die Längsregelung ausgewählten Verkehrsobjekts über der Streckenposition und der Aufzeichnungszeit für den zuvor beschriebenen

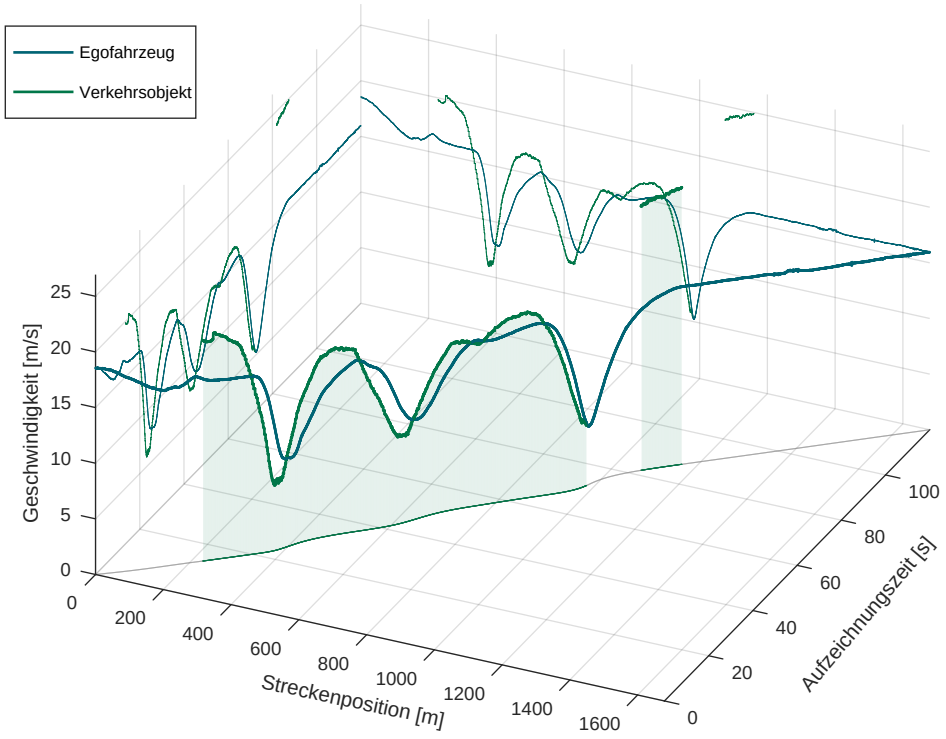


Abbildung 6.13: Aufgezeichnete Geschwindigkeit des Egofahrzeugs und des für die Längsregelung ausgewählten Verkehrsobjekts über die Streckenposition und die Aufzeichnungszeit. Die Projektionen auf die Grundflächen des 3D-Plots stellen die zweidimensionalen Zusammenhänge losgelöst dar.

Messdatenausschnitt. Von 13,5 s bis 72,0 s und 81,8 s bis 87,5 s sind Abschnitte mit einem für die Längsregelung relevanten Verkehrsobjekt in dem Ausschnitt enthalten. Für jeden zusammenhängenden Abschnitt wird zur Rekonstruktion der kausalen Zusammenhänge eine Sequenz mit unabhängiger Zeitreferenz abgeleitet, die über die Streckenposition des Egofahrzeugs zum Zeitpunkt der ersten Detektion des Objekts verknüpft ist.

Die unabhängige Zeitreferenz der abgeleiteten Sequenzen erhält das Verhalten des jeweiligen Objekts bei einer Änderung des Verhaltens des Egoobjekts. Abbildung 6.14 stellt die Geschwindigkeit des Verkehrsobjekts über der Streckenposition des Egofahrzeugs und der entkoppelten Sequenz-Zeit für jede der drei Trajektorien dar. Die Streckenposition zu Beginn der Sequenz ist für alle drei Trajektorien identisch und entwickelt sich mit der Sequenz-Zeit entsprechend der jeweiligen

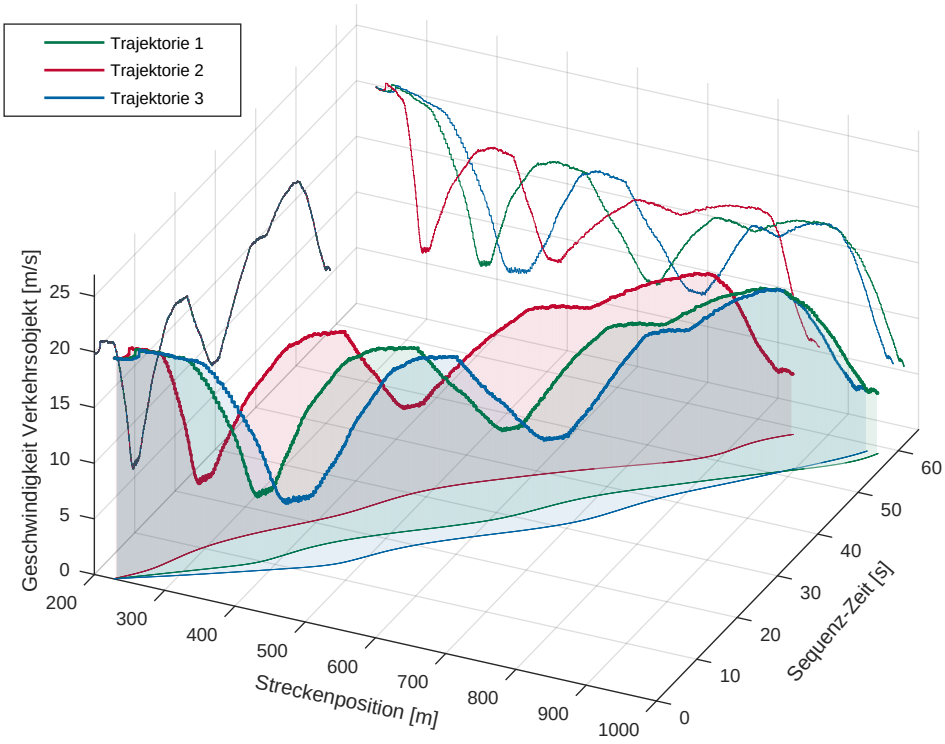


Abbildung 6.14: Vidualisierung der Geschwindigkeit des Verkehrsobjekts der ersten Sequenz über der Streckenposition des Egofahrzeugs und der entkoppelten Sequenz-Zeit für jede der drei Trajektorien.

Geschwindigkeit. Auf der Zeit-Geschwindigkeit-Ebene ist das zeitlich identische Verhalten des Verkehrsobjekts für alle drei Trajektorien zu erkennen, während die Verläufe auf der Position-Geschwindigkeit-Ebene entsprechend der unterschiedlichen Geschwindigkeiten und der daraus folgenden Streckenposition des Egofahrzeugs variieren. Zur Einbettung der Verkehrsobjekte in Bezug auf die globale Zeitreferenz der Simulation werden die Sequenzen durch eine Überschreitung der Startposition durch das Egofahrzeug ausgelöst und entsprechend des daraus resultierenden Zeitversatzes mit dem globalen Datenstrom synchronisiert.

Abbildung 6.15 stellt den rekonstruierten Geschwindigkeitsverlauf für die beiden Verkehrsobjektsequenzen eingebettet in die globale Simulationszeit und Streckenposition der drei realisierten Geschwindigkeitstrajektorien dar. Auf der Zeit-Geschwindigkeit-Ebene ist zu sehen, dass das Verhalten der Verkehrsobjekte um einen Versatz verschoben, zeitlich identisch ist. Der Versatz entspricht dem Zeitun-

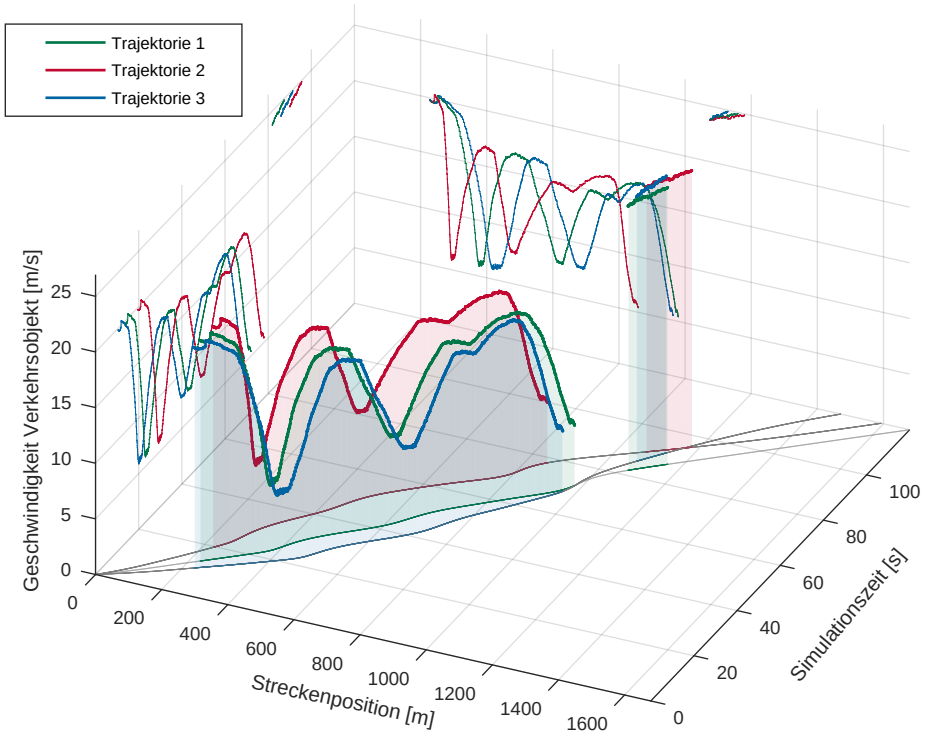


Abbildung 6.15: Drei den Geschwindigkeitstrajektorien entsprechende Realisierungen der in Abbildung 6.13 visualisierten Aufzeichnung. Das zeitliche Verhalten des Zielfahrzeugs ist für alle drei Trajektorien identisch, aber um den zeitlichen Offset verschoben, den das Egofahrzeug bis zum Erreichen der Triggerposition benötigt.

terschied, den das simulierte Egofahrzeug der jeweiligen Trajektorie benötigt um die Startposition der Sequenz zu erreichen. Die von der Simulationszeit losgelöste Wiedergabe als Sequenz erhält damit das individuelle Verhalten der Verkehrsobjekte, während deren Auftreten über die Streckenposition mit den übrigen Eingangssignalen synchronisiert ist.

6.3 Reactive-Replay Simulation

Mit dem vorgestellten Ansatz zur adaptiven Wiedergabe der Messdaten wurde die in Abschnitt 5.4.1 beschriebene Simulationsumgebung für den Test des PCC erweitert.

Die Erweiterung bietet den Entwicklern die Möglichkeit, während der Erprobung in der realen Welt aufgezeichnete Situationen und Szenarien am Schreibtisch detailliert zu analysieren und mit Änderungen an der Implementierung oder Parametrierung wiederholt auszuführen. Die proprietär entwickelte Simulationsumgebung ermöglicht es den Entwicklern darüber hinaus, benötigte Anpassungen und Erweiterungen innerhalb ihrer gewohnten Entwicklungsumgebung umzusetzen.

6.3.1 Implementiertes Gesamtkonzept

Das implementierte Gesamtkonzept (Abb. 6.16), das die entwickelte Methode umsetzt, besteht aus einer Komponente zur Vorverarbeitung der Aufzeichnungen, der Testumgebung für die Simulation der PCC-Software auf Subsystemebene und verschiedenen Modulen zur Bewertung und Analyse. Im Rahmen dieser Dissertation wurde die bestehende Längsdynamiksimulationsumgebung (vergl. Abschnitt 5.4.1 um die im Folgenden beschriebene Reactive-Replay Engine erweitert. Die Umgebung ist modular in C++ implementiert und kann auf handelsüblichen Desktopsystemen mit Windows-Betriebssystem ausgeführt werden.

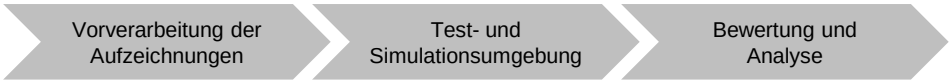


Abbildung 6.16: Umgesetztes Gesamtkonzept zur Anwendung der Reactive-Replay Methode.

Zur Vorverarbeitung der Messdaten werden die Signale der Eingangsschnittstellen entsprechend der ermittelten kausalen Abhängigkeiten gruppiert. Die Werte der positionsabhängigen Eingänge werden für jeden Zeitpunkt in je einem Datenframe für kontinuierliche und ereignisgesteuerte Signale zusammengefasst. Jeder dieser Frames wird dabei mit der zugehörigen Zeit- und Positionsreferenz versehen. Die einzelnen Nachrichten einer zusammengehörenden Datenstruktur des elektronischen Horizonts werden als einzelne ereignisgesteuerte Frames abgelegt. Die Eingangsschnittstelle der Software-Einheit zur Rekonstruktion der prädiktiven Streckendaten ist robust und kann auch die dadurch nicht äquidistant über die Zeit verteilten Einzelnachrichten einer zusammenhängenden Struktur verarbeiten.

Wie bereits beschrieben, wird in den Botschaften des Radarsensors eine feste Anzahl an detektierten Verkehrsobjekten übertragen. Aus den aufgezeichneten Daten werden zur Vorverarbeitung zunächst die enthaltenen Objektsequenzen extrahiert und mit der jeweiligen Startposition versehen. Alle Sequenzen werden

in aufsteigender Reihenfolge der Startposition in einer Liste abgelegt. Neben der Startposition beinhalten die Sequenzen die Startzeit und die Anzahl der Frames sowie eine Liste der zugehörigen Frames. Für jede Sequenz werden die in den Botschaften enthaltenen relativen Zustände in Abhängigkeit vom Zustand des Egofahrzeugs auf die globale Referenz abgebildet. Unsicherheiten der Lokalisierung des Egofahrzeugs werden dabei auf die Objektzustände übertragen. Diese Frames werden mit dem zugehörigen Zeitstempel, der Streckenposition des Egofahrzeugs zum Zeitpunkt der Aufzeichnung und der Position des Objekts in der festen Datenstruktur des Radarsensors in der Frameliste der Sequenz abgelegt.

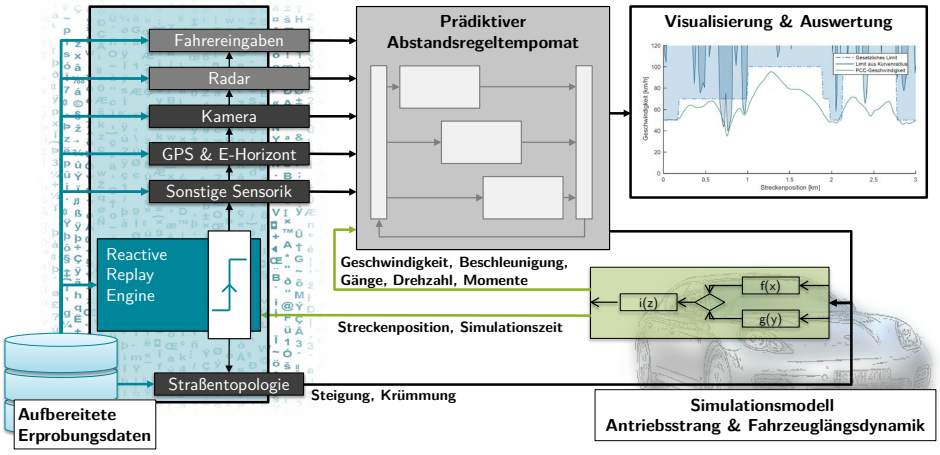


Abbildung 6.17: Testumgebung für die Simulation und den Test der PCC-Software auf Subsystemebene bestehend aus einem Simulationsmodell des Antriebsstrang und der Fahrzeuglängsdynamik, der Reactive-Replay Engine, dem Testobjekt und einem Block für die Visualisierung und Auswertung.

Abbildung 6.17 stellt den konzeptuellen Aufbau der Test- und Simulationsumgebung der PCC-Software mit den vier für dynamische Tests und Analysen relevanten Modulen dar. Neben dem Testobjekt beinhaltet die Testumgebung das Simulationsmodell des Antriebsstrangs und der Fahrzeuglängsdynamik, die Reactive-Replay Engine zur Rekonstruktion der kausal zusammengehörigen Daten und eine Visualisierung zur Analyse und Bewertung des Regelungsverhaltens der MPC. Im Simulationsmodell werden die Werte der grundlegenden Regelschleife berechnet. Die zur Berechnung des Roll- und Steigungswiderstands benötigte Steigung und Krümmung der Straße liefert die Reactive-Replay Engine aus den aufgezeichneten und vorverarbeiteten positionsabhängigen Signalen.

Neben der Position, Geschwindigkeit und Beschleunigung des Fahrzeugs stellt das

Simulationsmodell den Getriebezustand und die von verschiedenen Funktionselementen des Antriebsstrangs geschätzten oder gemessenen Werte für Drehzahlen und Momente zur Verfügung. Die übrigen Eingangsschnittstellen werden mit Ausnahme der Schnittstelle zur Systemaktivierung durch die Reactive-Replay Engine bedient. Das Aktivierungssignal des HMI wird mit einer festen Zeitverzögerung als Antwort auf eine durch das PCC signalisierte Systemverfügbarkeit gesendet.

6.3.2 Reactive-Replay Engine

Die Reactive-Replay Engine übernimmt die Datenhaltung und die Rekonstruktion der kausalen Zusammenhänge während der adaptiven Wiedergabe der aufgezeichneten Daten. Ein Aufruf der Modulschnittstelle mit der aktuellen Fahrzeugposition und dem Zeitstempel der Simulation gibt die rekonstruierten Daten der Eingangsschnittstelle zurück. Das Modul ist als dynamische Programmbibliothek in die bestehende Simulationsumgebung integriert.

Die positionsabhängigen Daten werden über ihre Positionsreferenz abgetastet und die Ausgabestruktur übernommen oder im Fall von ereignisgesteuerten Signalen zunächst in den Zwischenspeicher geschrieben. Aus diesem Speicher wird für jeden Aufruf der Reactive-Replay Schnittstelle ein Element übernommen. Die Daten bleiben inhaltlich unverändert, d.h. es wird keine Interpolation über die Streckenposition durchgeführt. Wie in Abschnitt 6.2.1 beschrieben wurde, ist der dadurch eingebrachte Fehler vernachlässigbar.

Die Verkehrsobjektsequenzen werden entsprechend ihrer Startposition aktiviert und für jeden Zeitschritt wird ein Objektframe aus der Sequenz entnommen. Es wird an der während der Vorverarbeitung gespeicherten Position in die Botschaftsstruktur des ACC-Sensors eingefügt. Bewegt sich das Fahrzeug in der Simulation schneller als zur Aufzeichnungszeit, kann dies dazu führen, dass mehrere Objekte derselben Botschaftsposition aktiv sind. In diesem Fall werden immer die Frames der zuletzt aktivierten Sequenz genutzt.

Die Rekonstruktion der zweidimensionalen Position, Geschwindigkeit und Beschleunigung in relativem Bezug zum Zustand des Egofahrzeugs stellt eine gewisse Herausforderung dar. Das Simulationsmodell beschreibt lediglich die Längsdynamik des Fahrzeugs, daher sind die zweidimensionale Position und die Orientierung des Egofahrzeugs nicht verfügbar. Anstelle einer simulierten Position und Orientierung

werden diese über die zurückgelegte Streckenposition aus den Streckendaten des elektronischen Horizonts gewonnen.

6.4 Evaluation und Bewertung

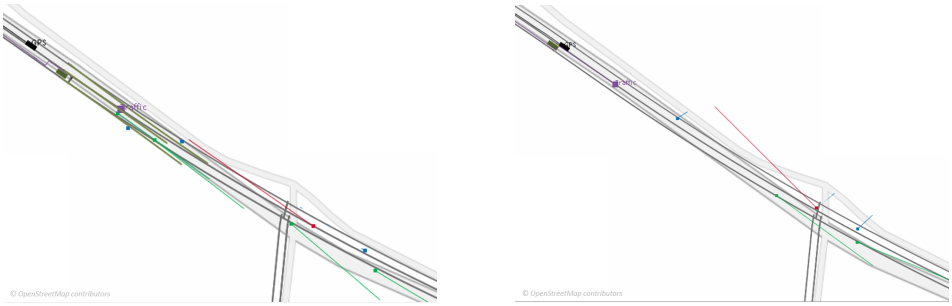
Die Reactive-Replay Methode ermöglicht die Wiederverwendung aufgezeichneter Testfahrten für den virtuellen Test regelungsbasierter Fahrzeugfunktionen. Die adaptive Wiedergabe der Daten zur Rekonstruktion der kausalen Zusammenhänge ermöglicht einen unmittelbaren Übergang von der Erprobung in die Simulation. Im öffentlichen Straßenverkehr erlebte Situationen können durch den Einsatz der Reactive-Replay Methode ohne zeitaufwändige manuelle Parametrierung von Simulationsszenarien reproduzierbar in eine virtuelle Testumgebung überführt werden.

Aufgrund der hohen Automatisierung unterstützt der Ansatz den funktionalen Entwurf, das Debugging und den Test der implementierten Software sowie eine initiale Validierung neuer Funktionskonzepte und -alternativen auf eine sehr effiziente, zeitsparende Art und Weise. Anstelle der manuellen Spezifikation aller Aspekte der konkreten Testszenarien werden diese direkt aus der realen Welt abgeleitet. Die nahtlose und durchgängige Vorgehensweise ermöglicht die Wiederholung beliebiger während einer Testfahrt erlebter Situationen und Szenarien in der Simulation. Auf die Modellierung eines großen Teils der Funktionselemente, insbesondere der physikalischen Sensormodelle und ihres spezifischen funktionalen Verhaltens, kann verzichtet werden. Alle Signale sind bereits in ihrer benötigten Form inklusive möglicher Unsicherheiten enthalten. Darüber hinaus müssen die Testszenarien nicht validiert werden, da sie direkt aus der Realität abgeleitet wurden. Sie beinhalten bereits die Merkmale und regionalen Besonderheiten mit denen der PCC unter Serienbedingungen umgehen muss.

Während dieser Ansatz insbesondere für die Vorentwicklung eine leistungsfähige Möglichkeit darstellt, gibt es klare Beschränkungen. Nur der Antriebsstrang und die Längsdynamik des Fahrzeugs werden simuliert und ermöglichen eine vollständige Rückkopplung. Alle anderen Eingangssignale werden lediglich adaptiv wiedergegeben. Zum Beispiel reagieren die wiedergegebenen Verkehrsobjekte nicht auf zu dichtes Auffahren und die angepasste Wiedergabe beeinflusst die Signalcharakteristik des Lenkradwinkels, da sich dessen Winkelgeschwindigkeit ändert. Für den letzten Fall werden die Daten ohne Anpassung der Werte inkonsistent, wenn

z.B. die Winkelgeschwindigkeit selbst Teil der Eingangsschnittstelle ist. Derartige Abhängigkeiten müssen bereits während der Analyse der kausalen Zusammenhänge identifiziert werden und entsprechend durch Modelle ersetzt werden.

Die Auswirkungen der Nachteile nehmen mit dem Unterschied zwischen aufgezeichneter und simulierter Fahrzeuggeschwindigkeit zu. Neben unerwünschten Änderungen in der Signaldynamik, wirkt eine deutlich geringere Fahrzeuggeschwindigkeit in der Simulation wie ein Tiefpassfilter und eine deutlich höhere Geschwindigkeit führt trotz der ursprünglich hohen Abtastraten zu Informationsverlusten. Um diesem Effekt entgegenzuwirken wird der simulierte Fahrzeugzustand auf den Fahrzeugzustand der Aufzeichnung zurückgesetzt, falls der Unterschied eine definierte Schwelle überschreitet. Die Höhe dieser Schwelle hängt dabei vom individuellen Kontext der untersuchten Fragestellung, bzw. der Nutzung der Reactive-Replay Simulation ab. So können bei der Analyse der Wirkung und Zusammenhänge von Parameteränderungen beispielsweise größere Freiräume erwünscht sein als bei der Durchführung von Regressionstests.



(a) Aufgezeichnete Szene **(b)** Rekonstruierte Szene

Abbildung 6.18: Vergleich der Position und Bewegung der aufgezeichneten und rekonstruierten Verkehrsobjekte

Die Rekonstruktion der zweidimensionalen Position, Geschwindigkeit und Beschleunigung in relativem Bezug zum Zustand des Egofahrzeugs stellt eine zentrale Herausforderung dar. Bei der Rücktransformation der Objekte von ihrem globalen in den relativen Bezug führen bei der Vorverarbeitung eingebrachte Fehler zu Abweichungen zwischen der aufgezeichneten und der rekonstruierten Situation. Dies schlägt sich insbesondere in der Bewegungsrichtung der Objekte nieder. Abbildung 6.18 stellt diese Abweichung zwischen der aufgezeichneten und der rekonstruierten Szene dar. Beispielsweise sind in der aufgezeichneten Szene mehr Objekte enthalten als in der Rekonstruktion. Dies liegt an einer geringeren Geschwindigkeit des Egofahrzeugs in der Simulation gegenüber der Aufzeichnung.

Trotz dieser Nachteile ermöglicht die Reactive-Replay Methode die effiziente Nutzung einer aussagekräftigen Stichprobe der in Kapitel 5 beschriebenen Variantenvielfalt für den Test der PCC-Wirkkette. Am Beispiel der Straßenmodelle, die in konventionellen Gesamtfahrzeugsimulationen erforderlich sind, wird dieser Vorteil deutlich. Allein die Länge des deutschen Straßennetzes[2] beträgt ca. 626.000 km. Um nur ein Zehntel des deutschen Straßennetzes abzudecken, bedarf es einer detaillierten Modellierung von sechzigtausend Kilometern Straße und der anschließenden Parametrierungen der Verkehrsobjekte. Während derartige Umfänge mit dem konventionellen Modellierungsansatz eine enorme Anstrengung bedeuten, sind solche Distanzen im Bereich von Erprobungskampagnen üblich und damit durch den Reactive-Replay Ansatz abgedeckt.

Der hier vorgestellte Ansatz wurde und wird auf täglicher Basis für die Entwicklung des PCC-Features genutzt. Es steht bereits eine breite Datenbasis aufgezeichneter Testszenarien mit den darin enthaltenen Verkehrssituationen zur Verfügung. Die Methode ist für sich wiederholende Tests besonders geeignet und ermöglicht die Sondierung unterschiedlicher Funktionsparameter. Über einen Zeitraum von ca. zwei Jahren haben sich dabei folgende Erfahrungen im Einsatz der Methode ergeben:

- Der Aufwand für operative Testaktivitäten auf der Straße wird signifikant reduziert.
- Der Ansatz ermöglicht eine Analyse von Parameteränderungen auf einer sehr breiten Stichprobe (beispielsweise kann eine Erprobungskampagne von zehn Tagen Länge für unterschiedliche Applikationsparameter innerhalb weniger Stunden mehrfach wiederholt werden).
- Die in der realen Welt gemachten Erfahrungen und erlebten Situationen im Umgang mit dem PCC-Feature werden in den Daten wiederverwendbar konserviert.
- Unterschiedliche Softwarestände können auf Systemebene in vielfältigen Situationen miteinander verglichen werden.

[2] Straßenerhaltungsplanung, Bundesanstalt für Straßenwesen, http://www.bast.de/DE/Strassenbau/Fachthemen/gs3-strassenerhaltungsplanung.html?nn=605006

6.5 Abgrenzung zu vergleichbaren aktuellen Forschungsansätzen

Ein vergleichbarer Ansatz wird durch Zofka et al. [264] zur Generierung kritischer Verkehrsszenarien beschrieben. Im realen Straßenverkehr mit einem Lidar-Sensor aufgezeichnete Verkehrsobjektsequenzen werden manuell aus den aufgezeichneten Daten extrahiert und innerhalb einer Simulationsumgebung wiedergegeben. Dabei wird deren Position angepasst um aus einem ursprünglich sicheren Szenario ein kritisches Szenario mit realem Objektverhalten zu erzeugen. Diesem Ansatz liegt ein ähnliches Prinzip zugrunde, er ermöglicht jedoch keine vollständige Rekonstruktion einer aufgezeichneten Testfahrt und beinhaltet hohe manuelle Aufwände.

Bojarski et al. [265] beschreiben in ihrem Blogpost „End-to-End Deep Learning for Self-Driving Cars“ die Nutzung eines Neuronalen Netzes, das direkt aus dem Pixelbild einer Frontkamera die Ansteuerung der Lenkung ableitet. Es wird beschrieben, dass für das Training des Netzes 72 Stunden Daten bei „klarem und bewölktem Wetter, Nebel, Regen und Schneefall sowie bei Tag und bei Nacht gesammelt wurden“. Um das Netz in einer simulierten Umgebung zu testen, wird das querdynamische Verhalten des Fahrzeugs simuliert und das zugehörige Videobild aus den aufgezeichneten Daten abgeleitet. Dies ist möglich, da ein größerer Bildausschnitt aufgezeichnet wurde als in das Netz eingespeist wird. Weicht die simulierte Querposition des Fahrzeugs von der aufgezeichneten Position ab, wird dieser Ausschnitt entsprechend verschoben. Übersteigt die Abweichung einen Abstand von 1 m, wird die simulierte Fahrzeugposition auf die aufgezeichnete Position zurückgesetzt. Dieser Ansatz kombiniert ein Fahrzeugsimulationsmodell und aufgezeichnete Daten in ähnlicher Weise zum oben vorgestellten Reactive-Replay. Im Gegensatz zum Reactive-Replay Ansatz werden lediglich Videodaten in angepassten Ausschnitten zeitsynchron wiedergegeben. Eine genauere Untersuchung und Berücksichtigung der kausalen Abhängigkeiten zwischen Egofahrzeug und Umgebung ist nicht beinhaltet.

7 Systematische messdatengestützte Absicherung mittels Reactive-Replay

Die im vorhergehenden Kapitel beschriebene Reactive-Replay Methode ermöglicht mit der Wiederverwendung aufgezeichneter fahrzeuginterner Kommunikationsdaten innerhalb der Simulationsumgebung den Test des PCC-Features anhand von in der Realität aufgetretenen Situationen und Szenarien. Der direkte Übergang zwischen Realität und Simulation unterstützt insbesondere die funktionale Absicherung in der Phase des detaillierten Softwareentwurfs und der Implementierung (SWE.3, siehe Abschnitt 2.7) durch die Verbindung der Entwicklungsansätze Rapid Prototyping (siehe Abschnitt 3.3.3) und simulationsgestützte Entwicklung (siehe Abschnitt 3.3.4). Während dieser Phase dient die Simulationsumgebung primär als Entwicklungs- und Debug-Werkzeug, das den Entwicklern ein schnelles Feedback bezüglich umgesetzter Implementierungen und Änderungen liefert. Die Analyse und Bewertung des Simulationsergebnisses erfolgt dabei manuell durch den jeweiligen Entwickler. Eine Verwendung der Simulationsumgebung als Testwerkzeug setzt eine automatisierte Bewertung des Simulationsergebnisses und ein systematisches Vorgehen bei der Auswahl der genutzten Messdaten voraus.

Der folgende Abschnitt 7.1 beschreibt das Vorgehen und die implementierten Werkzeuge zur Verwaltung der Erprobungsdaten und der Ableitung von Testszenarien. In Abschnitt 7.2 werden zwei Konzepte und deren Umsetzung zur automatisierten Bewertung der Simulationsergebnisse virtuell reproduzierter Testfahrten vorgestellt. Der erste Ansatz ermöglicht die Nutzung der Simulationsumgebung für Regressionstests auf Subsystemebene. Der zweite Ansatz ermöglicht die Prüfung eines virtuellen Fahrmanöverkatalogs und in einer erweiterten Form die kontinuierliche Prüfung der Subsystem-Anforderungen des PCC-Features. Das Kapitel schließt in Abschnitt 7.3 mit der Vorstellung einer Methode zur systematischen Auswahl einer repräsentativen Untermenge von Testszenarien ab. Teile dieses Kapitels wurden bereits im Rahmen von Konferenzbeiträgen veröffentlicht [JB4, JB5].

7.1 Verwaltung der Erprobungsdaten und Ableitung von Testszenarien

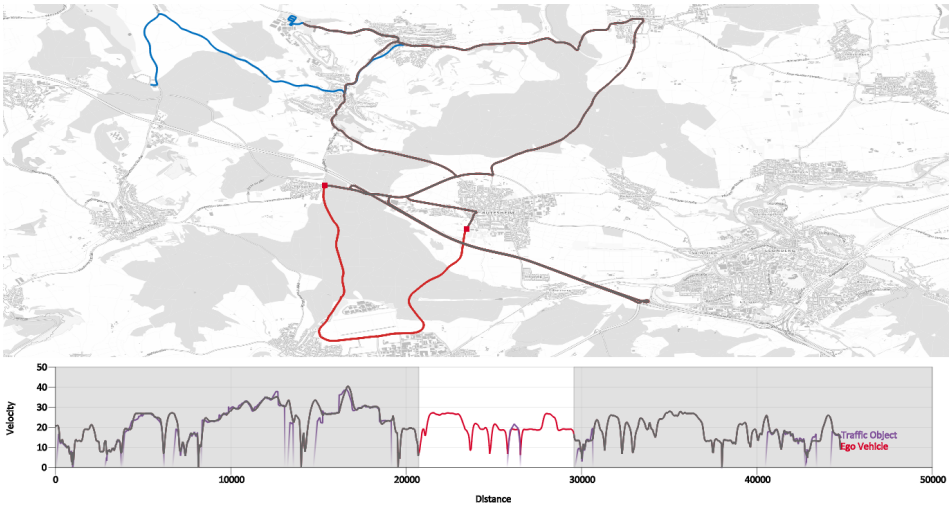
Die in den Erprobungen aufgezeichneten Szenarien werden entsprechend des Kontexts ihrer Aufzeichnung in Ordnern des Filesystems gruppiert abgelegt. Beispielsweise werden die einzelnen Dateien einer Länderkampagne zusammengefasst abgelegt. Sie werden zur Nutzung in der Reactive-Replay Simulation aufbereitet, d.h. die Streckenposition wird ergänzt und enthaltene Sequenzen werden extrahiert. Eine vorverarbeitete Aufzeichnung wird mit Metainformationen, wie Start- und Endposition ergänzt, um ein Testszenario darzustellen.

7.1.1 Manuelle Analyse und Auswahl

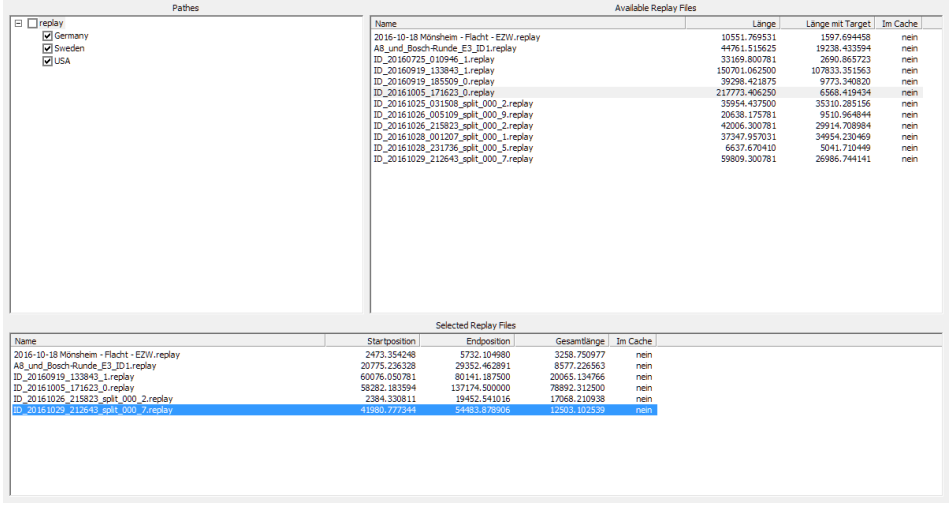
Für die Auswahl der Testszenarien wurde das in Abbildung 7.1 abgebildete Werkzeug implementiert. Die in Ordnern gruppierten Erprobungsdaten werden in das Werkzeug geladen und können über eine Kartenansicht (Abbildung 7.1a) und eine Listenansicht (Abbildung 7.1b) ausgewählt werden. In der Kartenansicht kann die Start- und Endposition des ausgewählten Testszenarios festgelegt werden. Dies wird durch eine Visualisierung der beinhalteten Signale im unteren Drittel der Kartenansicht unterstützt. Im Beispiel ist die Geschwindigkeit des Egofahrzeugs und enthaltener Verkehrsobjekte dargestellt. In der Listenansicht können die Dateien einzelner Ordner ein- und ausgeblendet werden. Ausgewählte Testszenarien können zu Testsets gruppiert werden (Liste „Selected Replay Files“ im unteren Drittel von Abbildung 7.1b), die gesammelt simuliert und weiterverarbeitet werden können.

7.1.2 Automatisierte Auswahl und Filterung

Manuell zu implementierende Filter unterstützen den Nutzer bei der Suche und Auswahl spezifischer Testszenarien. So können die Dateien der geladenen Ordner nach Sequenzen innerhalb bestimmter Wertebereiche (z.B. Geschwindigkeit > 20 m/s) oder bestimmten Metainformationen (z.B. Straßenklasse) durchsucht werden. Die Startpositionen können dabei um einen definierten Abstand vorgezogen werden, um der Regelung genügend Zeit zum Einschwingen zu geben. Die Zusammenstellung des finalen Testsets erfolgt jedoch weiterhin manuell.



(a) Die verfügbaren Erprobungsdaten sind in einer Karte visualisiert. Das ausgewählte Testszenario ist rot hervorgehoben. Im unteren Drittel wird der Geschwindigkeitsverlauf der ausgewählten Fahrt (rot) und die Geschwindigkeit enthaltener Verkehrsobjekte (violett) dargestellt. Aus dem abgeleiteten Testszenario ausgeschlossene Teile der Aufzeichnung sind ausgegraut.



(b) Listenansicht der geladenen Verzeichnisse, darin verfügbaren Erprobungsdaten und der zu einem Testset zusammengefassten Testszenarien.

Abbildung 7.1: Screenshots vom Werkzeug zur Verwaltung und Auswahl der aufbereiteten Erprobungsdaten und zur Ableitung von Testszenarien und Testsets.

7.2 Bewertung der Simulationsergebnisse und Prozessintegration

Eine vollständige automatisierte Bewertung des Funktionsverhaltens des PCC-Features stellt eine ähnlich komplexe Herausforderung dar, wie die eigentliche Entwicklung der modellprädiktiven Regelung. Das erwartete Regelungsverhalten kann nicht exakt spezifiziert werden, da in der Realität eine große Zahl an Sondersituationen und technisch nicht erfassten Randbedingungen auftreten können. Beispielsweise sind im System keine detaillierten Informationen zur Randbebauung, der lokalen Straßenqualität oder der Genauigkeit des elektronischen Horizonts verfügbar. Diese Faktoren beeinflussen jedoch das subjektive Empfinden der Passagiere zu Sicherheit, Fahrkomfort und Effizienz des Systems. Wie in Abschnitt 3.6.2 eingeführt wurde, werden diese Systemeigenschaften in der Fahrzeugerprobung mittels subjektiv interpretierten in Fahrmanöverkatalogen spezifizierten Kriterien geprüft. Beispiele für diese weichen Kriterien sind Formulierungen wie „die Annäherung erfolgt harmonisch und stimmig“ oder „das Fahrzeug zeigt ein fahrerkonformes Fahrverhalten“.

Die Zielsetzung der Tests in der Simulationsumgebung, auf System- und Subsystemebene einen Nachweis über das korrekte Verhalten des Features zu erbringen, erfordert eine Objektivierung der Kriterien. In den folgenden Unterabschnitten werden ein Ansatz für Regressionstests zur Identifikation unerwünschter Änderungen auf einer breiten Datenbasis und ein Ansatz für eine kontinuierliche Prüfung von Anforderungen auf Basis der Simulationsergebnisse vorgestellt.

7.2.1 Regressionsteststrategie auf Systemebene

Eine Metrik zur allgemeingültigen Bewertung der Simulationsergebnisse müsste alle Systemeingänge, deren Abhängigkeiten und variierenden Einfluss auf den Systemausgang berücksichtigen. Eine solche Metrik ist bereits in der modellprädiktiven Regelung in Form der zur Optimierung der Geschwindigkeitstrajektorie genutzten Gütefunktion enthalten. Da die Entwicklung einer kontinuierlichen und allgemeingültigen Bewertungsmetrik hohe Aufwände erfordert und unter Umständen eine abweichende und widersprüchliche Interpretation gegenüber der implementierten Gütefunktion realisiert, erfolgt die Bewertung und Parametrierung des Regelungsverhaltens primär auf Basis von Fahrzeugerprobungen und der manuellen Analyse

des Simulationsergebnisses durch einen Regelungsexperten. Um den übrigen Softwareentwicklern eine teilautomatisierte Bewertung von durchgeführten Änderungen zur Verfügung zu stellen, wird ein iteratives Vorgehen auf der Grundlage von Regressionstests auf Systemebene angewandt. Dieses in Abbildung 7.2 skizzierte Vorgehen basiert auf der Annahme, dass immer der zuletzt im Fahrzeug validierte und freigegebene Softwarerelease ein gültiges Zwischenergebnis darstellt.

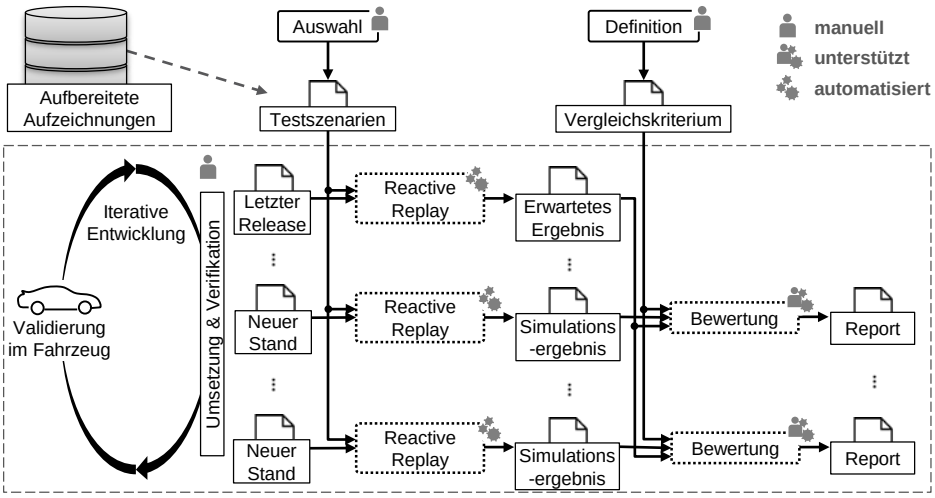
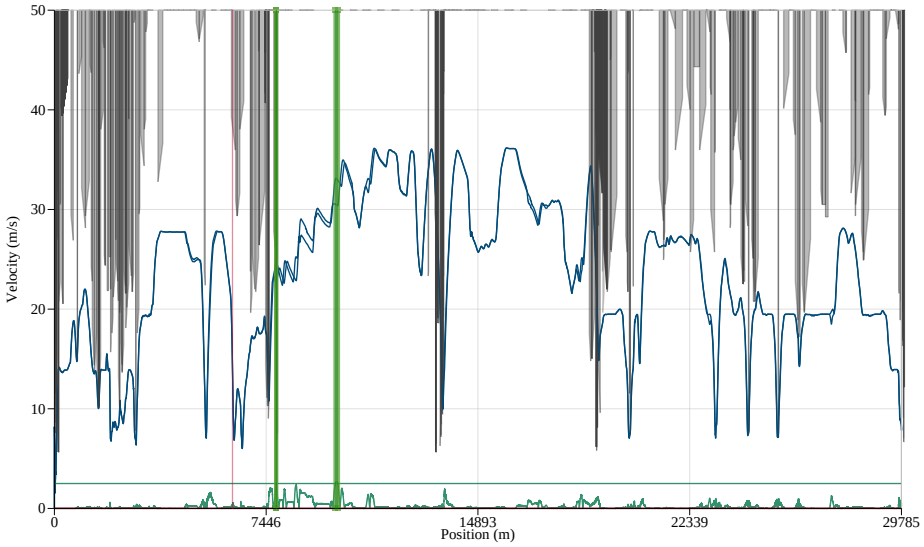


Abbildung 7.2: Regressionstest auf Systemebene durch Vergleich des Simulationsergebnisses mit dem erwarteten Ergebnis des letzten Releases

Für jeden Entwicklungszyklus wird zu Beginn eine Untermenge an Testszenarien aus den aufbereiteten Erprobungsdaten ausgewählt. Auf Basis des letzten Softwarerelease wird für den Entwicklungszyklus und die ausgewählten Testszenarien das erwartete Ergebnis durch die simulierte Ausführung des Releases mittels Reactive-Replay generiert. Änderungen, die während eines Entwicklungszyklus durchgeführt werden, führen zu einem neu zu bewertenden Softwarestand. Dieser neue Stand wird simuliert und das Simulationsergebnis wird über zuvor definierte Bewertungskriterien mit dem erwarteten Ergebnis verglichen. Neben einem Bit-identischen Vergleich der Ausgangssignale kommen in Abhängigkeit des Entwicklungsfortschritts und der jeweiligen Zielsetzung toleranzbehaftete Kriterien zum Einsatz. Bewegen sich die Abweichungen innerhalb einer zulässigen Toleranz, können die Änderungen in den nächsten Release übernommen werden. Übersteigen sie diese Toleranz prüft der Entwickler ob es sich bei den Abweichungen um erwünschte Änderungen oder unerwünschte Auswirkungen handelt. Sind alle Änderungen im Verhalten erwünscht, werden diese zusätzlich durch einen Regelungsexperten geprüft und freigegeben.



(a) Darstellung der Geschwindigkeitstrajektorien (blau), der Begrenzung des Geschwindigkeitsbereichs (grau) und des Vergleichsergebnisses (grün) über der zurückgelegten Streckenposition. Die grünen Markierungen signalisieren eine Toleranzüberschreitung.

(b) Visualisierung des Vergleichsergebnisses in Bezug auf die GPS-Position des Fahrzeugs in einer Kartenansicht. Grüne Markierungen signalisieren die Position von Toleranzüberschreitungen.

Abbildung 7.3: Beispiel eines Bewertungskriteriums zum Vergleich der simulierten Geschwindigkeitstrajektorie eines neuen Softwarestands mit dem erwarteten Ergebnis des letzten Releases.

Abbildung 7.3 stellt das Ergebnis eines einfachen Beispielbewertungskriteriums zum Vergleich der simulierten Geschwindigkeitstrajektorien dar. Das Kriterium prüft ob die absolute Differenz der simulierten Geschwindigkeiten (grüner Signalverlauf im unteren Bildbereich in Abb. 7.3a) unterhalb eines definierten Schwellwerts liegt (grüne Linie bei 2,5 m/s in Abb. 7.3a). In der Darstellung über der Streckenposition in Abbildung 7.3a werden Abschnitte mit überschrittenem Schwellwert farblich abgesetzt markiert. Der Verlauf der absoluten Differenz über der GPS-Position des Fahrzeugs ist in Abbildung 7.3b dargestellt. Hier markieren die grünen Quadrate jeweils den Beginn und das Ende einer Schwellwertüberschreitung.

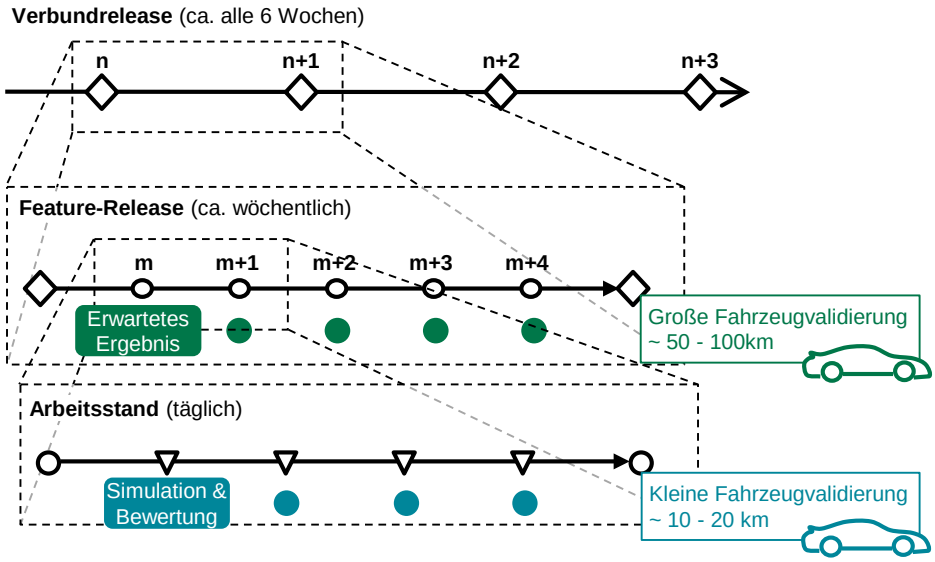


Abbildung 7.4: Anwendung der Regressionstests im bestehenden Entwicklungsprozess der Produktlinie.

Die Regressionsteststrategie ist ohne Anpassungen am bestehenden Entwicklungsprozess mit festen Meilensteinen zur Abgabe von integrationsfähigen Softwareständen anwendbar. Die entwicklungsbegleitenden Integrationsstände werden, wie in Abbildung 7.4 dargestellt ist, im übergeordneten Produktentstehungsprozess (vergl. Kap. 2.9) als Verbundrelease bezeichnet und werden in einem Abstand von ca. 6 Wochen der zuständigen Organisationseinheit zur Integration mit weiteren SWCs im Steuergerät geliefert. Jedes Verbundrelease wird vor der Abgabe durch eine Fahrzeugerprobung im Umfang einer Strecke von 50 bis 100 Kilometern validiert.

Der Zeitraum eines Verbundreleases lässt sich in wöchentliche Feature-Releases unterteilen. Ein solches Release fasst die Änderungen einer Arbeitswoche zusammen

und wird im Rahmen einer kleinen Fahrzeugerprobung von 10 bis 20 Kilometer validiert. Der Softwarestand des Feature-Releases dient zur Generierung des erwarteten Ergebnisses, das in der täglichen Entwicklungsarbeit zur Bewertung der durchgeführten Simulationen genutzt wird. Durch diese Regressionstests werden die Zyklen zwischen der Umsetzung einer gewünschten Änderung und deren Prüfung im Wirkkettenverbund auf Systemebene im Vergleich zur Fahrzeugerprobung extrem verkürzt und die Aufwände werden reduziert. Die Softwareentwickler erhalten zu kleinsten umgesetzten Implementierungen und Änderungen ein direktes Feedback in Bezug auf die Auswirkungen auf das Regelungsverhalten. Auch Entwickler mit geringem Vorwissen bezüglich der optimierten Längsdynamikregelung von Kraftfahrzeugen können auf dieser Grundlage eine erste Einschätzung zur Validität ihrer Arbeit treffen.

7.2.2 Prüfung von Anforderungen auf System- und Subsystemebene

Neben den bisher beschriebenen vergleichsbasierten Regressionstests ermöglicht die wiederholte Ausführung von in der Fahrzeugerprobung aufgezeichneten Situationen und Szenarien auch eine wiederkehrende Überprüfung von System- und Subsystemanforderungen des PCC-Features. Die im Folgenden vorgestellte Wiederverwendung einzelner Sequenzen aufgezeichneter Fahrzeugerprobungen, sowie eine kontinuierliche Anforderungsprüfung ergänzen die etablierten Testmethoden um einen reproduzierbaren und automatisierten Softwaretest der SWC. Die abgeleiteten Testszenarien werden in das Konzept der in der Entwicklung eingesetzten Continuous Integration Werkzeugkette (vergl. Abschnitt 5.4.4) eingebunden und ermöglichen damit eine kontinuierliche und tagesaktuelle Ermittlung des Reifegrads und des Fortschritts der PCC-Entwicklung.

Virtueller Fahrmanöverkatalog

In der systematischen Fahrerprobung (Abschnitt 3.6.2) werden die in den Prüfkatalogen spezifizierten Fahrmanöver durch Testfahrer umgesetzt und entsprechend ihrer subjektiven Interpretation bewertet. Durch die Nutzung der während dieser Erprobungen aufgezeichneten Fahrzeugdaten in der Reactive-Replay Simulation stehen diese Fahrmanöver als Testszenarien für den virtuellen Test der SWC zur Verfügung. Die spezifizierten Manöver können in eindeutig spezifizierte und subjektiv zu interpretierende Prüfungen unterteilt werden. Eindeutig spezifizierte

Prüfungen können dabei vollständig automatisiert getestet werden. Beispiele dafür sind die Änderung der Systemanzeigen (z.B. angezeigte Zeitlücke) als Reaktion auf eine Bedienhandlung oder die Einregelung eines bestätigten Tempolimits. Für die Auswertung dieser Testfälle wird die Position des aktivierenden Ereignisses markiert und eine Bedingung zur Prüfung der Systemreaktion mit zeitlichem oder positionsabhängigem Versatz definiert. Erfolgt die erwartete Reaktion innerhalb der definierten Zeitspanne oder Fahrstrecke wird der Testfall als bestanden gewertet, andernfalls als nicht bestanden.

Die Bewertung subjektiv zu interpretierender Fahrmanöver erfolgt zunächst durch einen Regelungsexperten. Der im vorherigen Abschnitt 7.2.1 vorgestellte Regressionsansatz realisiert eine automatisierte reproduzierbare Prüfung einzelner Szenarien und eine systematische Überwachung der subjektiven Regelungsgüte. Mit fortschreitender Entwicklung und Konkretisierung des entwickelten Features sinkt die Varianz des erwarteten Regelungsverhaltens. Einerseits können damit die zulässigen Wertebereiche der kontinuierlichen Bewertungskriterien eingeschränkt werden, andererseits ermöglicht dies die Spezifikation explizit erwarteten Verhaltens auf ausgewählten Sequenzen der verfügbaren Testszenarien. Abbildung 7.5 stellt einen entsprechenden Testfall auf Systemebene und dessen Auswertung mittels statischer Gates dar.

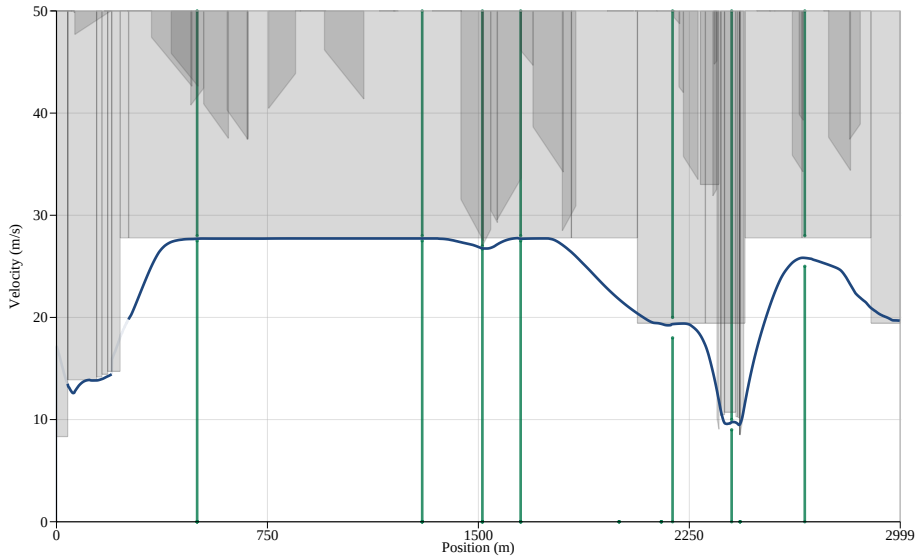


Abbildung 7.5: Testfall zur Prüfung der korrekten Funktionserfüllung einer ausgewählten Sequenz anhand statischer Gates.

Der vorgestellte Testfall basiert auf einer Sequenz von 2500m Länge. Für diese Sequenz wurden sieben Geschwindigkeits-Gates spezifiziert, die an ihrer jeweiligen Streckenposition für die in der Simulation realisierte Geschwindigkeit einen Toleranzbereich vorgeben. Wird dieser Bereich eingehalten, wird das jeweilige Gate als bestanden gewertet. Werden alle Gates als bestanden gewertet, gilt der gesamte Testfall als erfolgreich validiert. Zur Visualisierung werden die Gates entsprechend grün (für bestanden) bzw. rot (für durchgefallen) eingefärbt. Dieses Vorgehen ermöglicht die Integration der spezifizierten Systemtests in das auf Einheitenebene genutzte Unit-Test Werkzeug und damit auch eine nahtlose Integration in die verwendete Continuous-Integration Werkzeugkette.

Kontinuierliche Anforderungsprüfung

Für die kontinuierliche Anforderungsprüfung werden zustandsbasierte Prüfbedingungen implementiert, die parallel zur Simulationsausführung ausgewertet werden. Abbildung 7.6 stellt das Zustandsdiagramm der Prüfbedingungen dar. Eine Bedingung besitzt die drei Zustände „Passiv“, „Bereit“ und „Aktiv“ und löst bei einer Transition die Ereignisse „markReady()“, „markTrigger“, „markException()“, „markFail()“ und „markPass()“ aus. Die Transitionen erfolgen basierend auf den Bedingungen „Vorbedingung“, „Aktivierung“, „Abbruch“, „Illegale Auslösung“, „Zeit- oder Positionsüberschreitung“ und „Auslösung“. Das Prinzip dieser Zustände, Transitionen und Bedingungen wird im Folgenden anhand eines Beispiels erläutert.

Als Beispiel dient eine Anforderung zur Aktivierung des Systems. Mit dem Start der Simulation wird die Prüfbedingung im Zustand „Passiv“ initialisiert. Die Vorbedingung zur Aktivierung ist das Vorliegen des Signals „Systemstatus aktiv“. Liegt dieses Signal am Ausgang der SWC an, erfolgt eine Transition in den Zustand „Bereit“. Von diesem Zustand wird durch einen Wechsel zu „Systemstatus nicht verfügbar“ eine Abbruchbedingung und damit ein Übergang in den Prüfbedingungszustand „Passiv“ sowie das Ereignis „markException()“ ausgelöst. Erfolgt ohne eine durch den Fahrer angeforderte Aktivierung ein illegaler Wechsel in den „Systemstatus aktiv“ findet ein Übergang in den Zustand „Passiv“ statt und ein Fehlerereignis „markFail()“ wir ausgelöst. Als dritte Transition versetzt eine Aktivierungsanforderung des Fahrers die Prüfbedingung in den Zustand „Aktiv“. Durch die Abbruchbedingung „Systemstatus nicht verfügbar“ wird dieser in den Zustand „Passiv“ verlassen. Alternativ führt die Überschreitung einer maximal zulässigen Zeit- oder Streckendifferenz ohne erwartete Systemreaktion zu einem Fehlerereignis

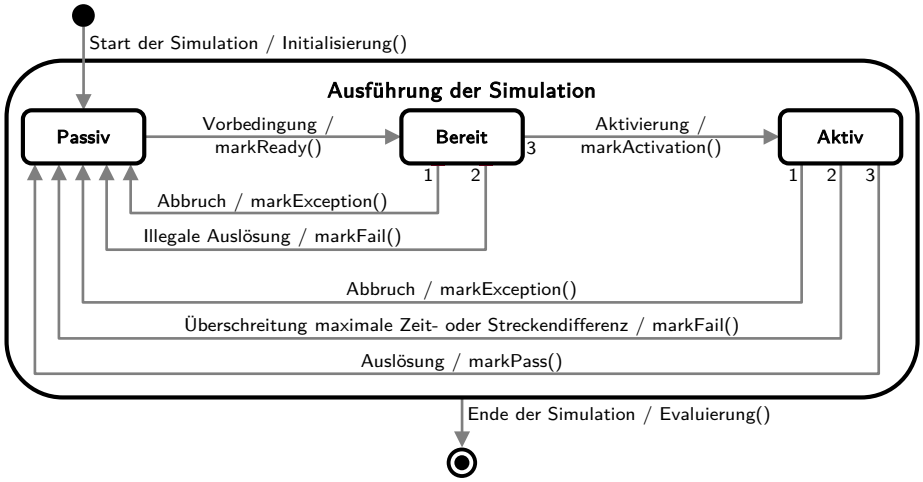


Abbildung 7.6: Zustandsdiagramm für eine kontinuierliche Prüfung spezifizierter Anforderungen

und einem Übergang in den Prüfbedingungszustand „Passiv“. Mit der erwarteten Systemreaktion eines Wechsels nach „Systemstatus aktiv“ wird der Übergang in den Prüfbedingungszustand „Passiv“ und ein Bestanden-Ereignis ausgelöst.

Die Auswertung der Prüfbedingungen wird in das Unit- und Modultest Framework der Continuous Integration Toolchain integriert. Eine Prüfbedingung wird darin als bestanden gewertet, falls während der Simulation mindestens ein Bestanden-Ereignis und kein Fehlerereignis ausgelöst wurde. Wird ein Testset mit einer großen Menge an Testszenarien ausgeführt, liefert die Anzahl enthaltener Bestanden-Ereignisse eine Aussage über die erreichte Testtiefe.

7.3 Systematische Testszenarien Auswahl

Die in Abschnitt 7.1 beschriebene Auswahl der Testszenarien basiert auf dem Expertenwissen der Entwickler. Während dies grundsätzlich eine angemessene Strategie darstellt, besteht jedoch das Risiko, dass aktuelle Themen und Fragestellungen der Entwickler die Auswahl beeinflussen. Darüber hinaus erhöhen die Daten, die auf internationalen Feldversuchskampagnen aufgezeichnet wurden, schnell die Anzahl der verfügbaren Testszenarien. Der daraus resultierende Umfang ist durch ein unstrukturiertes Auswahlkonzept nicht länger überschaubar. Aus diesen Gründen

wird ein systematischer Ansatz für eine objektive Testszenarienauswahl benötigt, der das Risiko minimiert wichtige Testszenarien zu übersehen oder ein unausgeglichenes und voreingenommenes Testset zusammenzustellen. Im folgenden wird ein Konzept und dessen Umsetzung als ein erster Ansatz zur systematischen Auswahl von Testszenarien präsentiert, dieses Forschungsfeld wird jedoch aufgrund des großen Umfangs nicht vollständig bearbeitet.

Eine Möglichkeit eines objektiven Auswahlprozesses ist die Bewertung der einzelnen Szenarien und die entsprechende Auswahl der am besten bewerteten Szenarien. Blumenstock [266] beschreibt eine Methode zur Bewertung einzelner Abschnitte des öffentlichen Straßennetzes in Bezug auf ihren Mehrwert für Fahrerprobungen. Spezifische Kriterien definieren eine kombinierte Metrik zur Bewertung des Straßenabschnitts. Während dieser gradlinige Bewertungsansatz für die Routenplanung von Erprobungsfahrten hilfreich ist, werden inhaltliche Wiederholungen nicht berücksichtigt. Ein derartiger Ansatz birgt daher die Gefahr einer übermäßigen Prüfung bestimmter hoch bewerteter Parameterkombinationen bei gleichzeitigem Auslassen anderer wichtiger Szenarien.

Die Methoden Design of Experiments (DOE) und Factorial Design [267] beschreiben eine Technik, um Korrelationen und Auswirkungen verschiedener mehrschichtiger Einflussfaktoren auf ein System zu identifizieren. Ein Full Factorial Design wertet das System auf alle möglichen Kombinationen von Eingängen unterschiedlicher Ebenen aus. Dies führt zu einer numerischen Explosion. Fractional Factorial Designs, wie z. B. orthogonale Kombinationsmatrizen [267], schränken den Kombinationsraum ein. DOE als statistischer Ansatz berücksichtigt keine zeitvarianten Faktoren, die für die Auswertung des PCC-Features wichtig sind. Die Interpretation der Einflussfaktoren als multidimensionaler Raum, der nicht vollständig abgedeckt sein muss, ist für die Auswahl der Testszenarien wichtig.

Die Auswahl eines ausgewogenen Satzes von Testszenarien erfordert eine Lösung, die alle aufgezeichneten Testszenarien auswertet und enthaltene Wiederholungen einbezieht. Das Verfahren sollte den mehrdimensionalen Eingangsraum der Regelung analysieren und eine abwechslungsreiche Auswahl ermöglichen, die alle relevanten Kombinationen abdeckt. Abbildung 7.7 stellt ein zweistufiges Konzept zur Auswahl von Testszenarien dar, das diese Anforderungen berücksichtigt. Es zielt darauf ab, einen auf der Spezifikation und einen auf der Struktur basierenden Ansatz zu kombinieren.

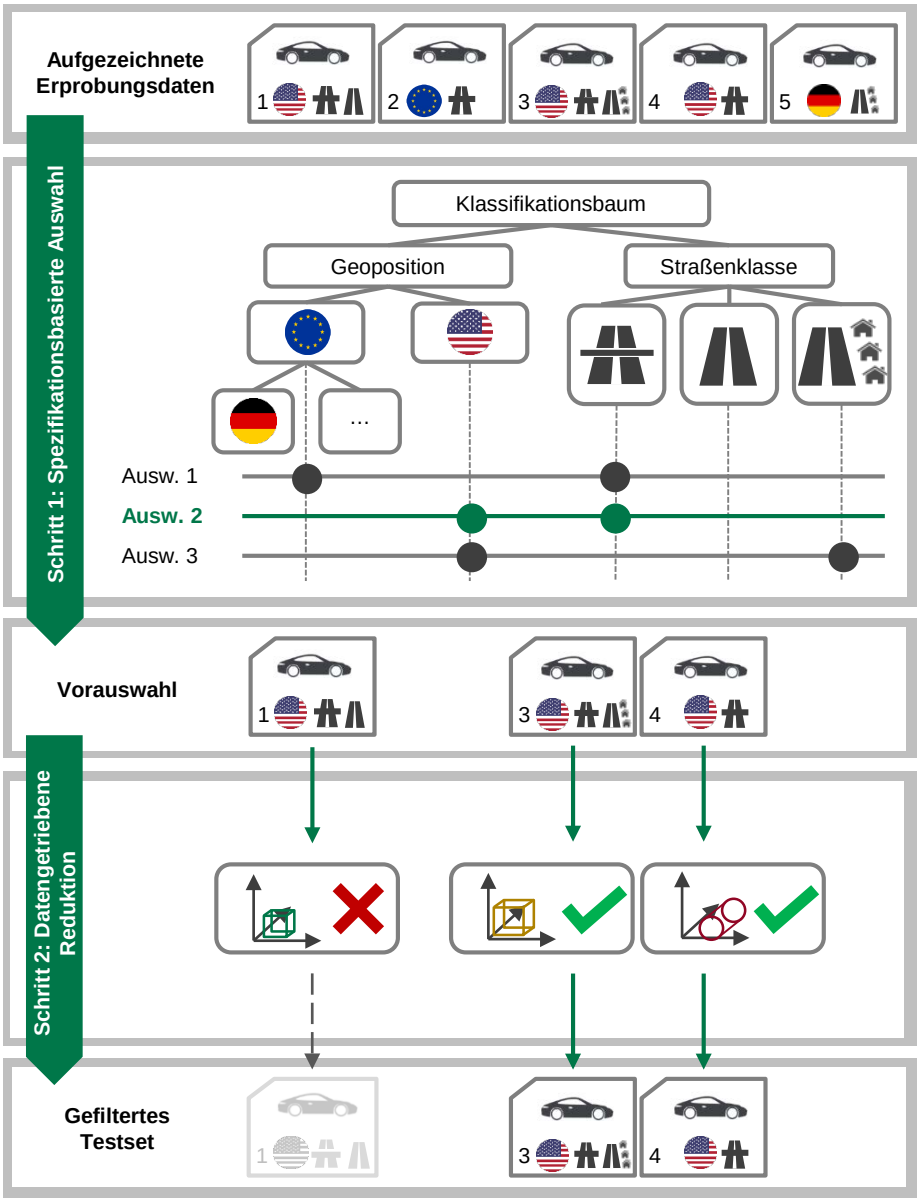


Abbildung 7.7: Das zweistufige Vorgehen zur Auswahl eines Testsets basiert auf der Anwendung einer Klassifikationsbaummethode zur Vorauswahl der Testszenarien und einer Analyse der Parameterüberdeckung zur Reduktion des Testset auf die relevanteren Szenarien.

Das Konzept folgt Bourque's [10] Rat: "Diese beiden Ansätze zur Auswahl der Tests sind nicht als Alternativen zu sehen, sondern eher als Ergänzungen"[1]. Während Bourque einen Kontrast zwischen funktionalen und strukturellen Tests beschreibt, lässt sich dieser Rat im Kontext der Testszenarienauswahl auf die ähnlichen Definitionen von Spezifikations- und Strukturbasierten Techniken übertragen, die in Abschnitt 3.7.1 vorgestellt wurden.

Der erste Schritt des Auswahlkonzepts ist ein spezifikationsbasierter Top-Down Ansatz, der vordefinierte Kriterien zur Klassifikation der Testszenarien nach abstrakten Kriterien nutzt. Dies bietet uns unter anderem die Möglichkeit, Labels und Markups, die während der Erprobung erstellt wurden, einzubeziehen. Im zweiten Schritt werden die vorausgewählten Szenarien mit Hilfe eines Bottom-up Ansatzes reduziert. Zunächst wird der Einfluss in verschiedenen Testszenarien enthaltener Information auf die strukturbasierte Testabdeckung untersucht, um auf dieser Basis sich wiederholende Szenarien zu identifizieren und entsprechend zu streichen. Wie im Folgenden gezeigt wird, ist für ein Regelungssystem eine Analyse des Parameterraums der Ein- und Ausgänge zur Testszenarienauswahl den konventionellen Metriken zur Ermittlung der Testabdeckung vorzuziehen. Daher wird der zweite Schritt als datengetriebene Reduktion bezeichnet.

7.3.1 Spezifikationsbasierte Auswahl

Der Schritt zur spezifikationsbasierten Auswahl basiert auf der Aufteilung der Szenarien entsprechend abstrakter, in Übereinstimmung mit festgelegten Systemanforderungen definierten Kategorien, vergleichbar mit dem Ansatz zur Äquivalenzklassenbildung (Abschnitt 3.7.1). Wie in Abbildung 7.7 dargestellt ist, beinhaltet dies eine geographische Unterteilung, beispielsweise entsprechend verschiedenen Ländern mit unterschiedlichen Verkehrsregeln, aber auch eine kontextuelle Differenzierung, wie beispielsweise unterschiedliche Straßenklassen (Autobahn, Landstraße, innenstädtisch). Weitere Kriterien können sich auf den Zustand des Ego-Fahrzeugs, zum Beispiel das jeweilige Fahrmanöver, die Fahrzeugumgebung, z.B. die Verkehrsdichte, und systemspezifische Labels und Mark-Ups beziehen.

Die spezifizierten Kategorien werden, der Klassifikationsbaum-Methode (Abschnitt 3.7.1) folgend, zur systematischen Vorauswahl geeigneter Szenarien genutzt. Damit

[1] engl.: „These two approaches to test selection are not to be seen as alternatives but rather as complements“

werden dedizierte Testsets erzeugt, die für das PCC-Feature relevante Use-Cases in den für eine Markteinführung vorgesehenen Ländern und Regionen enthalten. Der Ansatz stellt sicher, dass alle relevanten Kombinationen abstrakter Eigenschaften in den ausgewählten Szenarien enthalten sind. Dabei ist zu beachten, dass zwar die verwendeten Auswahlkriterien disjunkt sind, jedoch nicht die einzelnen Szenarien. Zum Beispiel kann ein Testszenario sowohl einen Autobahnabschnitt als auch einen Abschnitt einer Landstraße enthalten.

Die in diesem Schritt vorausgewählten Testszenarien können sich wiederholende Situationen und Informationen enthalten. Daher wird im zweiten, nachgelagerten Schritt die Länge der zu simulierenden Strecke komprimiert, in dem sich wiederholende Informationen entfernt werden.

7.3.2 Datengetriebene Reduktion

Eine Möglichkeit zur Bestimmung der Testtiefe bieten strukturelle Überdeckungsanalysen (Abschnitt 3.7.2). Im Kontext eines Regelungssystems ist die Messung der einmaligen Ausführung jeder Programmanweisung oder Bedingung unzureichend. Die in Tabelle 7.1 abgebildete werkzeuggestützte Analyse[2] der strukturellen Abdeckung während der Reactive-Replay Simulation zeigt, dass die Erhöhug der simulierten Distanz die Anweisungs- und Bedingungsüberdeckung nur geringfügig erhöht.

Anzahl der Testszenarien	**Gesamtlänge in Summe**	**Anweisungs-überdeckung**	**Bedingungs-überdeckung**
1	ca. 5 km	74%	60%
1	ca. 50 km	76%	68%
5	ca. 200 km	76%	71%
11	ca. 500 km	76%	73%
21	ca. 1000 km	76%	73%

Tabelle 7.1: Vergleich der Anweisungs- und Bedingungsüberdeckung einer verschiedenen Anzahl von Testszeanrios unterschiedlicher Gesamtlänge.

Um eine Vielzahl von Fällen abzudecken, muss die Ausführung der Funktion als Ganzes für unterschiedliche Eingangsvarianten ausgewertet werden. Der vorgestellte Ansatz basiert daher auf der in Abschnitt 3.7.2 vorgestellten Parameterüberdeckung, der eine „Sicht auf ein Softwareprogramm als eine Zuordnung eines Raums

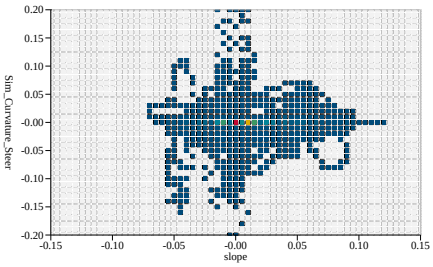
[2] BullseyeCoverage, Bullseye Testing Technology http://www.bullseye.com/

von Eingängen auf einen Raum von Ausgängen“[3] [161] zugrunde liegt. Daher konzentriert sich das vorgestellte Konzept auf die Analyse der Zusammensetzung des mehrdimensionalen Eingangsraum des PCC-Features. Die großen Wertebereiche der einzelnen Eingangsparameter summieren sich dabei zu einer sehr großen Anzahl von Parameterkombinationen. Aus diesem Grund ist auch eine generische Erzeugung aller Kombinationen nicht zielführend, da viele der Kombinationen eher von theoretischer Natur sind und in der Realität nicht auftreten. Die Reactive-Replay Methode beschränkt die genutzten konkreten Testszenarien (vergl. Abschnitt 3.3.4) auf aufgezeichnete Erprobungsdaten und begrenzt damit die Vielfalt der Kombinationen bereits auf ein realitätsnahes Maß. Eine Abdeckung der Extremfälle wird durch ergänzende, generische Testfälle und -szenarien mit Hilfe der Grenzwertanalyse (Abschnitt 3.7.1) sichergestellt.

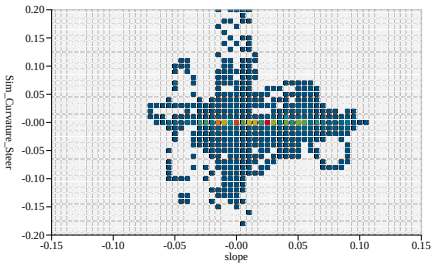
Reale Fahrdaten sind weitgehend redundant, da in den kontinuierlichen Aufzeichnungen wiederholt ähnliche Situationen auftreten. Das Ziel ist die Auswahl einer minimalen Teilmenge von Testszenarien, die eine hohe Variation des Parameterbereichs aufweist und die Fähigkeiten des Testobjekts auslastet. Untersucht werden müssen alle Größen, die einen entscheidenden Einfluss auf die Ausgangssignale und damit das Verhalten der analysierten Funktion haben. Die entscheidenden Ausgänge des PCC-Features sind dessen Steuergrößen Zielgeschwindigkeit, Beschleunigung und Gangwahl. Sie werden primär durch die Krümmung und Steigung der Straße, die aktuelle Geschwindigkeitsbegrenzung und, falls vorhanden, die Relativbewegung und Zeitlücke zu einem erfassten Zielobjekt beeinflusst.

Das Konzept der datengetriebenen Reduktion vergleicht die gesamtheitliche Parameterüberdeckung aller Testszenarien der Vorauswahl mit der Parameterüberdeckung möglicher Kandidaten. Das finale, gefilterte Testset wird durch den Vergleich der bivariaten Histogramme (Abb. 7.8) aller oben identifizierten relevanten Eingangsgrößen erstellt. Diese Histogramme ermöglichen einen einfachen visuellen Abgleich der jeweiligen Überdeckungen und der enthaltenen Korrelationen. Basierend auf dieser Analyse werden nur die Testszenarien ausgewählt, die zuvor leere oder dünn besetzte Regionen des Wertebereichs belegen. Testszenarien, die nur stark besetzte Regionen belegen, werden zur Reduktion des Datenumfangs fallengelassen. Diese Methode wählt damit eine Untermenge an Testszenarien mit dem größten Mehrwert und der geringsten Informationsredundanz für die Analyse der Funktion aus. Das Ziel, möglichst kompakte Testsets mit Hoher Variation und Belastung für das Testobjekt zu extrahieren, ist erfüllt.

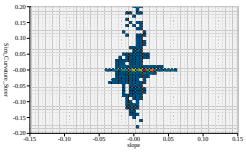
[3] „based on a view of a software program as a mapping from a space of inputs into a space of outputs“



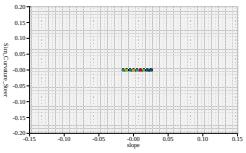
(a) Überdeckung alle Szenarien



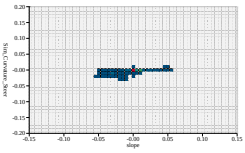
(b) Überdeckung ausgewählte Szenarien



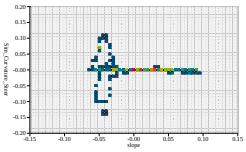
(c) Szen. 1 (gewählt)



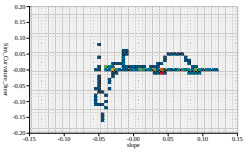
(d) Szen. 2 (reduziert)



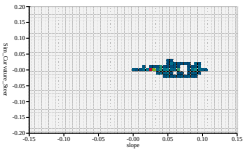
(e) Szen. 3 (reduziert)



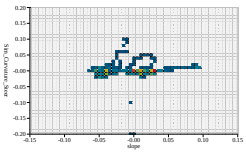
(f) Szen. 4 (gewählt)



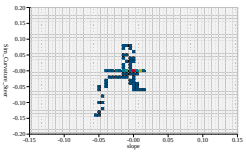
(g) Szen. 5 (reduziert)



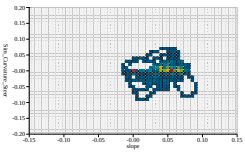
(h) Szen. 6 (gewählt)



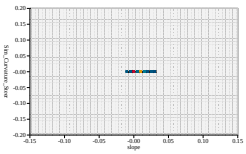
(i) Szen. 7 (reduziert)



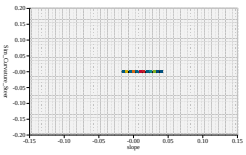
(j) Szen. 8 (reduziert)



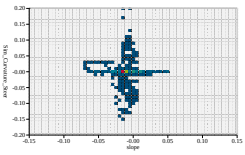
(k) Szen. 9 (gewählt)



(l) Szen. 10 (reduziert)



(m) Szen. 11 (reduziert)



(n) Szen. 12 (gewählt)

Abbildung 7.8: Beispiel des datengetriebenen Reduktionsschritts auf Basis der in 12 Testszenarien enthaltenen Werte für Kurvenkrümmung und Straßensteigung

Zur Demonstration des Konzepts wird die in Abbildung 7.7 hervorgehobene Kriterienkombination „Auswahl 2“ genutzt. Diese übernimmt alle Testszenarien auf Autobahnen in den USA in die Vorauswahl. Die Filterkriterien werden in dem in Abschnitt 7.1 vorgestellten Werkzeug für das Test-Szenario-Management implementiert. Aus Gründen der Übersichtlichkeit wurde das vorausgewählte Beispieltestset manuell auf 12 Szenarien reduziert, bevor der datengetriebene Reduktionsschritt gestartet wird. Der visuelle Abgleich wird am Beispiel der beiden Eingangsparameter Straßensteigung und Straßenkrümmung (abgeleitet aus dem Lenkwinkel) erläutert. In Abbildung 7.8 sind die entsprechenden bivariaten Histogramme der gesamten Vorauswahl, des finalen gefilterten Testsets und jedes einzelnen Testszenarios abgebildet.

Abbildungen 7.8c bis 7.8n zeigen für jedes der vorgewählten Szenarios die Parameterüberdeckung der Steigung und der Krümmung. Abbildung 7.8a bildet die akkumulierte Überdeckung aller Szenarien ab und die akkumulierte Überdeckung der reduzierten Auswahl ist in Abbildung 7.8b dargestellt. Das Beispiel zeigt, dass die Auswahl von 5 Szenarien einen sehr ähnlichen Parameterraum abdecken kann wie alle 12 Szenarien in Kombination abdecken. Obwohl dies nur ein Beispiel ist und aus Gründen der Übersichtlichkeit nur zwei Eingabedimensionen betrachtet werden, ist es offensichtlich, dass im Reduktionsschritt mit einigen wenigen gut gewählten Testszeanarien der Parameterraum belegt und eine Mehrheit der Testfälle weggelassen werden kann.

Das Beispiel Testset umfasst eine Gesamtdistanz von 22,9 km. Es wird mit der vorgestellten Methode ohne signifikante Verluste auf 8,6 km reduziert, wie ein visueller Vergleich der Abbildungen 7.8a und 7.8b zeigt. 7 von 12 Szenarien, oder mit anderen Worten, mehr als 50% werden während des datengetriebenen Reduktionsschritts fallen gelassen. Für dieses vereinfachte Beispiel ist der visuelle Vergleich ausreichend.

Im Falle einer höheren Anzahl und Länge der Testszenarien kann über den visuellen Vergleich eine statistisch repräsentative Auswahl nicht mehr garantiert werden. In diesem Fall sollte eine Annäherung der Wahrscheinlichkeitsverteilung des vollständigen Testsets durch den reduzierten Testsatz angestrebt werden.

7.4 Evaluation und Bewertung

Die in diesem Kapitel beschriebenen Ansätze für den systematischen Einsatz der Reactive-Replay Methode wurden am Beispiel des in Kapitel 5 beschriebenen PCC-Entwicklungsprojekts erprobt und evaluiert. Im Folgenden werden die während des prototypischen Einsatzes und im produktiven Alltag des Entwicklungsprojekts gesammelten Erfahrungen im Umgang mit den eingeführten Regressionstests auf Systemebene vorgestellt. Die Ansätze zur kontinuierlichen Anforderungsprüfung und zur datengetriebenen Testszenarienauswahl werden an ihrer prototypischen Realisierung diskutiert. Sie konnten sich jedoch nicht im produktiven Alltag des Entwicklungsprojekts bewähren.

Regressionstests auf Systemebene

Die Erfahrung zeigt, dass der Einsatz der vergleichenden Regressionstests auf Systemebene insbesondere gegen Ende der weitestgehend parallel durchlaufenen Phasen SWE.3 „Detaillierter Softwareentwurf und Implementierung“ und SWE.4 „Verifikation Softwareeinheiten“ das funktionale Verhalten gegenüber unerwünschten Änderungen absichert. Insbesondere verhindert der Ansatz während der parallelen Arbeit mehrerer Softwareentwickler ein auseinanderdriften des funktionalen Verhaltens in den verschiedenen Entwicklungszweigen. Bei der prototypischen Realisierung und der anschließenden Nutzung der Regressionstests haben sich folgende Erfahrungswerte ergeben:

- Für Refaktorierungen (s. Abschnitt 3.5.1) eignet sich insbesondere der Bit-identische Vergleich der Simulationsergebnisse unterschiedlicher Softwarestände.
- Das Hervorheben der Abweichungen unterstützt die Entwickler bei der Identifikation enthaltener Fehlerzustände.
- Der Mehrwert komplexer und allgemeingültiger Vergleichskriterien rechtfertigt den Aufwand zur Erstellung eher nicht. Statt dessen können mit fortschreitender Konkretisierung des Funktionsverhaltens für ausgewählte Testszenario-Sequenzen und die darin enthaltenen Manöver auf Grundlage von statischen Gates explizite Testfälle definiert werden.

Kontinuierliche Anforderungsprüfung

Die prototypische Umsetzung des Ansatzes zur kontinuierlichen Anforderungsprüfung wurde am Beispiel von spezifizierten Bedienhandlungen umgesetzt. Neben der Aktivierung und Deaktivierung des Systems wurden Prüfungen zur korrekten Übernahme angeforderter Änderungen der Verkehrs- und Topographie unabhängigen Setzgeschwindigkeit implementiert. Dabei konnten folgende Erfahrungen gesammelt werden:

- Die prinzipielle Fähigkeit des Ansatzes, während der Simulation kontinuierlich auf Anforderungserfüllung zu prüfen, konnte nachgewiesen werden.
- Die vollständige und eindeutige Spezifikation der Übergangsbedingungen zwischen den drei Zuständen ist mit erheblichem Aufwand verbunden.
- Die Auswahl expliziter Testszenarien und eine spezifisch auf das Szenario spezifizierte Prüfbedingung (z.B. anhand der Streckenposition) ist effizienter
- Die komplexe Struktur der kontinuierlichen Prüfungen beinhaltet ein zu hohes Fehlerrisiko.

Datengetriebene Auswahl von Testszenarien

Der zweistufige Ansatz zur systematischen Auswahl von Testszenarien wurde prototypisch in die Werkzeugkette des PCC-Entwicklungsprojekts integriert. Die Evaluation des Ansatzes erfolgte auf Basis der aus den Aufzeichnungen von drei Ländererprobungen abgeleiteten Testszenarien. Im Umgang mit dem Ansatz wurden folgende Erfahrungen gesammelt:

- Der Ansatz ermöglicht eine Identifikation von Testszenarien die seltene Situationen beinhalten.
- Der im Schritt der datengetriebenen Reduktion enthaltene manuelle Aufwand übersteigt die in der Entwicklung verfügbaren Ressourcen um ein Weites. Daher erfordert ein produktiver Einsatz eine geeignete Automatisierung des Reduktionsschritts.

- Die Auswertung der bivariaten Histogramme überdeckt möglicherweise enthaltene Korrelationen zu weiteren Signalen und zeitliche Abhängigkeiten.

- Die Analyse der Überdeckung des Wertebereichs kann einen erheblichen Mehrwert bei der Planung von effizienten und effektiven Erprobungsrouten bieten.

8 Zusammenfassung und Ausblick

Zum Abschluss dieser Dissertation wird in diesem Kapitel ein Resümee über die vorgestellten Ergebnisse gezogen und der wissenschaftliche Beitrag herausgestellt. Im Ausblick werden Themen aufgezeigt, die im Rahmen dieser Arbeit nicht oder nur unvollständig behandelt wurden und Anknüpfungspunkte für zukünftige Forschungsarbeiten darstellen.

8.1 Zusammenfassung

In der vorliegenden Arbeit wurde zu Beginn der Stand der Technik der im Automobilsektor etablierten Entwicklungsstrategien aufgearbeitet und analysiert. Dies umfasste die angewandten Entwicklungsvorgehen und -prozesse (Kapitel 2) sowie etablierte Methoden für die Entwicklung, Verifikation und Validierung von E/E-Systemen, insbesondere in Bezug auf Fahrzeugregelungsfunktionen (Kapitel 3). In der Analyse wurde identifiziert, dass die Zunahme Software-getriebener Innovationen eine Einführung agiler Prinzipien begünstigt und Ansätze zur frühzeitigen Validierung das Risiko in der Entwicklung disruptiver Technologien verringern. Die untersuchten Entwicklungs- und Testmethoden sind auf die jeweiligen Aktivitäten der einzelnen Phasen des Entwicklungsvorgehens zugeschnitten. Mit zunehmender Komplexität gewinnen Prototypen in frühen Entwicklungsphasen an Bedeutung. Für eine grundlegende Qualitätssicherung im Umgang mit diesen Prototypen sind daher bereits in frühen Phasen Testaufwände zu erbringen. Für innovative Konzepte steigt dabei das Risiko, im Falle eines Abbruchs der Entwicklung, Aufwände für die Erstellung komplexer Testmodelle und -szenarien zu verschwenden. Als mögliche Alternative wurde die Verwendung der in den Erprobungen mit den Prototypen generierten Daten als Input für virtuelle Ansätze gesehen.

In Kapitel 4 wurden Kriterien zur Charakterisierung von E/E- und Software-Features definiert. Diese dienten zur Erstellung einer Taxonomie aktueller und zukünftiger Kundenfunktionen. Basierend auf dieser Analyse konnte eine konzeptuelle

logische Systemarchitektur abgeleitet werden, die eine ganzheitliche Betrachtung der funktionalen Zusammenhänge unterschiedlichster Systeme ermöglicht. Die Fähigkeit dieser generalisierten Architektur zur Beschreibung aktueller und zukünftiger Systeme wurde durch die Modellierung ausgewählter Features dargestellt.

Auf Grundlage der vorgestellten logischen Systemarchitektur wurden in Kapitel 5 die aktuellen Herausforderungen an das Automotive Systems Engineering formuliert. Um diesen zu begegnen wurde ein neues Konzept vorgestellt, welches das etablierte Vorgehen um neue Ansätze und Methoden ergänzt. Kern dieses Konzepts bildet die in Kapitel 6 vorgestellte Reactive-Replay Methode. Sie ermöglicht eine nahtlose Überführung von Szenarien und Situationen, die während der Erprobung auf der realen Straße erlebt wurden, in die Simulation. Damit können erhebliche Modellierungs-, Spezifikations- und Parametrierungsaufwände in der virtuellen Absicherung von Fahrzeugregelungsfunktionen eingespart werden. Durch die virtuelle Wiederholung durchgeführter Erprobungen können unterschiedliche Versionen und Varianten der entwickelten Softwareeinheiten und -komponenten ohne maßgebliche Mehraufwände gegen eine Vielzahl von Szenarien verifiziert werden. Erfahrungswerte zeigen eine signifikante Reduktion der operativen Testaktivitäten im realen Fahrzeug.

In Kapitel 7 wurde der systematische Einsatz der Reactive-Replay Methode untersucht. Neben der Verwaltung der Daten wurden Methoden zur Analyse und Auswahl geeigneter Testszenarien vorgestellt. Die präsentierten Visualisierungen und Filter unterstützen die Entwickler bei der Identifikation geeigneter Testszenarien. Um eine automatisierte Bewertung der Simulationsergebnisse zu ermöglichen, wurde eine schlanke Regressionsteststrategie in den Entwicklungsprozess integriert, die auf Basis eines Ergebnisvergleichs umgesetzte Änderungen gegen unerwünschte Auswirkungen absichert. Die vorgestellte kontinuierliche Anforderungsprüfung analysiert die Simulationsergebnisse über zustandsorientierte Prüfbedingungen. Abgeschlossen wurde die Arbeit mit einer Untersuchung zur Auswahl eines ausgewogenen Satzes von Testszenarien, ohne dabei die Testabdeckung zu verringern. Der vorgestellte zweistufige Ansatz vereint dabei die spezifikationsbasierte Auswahl mittels Klassifikationsbaum mit einem neuartigen Ansatz zur datengetriebenen Reduktion. Dieser ermöglicht die Identifikation von Szenarien mit speziellen Situationen und Sequenzen, die in dieser Art und Ausprägung im Datensatz einzigartig sind.

8.2 Wissenschaftlicher Beitrag

Aufbauend auf einer ganzheitlichen Analyse der etablierten Prozesse und Methoden des Automotive Systems Engineering wurde in der vorliegenden Dissertation ein neues Konzept für die Entwicklung und den Test prädiktiver Fahrzeugregelungsfunktionen entwickelt, umgesetzt und erprobt. Folgende Ergebnisse stellen den wissenschaftlichen Beitrag der Arbeit dar:

- Die aus der Taxonomie aktueller und zukünftiger Kundenfunktionen abgeleitete konzeptuelle logische Systemarchitektur stellt aufgrund der ganzheitlichen Systembetrachtung eine Neuerung dar, da eine gesamtheitliche Betrachtung funktionaler Elemente sämtlicher Fahrzeugdomänen enthalten ist. Die eingeführten Abstraktionsebenen ermöglichen die Identifikation geeigneter Entwicklungsmethoden und -vorgehensweisen.

- Durch die Gegenüberstellung der etablierten Methoden und Prozesse und der Features höherer Abstraktionsebenen konnten die offenen Herausforderungen bei der Entwicklung prädiktiver Fahrzeugregelungsfunktionen identifiziert werden.

- Die entwickelte Reactive-Replay Methode ermöglicht erstmalig die Verwendung von aufgezeichneten Erprobungsdaten aus Testfahrten für die simulierte Ausführung von Kundenfunktionen mit geschlossener Regelschleife. Durch die Reduktion von Modellierungs- und Parametrierungsaufwänden trägt sie substantiell zur Stärkung virtueller Methoden in der Produktentwicklung bei.

- Die vorgestellte Regressionsteststrategie realisiert eine schlanke Prozessintegration der Reactive-Replay Methode, die einen hohen Automatisierungsgrad aufweist.

- Die prototypisch realisierte Anforderungsprüfung auf Basis von Prüfbedingungen ermöglicht einen Übergang von der expliziten und sequentiellen Ausführung einzelner Testfälle hin zur kontinuierlichen Bewertung der Konformität.

- Der evaluierte Ansatz, die Auswahl der Testszenarien auf Basis der Parameterüberdeckung durchzuführen, überträgt Ansätze der explorativen Datenanalyse auf die Fragestellungen des Automotive Systems Engineering.

8.3 Ausblick

In dieser Dissertation lag der Fokus auf der Erforschung und Entwicklung der Reactive-Replay Methode im Kontext der prädiktiven Fahrzeuglängsregelung. In folgenden Arbeiten können die Übertragbarkeit der Methode auf weitere Features evaluiert und notwendige Anpassungen identifiziert und umgesetzt werden. Um darüber hinaus das volle Potential der Reactive-Replay Methode auszuschöpfen, sollten statistische und explorative Vorgehensweisen sowie Methoden untersucht werden, die Auswertungen auf großen Datensätzen unterstützen. Erste Konzepte und Umsetzungen dazu werden in [268] vorgestellt.

Ein besonderer Fokus kann auf die Chancen und Möglichkeiten verschiedener Datenanalyse- und -explorationsverfahren gelegt werden, um in den Daten enthaltene und bisher unbekannte Abhängigkeiten des Funktionsverhaltens zu identifizieren. Beispielsweise ermöglichen Methoden zur Klassifizierung der Daten die Bereitstellung für die Funktionsentwickler hilfreicher Kontextinformtionen. Diese dienen als Grundlage für die Definition von Bewertungsmetriken und für die Auswahl spezifischer Testszenarien. Die Anwendung von Verfahren zur Optimierung komplexer Systeme, wie z.B. die Entwurfsraumexploration, können einen Beitrag zur länderspezifischen Anpassung von Applikationsparametern liefern.

Extrahierte Kontextinformationen können zusätzlich eine Abstraktion der Testszenarien ermöglichen, die eine Zuordnung der Aufzeichnungen zu spezifizierten logischen oder funktionalen Szenarien ermöglichen. Wird diese Abstraktion ohne den Verlust relevanter Informationen und Beziehungen erreicht, können aus den Aufzeichnungen generalisierte Beschreibungen von Testszenarien zur Anwendung in unterschiedlichen Simulationsumgebungen (z.B. entsprechend der OpenSCENARIO Spezifikation) abgeleitet werden.

In [JB6] werden darüber hinaus Chancen zur Nutzung der in den Erprobungen aufgezeichneten Fahrzeugdaten über alle Entwicklungsphasen hinweg diskutiert. Um diese systematisch und effizient zu verwerten, bedarf es der Entwicklung von auf den Automobilsektor zugeschnittenen Lösungen, die eine Analyse und Beantwortung der spezifischen Fragestellungen einzelner Entwicklungsphasen unterstützen.

A Abkürzungsverzeichnis

ABS	Antiblockiersystem
ACC	Abstandsregeltempomat (Adaptive Cruise Control)
ADC	Analog-Digital-Wandler (Analog-to-Digital Converter)
ADTF	Automotive Data and Time-Triggered Framework
API	Schnittstelle zur Anwendungsprogrammierung (Application Programming Interface)
ASE	Automotive Systems Engineering
ASIL	Automotive Safety Integrity Level
CAN	Controller Area Network
CPU	Prozessor (Central Processing Unit)
DAC	Digital-Analog-Wandler (Digital-to-Analog Converter)
DDT	Fahraufgabe (Dynamic Driving Task)
DOE	Design of Experiments
E/E	Elektrik/Elektronik
ECU	Steuergerät (Electronic Control Unit)
EPS	Elektrisch Angetriebene Servolenkung (Electric Power Assisted Steering)
ESC	Elektronische Stabilitätskontrolle (Electronic Stability Control)
FPGA	Field Programmable Gate Array
GNSS	Globales Navigationssatellitensystem (Global Navigation Satellite System)
GPU	Grafikprozessor (Graphics Processing Unit)
HIL	Hardware-in-the-Loop
HMI	Mensch-Maschine-Schnittstelle (Human Machine Interface)
IDE	integrierten Entwicklungsumgebung (Integrated Development Environment)
LIN	Local Interconnect Network
LKA	Spurhalteassistent (Lane Keep Assist)

MBSE modellbasierter Systementwurf (Model-Based Systems Engineering)
MDA modellgetriebene Architektur (Model-Driven Architecture)
MDD modellgetriebene Entwicklung (Model-Driven Development)
MDE modellgetriebener Systementwurf (Model-Driven Engineering)
MIL Model-in-the-Loop
MOST Media Oriented Systems Transport
MPC Modellprädiktive Regelung (Model Predictive Control)
ODD vorgesehene Betriebsbedingung (Operational Design Domain)
OEM Erstausstatter (Original Equippment Manufacturer)
OMG Object Management Group
PCC Prädiktiver Abstandsregeltempomat (Predictive Cruise Control)
PEP Produktentstehungsprozess
PIL Prozessor-in-the-Loop
RAM Random-Access Memory
ROM Read-Only Memory
ROS Robot Operating System
RP Rapid Prototyping
SIL Software-in-the-Loop
SOP Start der Produktion (Start of Production)
SWC Softwarekomponente (Software Component)
SysML Systems Modeling Language
UML Unified Modeling Language
XiL X-in-the-Loop

B Abbildungsverzeichnis

C Tabellenverzeichnis

D Definitionsverzeichnis

E Literaturverzeichnis

[1] W. Thierse, Hrsg. *Traditionswahrung und Modernisierung - Sozialdemokratie in der Entscheidung.* 140 Jahre Gründung von Lassalles Allgemeinem Deutschen Arbeiterverein 1863 in Leipzig - zur Frühgeschichte der deutschen Sozialdemokratie: Dokumentation einer Veranstaltung am 19. Mai 2003 in der Alten Handelsbörse in Leipzig. 2003. url: http://library.fes.de/fulltext/historiker/01705-03.htm#vortrag (besucht am 12. 10. 2017).

[2] C. Brünglinghaus. „Elektronik und Software beherrschen Innovationen im Auto". In: *Automobilelektronik und Softwre* (2014). url: https://www.springerprofessional.de/automobilelektronik---software/antriebsstrang/elektronik-und-software-beherrschen-innovationen-im-auto/6561802 (besucht am 21. 07. 2016).

[3] K. Reif. *Automobilelektronik.* 5. Aufl. Springer Vieweg, 2014. isbn: 3-658-05047-4.

[4] H.-C. Dirschler. *Google Maps im Test: Gratis-Navigation mit exakten Verkehrslage-informationen.* Hrsg. von P. Welt. 2017. url: https://www.pcwelt.de/produkte/So-gut-arbeitet-Google-Maps-Navigation-Tolle-Gratis-Navi-App-446956.html (besucht am 19. 10. 2017).

[5] Verband der Automobilindustrie VDA, Hrsg. *Politikbrief 01/2015 - Digitalisierung und Mobilität wachsen zusammen.* 2017. url: https://www.vda.de/de/services/Publikationen/vda-politikbrief-01-2015.html (besucht am 15. 09. 2017).

[6] C. Ebert und J. Favaro. „Automotive Software". In: *IEEE Software* 34.3 (2017), S. 33–39. issn: 0740-7459. doi: doi.ieeecomputersociety.org/10.1109/MS.2017.82.

[7] W. Wachenfeld und H. Winner. „Die Freigabe des autonomen Fahrens". In: *Autonomes Fahren* (2015). Hrsg. von M. Maurer u. a., S. 439–464.

[8] J. Mazzega u. a. „Absicherung hochautomatisierter Fahrfunktionen". In: *ATZ - Automobiltechnische Zeitschrift* (Okt. 2016), S. 48–52.

[9] I. Renner. „Methodische Unterstützung funktionsorientierter Baukastenentwicklung am Beispiel Automobil". Diss. München: Technische Universität München, Mai 2007.

[10] P. Bourque, R. E. Fairley u. a. *Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0.* IEEE Computer Society Press, 2014.

[11] J. Weber. *Automotive Development Process*. Springer-Verlag, 2009. isbn: 3-642-01252-5. doi: 10.1007/978-3-642-01253-2.

[12] M. E. Conway. „How do committees invent“. In: *Datamation* 14.4 (1968), S. 28–31.

[13] D. Reinhardt und M. Kucera. „Domain Controlled Architecture - A New Approach for Large Scale Software Integrated Automotive Systems“. In: *3rd International Conference on Pervasive Embedded Computing and Communication Systems*. 2013, S. 221–226. doi: 10.5220/0004340702210226.

[14] J. Schäuffele und T. Zurawka. *Automotive Software Engineering - Grundlagen, Prozesse, Methoden und Werzeuge effizient einsetzen*. 5. Aufl. Springer Fachmedien Wiesbaden GmbH, 2012.

[15] T. Streichert und M. Traub. *Elektrik/Elektronik-Architekturen im Kraftfahrzeug: Modellierung und Bewertung von Echtzeitsystemen*. Springer-Verlag, 2012.

[16] Gemeinschaftswerk, siehe Autorenliste. *Bosch Automotive Electrics and Automotive Electronics*. Bd. 5. Plochingen: Robert Bosch GmbH, 2007.

[17] Wikipedia-Autoren, siehe Versionsgeschichte. *Automobilfertigung*. 2017. url: https://de.wikipedia.org/w/index.php?title=Automobilfertigung&oldid=161324545 (besucht am 06. 01. 2017).

[18] K. Sattler. „Methodik für den Systemtest in der integralen Fahrzeugsicherheit“. Diss. Universität Magdeburg, 2015.

[19] E. Sax. *Automatisiertes Testen Eingebetteter Systeme in der Automobilindustrie*. München: Hanser, Carl, 2008. isbn: 9783446416352 3446416358.

[20] W. Haas und P. Langjahr. „Cross-domain vehicle control units in modern E/E architectures“. In: *16. Internationales Stuttgarter Symposium*. 2016, S. 1619–1627.

[21] „Systems and software engineering - Vocabulary“. In: *ISO/IEC/IEEE 24765:2010(E)* (Dez. 2010), S. 1–418. doi: doi:10.1109/IEEESTD.2010.5733835.

[22] SAE international. *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. Sep. 2016.

[23] Verband der Automobilindustrie VDA, Hrsg. *Automatisiertes Fahren*. 2016. url: https://www.vda.de/de/themen/innovation-und-technik/automatisiertes-fahren/automatisiertes-fahren.html (besucht am 24. 03. 2016).

[24] *VDA Magazin - Automatisierung*. Verband der Automobilindustrie e. V., 2015.

[25] W. W. Royce. „Managing the development of large software systems“. In: *Proceedings of IEEE WESCON 26*. Aug. 1970, S. 1–9.

[26] T. Bell und T. T. „Software requirements: Are they really a problem?“ In: *Proceedings of the 2nd international conference on Software engineering*. 1976.

[27] B. Boehm. *Software Risk Management*. Hrsg. von I. C. Society. Massachusetts: IEEE Computer Society Press, 1989. isbn: 0-8186-8906-4.

[28] „ISO/IEC/IEEE Standard for Systems and Software Engineering - Software Life Cycle Processes“. In: *ISO/IEC/IEEE 12207:2008(E)* (Jan. 2008), S. c1–138. doi: 10.1109/IEEESTD.2008.4475826.

[29] B. W. Boehm. „A spiral model of software development and enhancement“. In: *Computer* 21.5 (Mai 1988), S. 61–72. issn: 0018-9162. doi: 10.1109/2.59.

[30] B. Boehm. *Spiral Development: Experience, Principles, and Refinements*. Techn. Ber. CMU/SEI-2000-SR-008. Software Engineering Institute, Feb. 2000. url: http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=5053 (besucht am 01. 03. 2016).

[31] *V-Modell 97 - komplett im Format Word (Teil 1-3)*. IABG Industrieanlagen-Betriebsgesellschaft mbH. Ottobrunn, Germany, Juni 1997. url: http://v-modell.iabg.de/ (besucht am 05. 03. 2016).

[32] *V-Modell XT*. Bonn, Deutschland: Informationstechnikzentrum Bund. url: https://www.itzbund.de/DE/Produkte/V-Modell-XT/v-modell-xt_node.html (besucht am 25. 07. 2016).

[33] B. Broekman und E. Notenboom. *Testing Embedded Software*. Addison-Wesley, 2002.

[34] B. Schick und J. Henning. „Simulation methods to evaluate and verify functions, quality and safety of driver assistance systems in the continuous Mil, SiL and HiL process“. In: *3rd Conference Active Safety through Driver Assistance*. Apr. 2008.

[35] C. Ebert u. a. *Introducing automotive E/E safety engineering: Challenges and solutions*. Vector Consulting Services GmbH. Stuttgart, 2011.

[36] S. Burton und C. J. *Kombinierte Sicherheit*. Hrsg. von elektroniknet.de. 2012. url: http://www.elektroniknet.de/automotive/assistenzsysteme/artikel/93419/ (besucht am 22. 07. 2016).

[37] B. Deutschland, Hrsg. *V-Modell XT - häufig gestellte Fragen*. 2016. url: http://www.cio.bund.de/Web/DE/Architekturen-und-Standards/V-Modell-XT/Haeufig-gestellte-Fragen/haeufig_gestellte_fragen_node.html (besucht am 21. 07. 2016).

[38] *V-Modell XT*. 1.4. IABG Industrieanlagen-Betriebsgesellschaft mbH. Ottobrunn, Germany, 2006. url: http://v-modell.iabg.de/ (besucht am 05. 03. 2016).

[39] C. Bartelt u. a. *V-Modell XT - Das deutsche Referenzmodell für Systementwicklungsprojekte*. 2.0. Verein zur Weiterentwicklung des V-Modell XT e.V. (Weit e.V.), c/o 4Soft GmbH. München, Germany, Aug. 2015. url: http://weit-verein.de/ (besucht am 05. 03. 2016).

[40] R. Cooper. „Perspective: The Stage-Gate Idea-to-Launch Process - Update, What's New and NexGen Systems“. In: *Journal of Product Innovation Management* 25.3 (Mai 2008), S. 213–232.

[41] F. Billing. „Koordination in radikalen Innovtionsvorhaben“. Diss. Technische Universität Berlin, 2002.

[42] S. Rietz. „Prozessmanagement: mit Engineering-Standads“. In: *Prozessmanagement: Strategien, Methoden, Umsetzung* 1 (2010), S. 255–289.

[43] Defense Acquisition University, Hrsg. *Integrated Defense Acquisition, Technology, & Logistics Life Cycle Management Framework*. 2016. url: http://www.public.navy.mil/spawar/PEOC4I/ASPG/Documents/APSG_Manuals/files/Integrated_Def_Acq_Management_Frmwk.pdf (besucht am 22. 12. 2016).

[44] G. Hab und R. Wagner. „Management einzelner Automotive-Projekte („Single-PM“)“. In: *Projektmanagement in der Automobilindustrie* 1 (Okt. 2012), S. 23–194.

[45] V. Q. W. G. 13 und A. SIG. *Automotive SPICE Process Assessment / Reference Model*. 3.0. VDA QMC. Berlin, Germany, Juli 2015. url: http://www.automotivespice.com/ (besucht am 05. 03. 2016).

[46] „Information technology – Process assessment – Process measurement framework for assessment of process capability“. In: *ISO/IEC 33020:2015* (März 2015), S. 1–18.

[47] S. J. und K. Schwaber. *The Scrum Guide*. Scrum.org. Boston, Massachusetts (USA), Juli 2016. url: http://www.scrumguides.org/ (besucht am 05. 03. 2016).

[48] K. Beck u. a., Hrsg. *Manifest für Agile Softwareentwicklung*. 2016. doi: http://agilemanifesto.org/iso/de/. (Besucht am 02. 08. 2016).

[49] D. (o.J.) *duden.de*. 2017. url: http://www.duden.de/ (besucht am 20. 09. 2017).

[50] *What is Scrum?* Scrum.org. url: https://www.scrum.org/Resources/What-is-Scrum (besucht am 02. 08. 2016).

[51] Wikipedia-Autoren, siehe Versionsgeschichte. *Disruptive Technologie*. 2017. url: https://de.wikipedia.org/wiki/Disruptive_Technologie (besucht am 22. 09. 2017).

[52] A. López u. a. „Nighttime Vehicle Detection for Intelligent Headlight Control“. In: *Advanced Concepts for Intelligent Vision Systems: 10th International Conference, ACIVS 2008, Juan-les-Pins, France, October 20-24, 2008. Proceedings* (Mai 2008), S. 113–124. doi: 10.1007/978-3-540-88458-3_11.

[53] J. Lunze. *Regelungstechnik 1*. 10. Aufl. Springer Vieweg, 2014.

[54] „Software and systems engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models“. In: *ISO/IEC 25010:2011* (März 2011), S. 1–34.

[55] A. Spillner und T. Linz. *Basiswissen Softwaretest*. Bd. 3. Heidelberg: dpunkt.verlag, 2005. isbn: 3-89864-358-1.

[56] I. Burnstein. *Practical Software Testing: A Process-Oriented Approach*. Springer Professional Computing. Springer New York, 2006. isbn: 9780387216584.

[57] „Software and systems engineering - Software testing - Part 1: Concepts and Definitions“. In: *ISO/IEC/IEEE 29119-1:2013* (Sep. 2013), S. 1–64. doi: 10.1109/IEEESTD.2013.6588537.

[58] „Software and systems engineering - Software testing - Part 2: Test processes“. In: *ISO/IEC/IEEE 29119-2:2013(E)* (Sep. 2013), S. 1–68. doi: 10.1109/IEEESTD.2013.6588543.

[59] „Road vehicles - Functional safety - Part 6: Product development at the software level“. In: *ISO 26262-6:2011(E)* (Nov. 2011), S. 1–36.

[60] H. Stachowiak. *Allgemeine Modelltheorie*. Wien - New York: Springer-Verlag, 1973.

[61] M. Törngren u. a. „Model-based development of automotive embedded systems“. In: (2008), S. 258–309.

[62] N. Aeronautics und S. Administration, Hrsg. *Langley Formal Methods Program - What is Formal Methods?* 2016. url: https://shemesh.larc.nasa.gov/fm/fm-what.html (besucht am 06. 05. 2017).

[63] D. Long und Z. Scott. *A primer for model-based systems engineering*. Lulu. com, 2011.

[64] M. Brambilla, J. Cabot und M. Wimmer. *Model-Driven Software Engineering in Practice (Synthesis Lectures on Software Engineering)*. Morgan & Laypool, 2012.

[65] N. Marko u. a. „Model-based engineering for embedded systems in practice“. In: *Research reports in software engineering and management* 1 (2014).

[66] *OMG Unified Modeling Language (OMG UML)*. Object Management Group. 2013.

[67] I. Jacobson, G. Booch und J. Rumbaugh. *The Unified Software Development Process*. Boston, MA: Addison-Wesley, 1999. isbn: 9783446416352 3446416358.

[68] *OMG Systems Modeling Language (OMG SysML)*. Object Management Group. 2012.

[69] J. Broy. „Modellbasierte Entwicklung und Optimierung flexibler zeitgesteuerter Architekturen im Fahrzeugserienbereich“. Diss. Karlsruhe: Karlsruher Institut für Technologie, 2010.

[70] Vector Informatik GmbH. *PREEvision User Manual Version 6.5*. Stuttgart, 2013.

[71] AUTOSAR development cooperation. *Specification of RTE*. 4.2.1. Munich, Juli 2015. url: https://www.autosar.org/specifications/release-42/ (besucht am 10. 11. 2016).

[72] *Handbuch Kraftfahrzeugelektronik*. Wallentowitz H. und Reif, K., 2006.

[73] A. Angermann u. a. *Matlab - Simulink - Stateflow*. Oldenbourg Wissenschaftsverlag GmbH, 2014.

[74] C. Rossi u. a. „Achievements with model-based development on the innovative traction system of the AMBER-ULV Car“. In: *16th Stuttgart International Symposium Automotive and Engine Technology*. Hrsg. von F. für Kraftfahrwesen und Fahrzeugmotoren Stuttgart. Bd. 2. Springer Fachmedien Wiesbaden GmbH, März 2016, S. 237–251.

[75] *Embedded Coder*. The MathWorks, Inc. Natick, Massachusetts, 2016. url: https://de.mathworks.com/products/embedded-coder/ (besucht am 03. 11. 2016).

[76] *TargetLink*. dSPACE GmbH. Paderborn, 2016. url: https://www.dspace.com/en/inc/home/products/sw/pcgs/targetli.cfm (besucht am 03. 11. 2016).

[77] D. Abel und A. Bolling. *Rapid Control Prototyping*. Springer, 2006. isbn: 3-540-29524-2.

[78] M. Traub, A. Maier und K. Barbehon. „Future Automotive Architecture and the Impact of IT Trends“. In: *IEEE Software* 34.3 (2017), S. 27–32. issn: 0740-7459. doi: doi.ieeecomputersociety.org/10.1109/MS.2017.69.

[79] M. Woods u. a. „Seeker - Autonomous Long-range Rover Navigation for Remote Exploration“. In: *Journal of Field Robotics* 31.6 (2014), S. 940–968. issn: 1556-4967. doi: 10.1002/rob.21528. url: http://dx.doi.org/10.1002/rob.21528.

[80] *dSPACE Prototyping Systems*. dSPACE GmbH. Paderborn, 2016. url: https://www.dspace.com/de/gmb/home/products/systems/functp.cfm (besucht am 03. 11. 2016).

[81] *Hardware Netzwerk-Interfaces*. Vector Informatik GmbH. Stuttgart, 2016. url: https://vector.com/vi_interfaces_de.html (besucht am 03. 11. 2016).

[82] *EB Assist ADTF 2.11.0 - User's Manual*. 2.11.0. Elektrobit Automotive GmbH. Erlangen, 2014.

[83] *About ROS*. Open Source Robotics Foundation, 2016. url: http://www.ros.org/about-ros/ (besucht am 03. 11. 2016).

[84] C. Urmson u. a. *Tartan Racing: A Multi-Modal Approach to the DARPA Urban Challenge*. Techn. Ber. Carnegie Mellon University, 2007.

[85] M. Montemerlo u. a. „Junior: The Stanford entry in the Urban Challenge“. In: *Journal of Field Robotics* 25.9 (2008), S. 569–597. issn: 1556-4967. doi: 10.1002/rob.20258. url: http://dx.doi.org/10.1002/rob.20258.

[86] C. Urmson u. a. „Autonomous Driving in Traffic: Boss and the Urban Challenge“. In: *AI Magazine* 30.2 (2009), S. 17.

[87] J. Levinson u. a. „Towards fully autonomous driving: Systems and algorithms“. In: *Intelligent Vehicles Symposium (IV), 2011 IEEE*. Juni 2011, S. 163–168. doi: 10.1109/IVS.2011.5940562.

[88] T. Nothdurft u. a. „Stadtpilot: First fully autonomous test drives in urban traffic“. In: *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. Aug. 2011, S. 919–924. doi: 10.1109/ITSC.2011.6082883.

[89] D. Lenz u. a. „Mehrstufiges Planungskonzept für pilotierte Parkhausfunktionen“. In: *30. VDI/VW-Gemeinschaftstagung "Fahrerassistenz und Integrierte Sicherheit 2014"*. Wolfsburg, Germany, 2014.

[90] O. Pink, J. Becker und S. Kammel. „Automated Ddriving on public rroad: Experiences in real traffic“. In: *it - Information Technology* 57.4 (Aug. 2015), S. 223–230. doi: 10.1515/itit-2015-0010.

[91] M. Aeberhard u. a. *Automated Driving with ROS at BMW*. 2016. url: http://roscon.ros.org/2015/presentations/ROSCon-Automated-Driving.pdf (besucht am 18. 12. 2016).

[92] A. Kecskeméthy. „Mehrkörpersimulation“. In: *Essener Unikate* 31 (2007).

[93] P. R. Muessig, D. R. Laack und J. J. Wrobleski. „An Integrated Approach to Evaluating Simulation Credibility“. In: *U.S. Naval Air Warfare Center* (2001).

[94] J. Craighead u. a. „A Survey of Commercial & Open Source Unmanned Vehicle Simulators“. In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. Roma, Apr. 2007, S. 852–857.

[95] E. Onieva u. a. „A modular parametric architecture for the torcs racing engine“. In: *IEEE Symposium on Computational Intelligence and Games*. 2009, S. 256–262,

[96] C. Schmidt. „Harware-in-the-Loop gestützte Entwicklungsplattform für Fahrerassistenzsysteme, Analyse und Generierung kritischer Verkehrsszenarien“. Diss. Universität Kassel, 2010.

[97] I. Shimchik u. a. „Golf cart prototype development and navigation simulation using ROS and Gazebo“. In: *2016 International Conference on Measurement Instrumentation and Electronics (ICMIE 2016)*. Munich, Juni 2016.

[98] *IPG Documentation - CarMaker - User's Guide Version 4.5.2*. 4.5.2. IPG Automotive GmbH. Karlsruhe, 2014.

[99] *VIRES Virtual Test Drive*. VIRES Simulationstechnologie GmbH. Bad Aibling, 2014.

[100] R. Molenaar, A. van Bilsen und R. van der Made. „Full Spectrum Camera Simulation for Reliable Virtual Development and Validation of ADAS and Automated Driving Applications“. In: *2015 IEEE Intelligent Vehicles Symposium (IV)*. Juni 2015, S. 47–52.

[101] M. Dupuis, M. Strobl und H Grezlikowski. „OpenDRIVE 2010 and Beyond -Status and future of the de facto standard for the description of road networks“. In: *Trends in driving simulation design and experiments: (proceedings of the Driving simulation conference Europe 2010)*. Sep. 2010, S. 231–242.

[102] G. Bagschik u. a. „Szenarien für Entwicklung, Absicherung und Test von automatisierten Fahrzeugen“. In: *Workshop Fahrerassistenzsysteme und automatisiertes Fahren*. Bd. 11. Walting im Altmühltal, März 2017.

[103] S. Geyer u. a. „Concept and development of a unified ontology for generating test and use-case catalogues for assisted and automated vehicle guidance“. In: *IET Intelligent Transport Systems*. 8. Ser. 8.3 (2013), S. 183–189. url: http://tubiblio.ulb.tu-darmstadt.de/62301/.

[104] Z. Xiong. „Creating a Computing Environment in a Driving Simulator to Orchestrate Scenarios with Autonomous Vehicles“. Diss. The University of Leeds, Institute for Transport Studies & School of Computing, 2013.

[105] S. Ulbrich u. a. „Defining and Substantiating the Terms Scene, Situation, and Scenario for Automated Driving“. In: *Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on*. Sep. 2015, S. 982–988. doi: 10.1109/ITSC.2015.164.

[106] VIRES Simulationstechnologie GmbH, Hrsg. *OpenSCENARIO*. 2016. url: http://www.openscenario.org/index.html (besucht am 18. 12. 2016).

[107] T. Helmer u. a. „Safety Performance Assessment of Assisted and Automated Driving by Virtual Experiments: Stochastic Microscopic Traffic Simulation as Knowledge Synthesis“. In: *Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on*. IEEE. 2015, S. 2019–2023.

[108] S. Khastgir u. a. *Test Scenario Generation for Driving Simulators Using Constrained Randomization Technique*. Techn. Ber. SAE Technical Paper, 2017.

[109] F. Sazama. *Agile in Automotive - Scrum Pocket Guide*. Bd. 3. Kugler Maag Cie GmbH, 2015.

[110] P. Duvall, S. Matyas und A. Glover. *Continuous Integration*. Hrsg. von D. Kindersley. Pearson Education, Inc., 2008.

[111] J. Humble und D. Farley. *Continuous Delivery*. Addison-Wesley, 2010.

[112] M. Feilhauer, J. Häring und B. J. „Continuous delivery for simulation-model development“. In: *16th Stuttgart International Symposium Automotive and Engine Technology*. Hrsg. von F. für Kraftfahrwesen und Fahrzeugmotoren Stuttgart. Bd. 1. Springer Fachmedien Wiesbaden GmbH, März 2016, S. 467–477.

[113] „IEEE Standard for Software Reviews and Audits“. In: *IEEE Std 1028* (Aug. 2008). doi: 10.1109/IEEESTD.2008.4601584.

[114] S. Sijbrandij. *The 11 rules of gitlab flow*. 2016. url: https://about.gitlab.com/2016/07/27/the-11-rules-of-gitlab-flow/ (besucht am 18. 12. 2016).

[115] SmartBearSoftware, Hrsg. *What is Code Review?* 2016. url: https://smartbear.com/learn/code-review/what-is-code-review/ (besucht am 02. 10. 2016).

[116] D. Mytton. *Why You Need Coding Standards*. 2004. url: https://www.sitepoint.com/coding-standards/ (besucht am 02. 10. 2016).

[117] D. Wells. *Extreme Programming: A gentle introduction*. 2009. url: http://www.extremeprogramming.org/index.html (besucht am 02. 10. 2016).

[118] W. Wögerer. *A survey of static program analysis techniques*. Techn. Ber. Technische Universität Wien, 2005.

[119] *The Leader in Static Analysis for C/C++ – PC-lint and FlexeLint*. Gimpel Software LLC. Collegeville, Pennsylvania, 2016. url: http://www.gimpel.com/html/index.htm (besucht am 19. 12. 2016).

[120] I. Google. *Google C++ Style Guide*. url: https://google.github.io/styleguide/cppguide.html (besucht am 07. 10. 2016).

[121] *MISRA-C:2004 Guidelines for the use of the C language in critical systems*. MIRA Limited. 2004.

[122] H. Kuder. „HIS Source Code Metrics“. In: *Hersteller Initiative Software - AK Softwaretest* 1.3.1 (Apr. 2008), S. 1–8. doi: 10.1109/IEEESTD.2013.6588537.

[123] P. Bleicher. *Automate Your Code Reviews with Static Code Analysis*. 2016. url: http://blog.codacy.com/2016/02/08/automate-your-code-reviews-with-codacy/ (besucht am 02. 10. 2016).

[124] U. Hafner und M. Rust. *Static Code Analysis Plug-ins*. 2016. url: https://wiki.jenkins-ci.org/display/JENKINS/Static+Code+Analysis+Plug-ins (besucht am 02. 10. 2016).

[125] *Comprehensive Static Analysis Using Polyspace Products*. The MathWorks, Inc. Natick, Massachusetts, 2013. url: https://de.mathworks.com/products/polyspace/ (besucht am 05. 03. 2016).

[126] H. Shokry und M. Hinchey. „Model-Based Verification of Embedded Software“. In: *Computer* 42.4 (Apr. 2009), S. 53–59.

[127] A. D. Junior und D. C. da Silva. „Code-coverage Based Test Vector Generation for SystemC Designs“. In: *IEEE Computer Society Annual Symposium on VLSI (ISVLSI '07)*. März 2007, S. 198–206.

[128] S. Grünfelder. *Software-Test für Embedded Systems*. Bd. 1. Heidelberg: dpunkt.Verlag, 2013.

[129] A. Noack, A. Bertl und B. Eggl. „Fahrerassistenzsysteme - Validierung von Hochband-Sensorik“. In: *ATZ elektronik* (Feb. 2016), S. 54–59.

[130] A. Geiger u. a. „Vision Meets Robotics: The KITTI Dataset“. In: *International Journal of Robotics Research* 32.11 (Sep. 2013), S. 1231–237.

[131] M. Cordts u. a. „The Cityscapes Dataset for Semantic Urban Scene Understanding“. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016. url: https://www.cityscapes-dataset.com/ (besucht am 21. 04. 2016).

[132] I. Passchier, G. van Vugt und M. Tideman. „An integral approach to autonomous and cooperative vehicles development and testing“. In: *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. Sep. 2015, S. 348–352.

[133] A. Albers u. a. „X-in-the-Loop-Framework für Fahrzeuge, Steuergeräte und Kommunikationssysteme“. In: *ATZ elektronik* (Mai 2010), S. 60–65.

[134] K. Neumann-Cosel. „Virtual Test Drive - Simulation umfeldbasierter Fahrzeugfunktionen“. Diss. Technische Universität München, Fakultät für Informatik, 2013. url: http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20140206-1126934-0-2.

[135] D. Tuscherer, A. Weibert und F. Tränkle. „Modern C++ as a Modelling Language for Automated Driving and Human-Robot Collaboration“. In: *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems*. MODELS ’16. Saint-malo, France: ACM, Okt. 2016, S. 136–142.

[136] T. Fischer. „Automatisierter Test und Analyse eingebetteter Software“. Diss. Karlsruher Institut für Technologie, 2016. doi: todo.

[137] M. Reiff. „Radikale Innovationen in der Mobilität“. In: Proff, H., 2014. Kap. HIL-Optimierung der Regelgüte eines Kaskadenreglers zur Ansteuerung eines passiven elektroschen Phasenstellers im Anwendungsbetrieb eines Verbennungsmotors, S. 299–310.

[138] C. Wohlfahrt. „Von Systematischer Absicherung zur Digitalen Erprobungsfahrt“. In: *IBS - Workshop Automotive Software Engineering - Virtuelle Absicherung*. Juni 2016.

[139] M. Miegler u. a. „Hardware-in-the-Loop-Test von vorausschauenden Fahrerassistenzsystemen“. In: *ATZ elektronik* (Mai 2009), S. 60–65.

[140] L. Brader, H. Hilliker und A. C. Wills. *Testing for Continuous Delivery with Visual Studio 2012*. Redmond, Washington: Microsoft, Mai 2012. isbn: 1-62114-019-1.

[141] T. Ringler. „Virtual Integrtion and Test of AUTOSAR Systems at Daimler“. In: *7. Vector Congress 2014*. Stuttgart, Nov. 2014. url: https://vector.com/vi_veco14_downloads_de.html (besucht am 03. 11. 2016).

[142] I. Zöller u. a. „Fahrsimulatorvalidität - Systematisierung und quantitative Analyse bisheriger Forschungen“. In: *Zeitschrift für Arbeitswissenschaft* 67.4 (Dez. 2013), S. 197–206.

[143] T. Geiger. *So testet Mercedes die neue C-Klasse*. 2006. url: https://www.welt.de/motor/article703156/So-testet-Mercedes-die-neue-C-Klasse.html (besucht am 22. 12. 2016).

[144] S. Scherer u. a. „How the Driver Wants to be Driven - Modelling Driving SStyle in Highly Automated Driving“. In: *7. Tagung Fahrerassistenzsysteme*. Nov. 2015.

[145] L. Lorenz u. a. „Der Fahrer im Hochautomatisierten Fahrzeug. Vom Dual-Task zum Sequential-Task PParadigm.“ In: *7. Tagung Fahrerassistenzsysteme*. Nov. 2015.

[146] A. Zlocki u. a. „Evaluation of Automated Road Vehicles“. In: *Road Vehicle Automation, Lecture Notes in Mobility* (2014), S. 49–59.

[147] T. Pretsch und J. Spiegel. „Vernetzte Welt“. In: *Porsche Engineering MAGAZIN* 1 (2014), S. 44–51.

[148] D. Reich u. a. „Scheduling Crash Tests at Ford Motor Company“. In: *Interfaces* 46.5 (2016), S. 409–423.

[149] U. Baake u. a. „Versuchs- und Simulationsbasierte Absicherung von ESP-Systemen für Transporter“. In: *ATZ - Automobiltechnische Zeitschrift* (Feb. 2014), S. 46–52.

[150] E. Lutz und A. Zlocki. „Safety Potential of ADAS: Combined Methods for an Effective Evaluation“. In: *The 23rd International Technical Conference on the Enhanced Safety of Vehicles (ESV)*. Seoul, Republic of Korea, Mai 2013.

[151] Z. Bareket u. a. „Methodology for assessing adaptive cruise control behavior“. In: *IEEE Transactions on Intelligent Transportation Systems* 4.3 (Sep. 2003), S. 123–131. issn: 1524-9050. doi: 10.1109/TITS.2003.821288.

[152] R. Nörenberg. „Effizienter Regressionstest von E/E-Systemen nach ISO 26262“. Diss. Karlsruher Institut für Technologie, 2012. isbn: 978-3-86644-842-1.

[153] S. Grundhoff. „Winterparadies - Wintererprobung am Polarkreis“. In: *Stern.de* (März 2008). url: http://www.stern.de/auto/service/wintererprobung-am-polarkreis-winterparadies-3085642.html (besucht am 22. 12. 2016).

[154] MAN Truck & Bus AG. *Am Limit - Härtetest am Hitzepol*. März 2015. url: http://www.truck.man.eu/de/de/man-welt/man-stories/Am-Limit-_-Haertetest-am-Hitzepol-257158.html (besucht am 22. 12. 2016).

[155] J. Löchner u. a. „Validating advanced driver assistance systems (ADAS) using comprehensive, loss-free in-vehicle measurements“. In: *16th Stuttgart International Symposium Automotive and Engine Technology*. Hrsg. von F. für Kraftfahrwesen und Fahrzeugmotoren Stuttgart. Bd. 2. Springer Fachmedien Wiesbaden GmbH, März 2016, S. 307–318.

[156] „Software and systems engineering - Software testing - Part 4: Test techniques“. In: *ISO/IEC/IEEE 29119-4:2015* (Dez. 2015), S. 1–149. doi: 10.1109/IEEESTD.2015.7346375.

[157] OpenCppCoverage, Hrsg. *OpenCppCoverage - Home*. 2016. url: https://opencppcoverage.codeplex.com/ (besucht am 20. 12. 2016).

[158] Bullseye Testing Technology, Hrsg. *BullseyeCoverage Measurement Technique*. 2016. url: http://www.bullseye.com/measurementTechnique.html (besucht am 20. 12. 2016).

[159] B. J. Gough. *An Introduction to GCC*. Network Theory Ltd., 2004.

[160] S. M. Ooi, R. Lim und C. C. Lim. „An integrated system for end-to-end traceability and requirements test coverage“. In: *2014 IEEE 5th International Conference on Software Engineering and Service Science*. Juni 2014, S. 45–48.

[161] B. W. Boehm. „Software Engineering“. In: *IEEE Transactions on Computers* 25.12 (Dez. 1976), S. 1226–1241. issn: 0018-9340. doi: 10.1109/TC.1976.1674590. url: http://dx.doi.org/10.1109/TC.1976.1674590.

[162] E. Sax. *Tatort Test - Test is not the Last*. 2015. url: https://vector.com/portal/medien/cmc/events/commercial_events/VTS15/VTS15_02_Sax_Vortrag.pdf (besucht am 07. 05. 2017).

[163] D. Wittmann, C. Wang und M. Lienkamp. „Definition and identification of system boundaries of highly autoated driving“. In: *7. Tagung Fahrerassistenzsysteme*. Nov. 2015.

[164] M. Hillenbrand. „Funktionale Sicherheit nach ISO 26262 in der Konzeptphase der Entwicklung von Elektrik/Elektronik Architekturen von Fahrzeugen“. Diss. Karlsruher Institut für Technologie, 2012. isbn: 978-3-86644-803-2.

[165] K. Bengler u. a. „Three Decades of Driver Assistance Systems“. In: *IEEE Intelligent Transportation Systems Magazine* 6.4 (2014), S. 6–22.

[166] J. H. Park und C. Y. Kim. „Wheel slip control in traction control system for vehicle stability“. In: *Vehicle system dynamics* 31.4 (1999), S. 263–278.

[167] M. Kulkarni, T. Shim und Y. Zhang. „Shift dynamics and control of dual-clutch transmissions“. In: *Mechanism and Machine Theory* 42.2 (2007), S. 168 –182. url: http://www.sciencedirect.com/science/article/pii/S0094114X06000565.

[168] P. D. Walker, N. Zhang und R. Tamba. „Control of gear shifts in dual clutch transmission powertrains“. In: *Mechanical Systems and Signal Processing* 25.6 (2011), S. 1923 –1936.

[169] J. Zechmann und A. Irion. *Method and apparatus for controlling the brake system of a vehicle*. US Patent 6,009,984. Jan. 2000. url: https://www.google.com/patents/US6009984.

[170] B. Boll u. a. *Hill holder device for a motor vehicle*. US Patent 6,679,810. Jan. 2004. url: https://www.google.com/patents/US6679810.

[171] H. E. Tseng u. a. „The development of vehicle stability control at Ford“. In: *IEEE/ASME Transactions on Mechatronics* 4.3 (Sep. 1999), S. 223–234. issn: 1083-4435. doi: 10.1109/3516.789681.

[172] E. K. Liebemann u. a. „Safety and performance enhancement: The Bosch electronic stability control (ESP)“. In: *SAE Paper* 20004 (2004), S. 21–0060.

[173] A. van Zanten und F. Kost. „Handbuch Fahrerassistenzsysteme“. In: Wiesbaden: ViewegTeubner Verlag, Springer Fachmedien, 2012. Kap. Bremsenbasierte Assistenzfunktionen, S. 356–394.

[174] J.-W. Kim, K.-J. Lee und H.-S. Ahn. „Development of software component architecture for motor-driven power steering control system using AUTOSAR methodology“. In: *Control, Automation and Systems (ICCAS), 2015 15th International Conference on*. Okt. 2015, S. 1995–1998. doi: 10.1109/ICCAS.2015.7364695.

[175] J. E. Naranjo u. a. „Power-steering control architecture for automatic driving“. In: *IEEE Transactions on Intelligent Transportation Systems* 6.4 (Dez. 2005), S. 406–415. issn: 1524-9050. doi: 10.1109/TITS.2005.858622.

[176] K. N. Majeed. *On/off semi-active suspension control*. US Patent 5,062,657. Sep. 1991. url: https://www.google.com/patents/US5062657.

[177] M. Ahmed und F. Svaricek. „Preview optimal control of vehicle semi-active suspension based on partitioning of chassis acceleration and tire load spectra“. In: *2014 European Control Conference (ECC)*. Juni 2014, S. 1669–1674.

[178] M. Wiesenthal, H. Collenberg und H. Krimmel. „Aktive Hinterachskinematik AKC - ein Beitrag zu Fahrdynamik, Sicherheit und Komfort“. In: *17. Aachener Kolloquium Fahrzeug- und Motorentechnik*. 2008.

[179] M. Takagi, T. Asano und T. Yamada. *Automotive vehicle with adjustable aerodynamic accessory and control therefor*. US Patent 4,810,022. März 1989. url: https://www.google.com/patents/US4810022.

[180] W. Seidel. *Process for controlling front or rear spoilers*. US Patent 7,113,855. Sep. 2006. url: https://www.google.com/patents/US7113855.

[181] R. Willats u. a. *Vehicle access control and start system*. US Patent App. 10/348,233. Dez. 2003. url: https://www.google.com/patents/US20030222758.

[182] P. Dix und M. Bojarski. *Reprogrammable vehicle access control system*. US Patent App. 10/602,750. Dez. 2004. url: https://www.google.com/patents/US20040263316.

[183] S. Dimig u. a. *Steering column lock apparatus and method*. US Patent 6,571,587. Juni 2003. url: https://www.google.com/patents/US6571587.

[184] S. M. Prabhu und P. J. Mosterman. „Model-based design of a power window system: Modeling, simulation and validation“. In: *Proceedings of IMAC-XXII: A Conference on Structural Dynamics, Society for Experimental Mechanics, Inc., Dearborn, MI*. 2004.

[185] V. Rabinovich, N. Alexandrov und B. Alkhateeb. *Automotive antenna design and applications*. CRC press, 2010.

[186] R. Broström u. a. „Towards the next generation intelligent driver information system (IDIS): The Volvo car interaction manager concept“. In: *Proceedings of the 2006 ITS World Congress*. 2006, S. 32.

[187] E. Rahimzei, K. Sann und M. Vogel. *Kompendium: Li-Ionen-Batterien*. Techn. Ber. VDE Verband der Elektrotechnik, 2015.

[188] J. Daniel und J.-P. Lauffenburger. „Fusing navigation and vision information with the Transferable Belief Model: Application to an intelligent speed limit assistant“. In: *Information Fusion* 18 (2014), S. 62 –77.

[189] M. van der Voort, M. S. Dougherty und M. van Maarseveen. „A prototype fuel-efficiency support tool“. In: *Transportation Research Part C: Emerging Technologies* 9.4 (2001), S. 279 –296. url: http://www.sciencedirect.com/science/article/pii/S0968090X00000383.

[190] AUDI AG, Hrsg. *Predictive efficiency assistant*. 2012. url: http://www.audi-technology-portal.de/en/mobility-for-the-future/audi-future-lab-mobility_en/audi-future-engines_en/predictive-efficiency-assistant (besucht am 07. 01. 2017).

[191] M. Muñoz-Organero und V. Magaña. „Validating the Impact on Reducing Fuel Consumption by Using an EcoDriving Assistant Based on Traffic Sign Detection and Optimal Deceleration Patterns“. In: *IEEE Transactions on Intelligent Transportation Systems* 14.2 (Juni 2013), S. 1023–1028.

[192] Y. Zhang u. a. „Remaining driving range estimation of electric vehicle“. In: *2012 IEEE International Electric Vehicle Conference*. März 2012, S. 1–7.

[193] M. Bechler und L. Makeschin. *Reichweitenschätzung für Elektrofahrzeuge*. DE Patent App. DE201,410,204,308. Sep. 2015. url: http://google.com/patents/DE102014204308A1?cl=de.

[194] A. Golding. *Automobile navigation system with dynamic traffic data*. US Patent 5,933,100. Aug. 1999. url: https://www.google.com/patents/US5933100.

[195] J. Fawcett und P. Robinson. „Adaptive Routing for Road Traffic“. In: *IEEE Computer Graphics and Applications* 20.3 (Juni 2000), S. 46–53.

[196] T. Kleine-Besten u. a. „Handbuch Fahrerassistenzsysteme“. In: Wiesbaden: ViewegTeubner Verlag, Springer Fachmedien, 2012. Kap. Navigation und Telematik, S. 599–624.

[197] E. Dagan u. a. „Forward collision warning with a single camera“. In: *IEEE Intelligent Vehicles Symposium, 2004*. Juni 2004, S. 37–42.

[198] M. Stämpfle, W. Branz u. a. „Kollisionsvermeidung im Längsverkehr–die Vision vom unfallfreien Fahren rückt näher“. In: *3. Tagung Aktive Sicherheit durch Fahrerassistenz* (2008).

[199] R. H. Miller und A. L. Tascillo. *Blind spot warning system for an automotive vehicle*. US Patent 6,859,148. Feb. 2005. url: https://www.google.com/patents/US6859148.

[200] M. Walter u. a. „Handbuch Fahrerassistenzsysteme“. In: Wiesbaden: ViewegTeubner Verlag, Springer Fachmedien, 2012. Kap. Lane Departure Warning, S. 543–553.

[201] R. Katzwinkel u. a. „Handbuch Fahrerassistenzsysteme“. In: Wiesbaden: ViewegTeubner Verlag, Springer Fachmedien, 2012. Kap. Bremsenbasierte Assistenzfunktionen, S. 471–477.

[202] K. Schofield, M. Larson und K. Vadas. *Vehicle headlight control using imaging sensor*. US Patent 5,796,094. Aug. 1998. url: https://www.google.com/patents/US5796094.

[203] S. Kuroda u. a. *Engine automatic start stop control apparatus*. US Patent 6,504,259. Jan. 2003. url: https://www.google.com/patents/US6504259.

[204] D. Wolf, G. Hess und J. Twichel. *Automatic start/stop system and method for locomotive engines*. US Patent 6,941,218. Sep. 2005. url: https://www.google.com/patents/US6941218.

[205] R. Bogenrieder, M. Fehring und R. Bachmann. „PRE-SAFE In Rear-End Collision Situations“. In: *Proceedings 21st International Technical Conferrence on the Enhanced Safety of Vehicles*. Stuttgart, 2009.

[206] H. Winner, B. Danner und J. Steinle. „Handbuch Fahrerassistenzsysteme“. In: Wiesbaden: ViewegTeubner Verlag, Springer Fachmedien, 2012. Kap. Adaptive Cruise Control, S. 478–521.

[207] S. Matsumoto u. a. *Lane keep control for vehicle*. US Patent 6,556,909. Apr. 2003. url: https://www.google.com/patents/US6556909.

[208] S. Ishida und J. E. Gayko. „Development, evaluation and introduction of a lane keeping assistance system“. In: *Intelligent Vehicles Symposium, 2004 IEEE*. Juni 2004, S. 943–944. doi: 10.1109/IVS.2004.1336512.

[209] C. Schmitz. *Method and apparatus for driver assistance*. US Patent 8,031,063. Apr. 2011. url: https://www.google.com/patents/US8031063.

[210] C. Lundquist, W. Reinelt und O. Enqvist. „Back Driving Assistant for Passenger Cars with Trailer“. In: *SAE 2006 World Congress & Exhibition*. 2006. doi: 10.4271/2006-01-0940.

[211] R. Baldessari u. a. *CAR 2 CAR Communication Consortium Manifesto*. 1.1. CAR 2 CAR Communication Consortium. Brussels, Aug. 2007.

[212] P. Conradi. „Reichweitenprognose für Elektromobile“. In: *Vernetztes Automobil* (2014), S. 179–186.

[213] S. Strobel, K. Rösinger und M. Bröcker. „Radikale Innovtionen in der Mobilität“. In: Proff, H., 2014. Kap. Fuzzy-Logik basiertes Energiemanagement für Elektrofahrzeuge, S. 211–224.

[214] J. Woestman u. a. *Strategy to use an on-board navigation system for electric and hybrid electric vehicle energy management*. US Patent 6,487,477. Nov. 2002. url: https://www.google.com/patents/US6487477.

[215] E. Granier. *Device and method for emergency call*. US Patent 6,711,399. März 2004. url: https://www.google.com/patents/US6711399.

[216] N. Kaempchen, B. Schiele und K. Dietmayer. „Situation Assessment of an Autonomous Emergency Brake for Arbitrary Vehicle-to-Vehicle Collision Scenarios“. In: *IEEE Transactions on Intelligent Transportation Systems* 10.4 (Dez. 2009), S. 678–687.

[217] E. Coelingh, A. Eidehall und M. Bengtsson. „Collision warning with full auto brake and pedestrian detection-a practical example of automatic emergency braking“. In: *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*. IEEE. 2010, S. 155–160.

[218] H.-G. Wahl. „Optimale Regelung eines prädiktiven Energiemanagements von Hybridfahrzeugen“. Diss. Karlsruher Institut für Technologie, 2015. isbn: 978-3-73150-422-1.

[219] K.-L. Bauer und F. Gauterin. „A Two-Layer Approach for Predictive Optimal Cruise Control“. In: *SAE 2016 World Congress and Exhibition*. 2016. doi: 10.4271/2016-01-0634.

[220] S. Cramer, A. Lange und K. Bengler. „Path Planning and Steering Control Concept for a Cooperative Lane Change Maneuver According to the H-Mode Concept“. In: *7. Tagung Fahrerassistenzsysteme*. Nov. 2015.

[221] J. Nilsson u. a. „Lane Change Maneuvers for Automated Vehicles“. In: *IEEE Transactions on Intelligent Transportation Systems* PP.99 (2016), S. 1–10. issn: 1524-9050. doi: 10.1109/TITS.2016.2597966.

[222] T. Müller. „Chancen und Risiken auf dem Weg zum pilotierten Fahren“. In: *Internationaler Automobil Kongress*. Okt. 2016.

[223] Robert Bosch GmbH, Hrsg. *Traffic jam assist*. url: http://products.bosch-mobility-solutions.com/en/de/driving_comfort/driving_comfort_systems_for_passenger_cars_1/driver_assistance_systems_4/driver_assistance_systems_5.html (besucht am 07. 01. 2017).

[224] AUDI AG, MediaCenter, Hrsg. *Automatisiertes Fahren auf einem neuen Level: der Audi AI Staupilot*. url: https://www.audi-mediacenter.com/de/techday-piloted-driving-der-staupilot-im-neuen-audi-a8-9276/automatisiertes-fahren-auf-einem-neuen-level-der-audi-ai-staupilot-9283 (besucht am 03. 10. 2017).

[225] M. Aeberhard u. a. „Experience, Results and Lessons Learned from Automated Driving on Germany's Highways". In: *IEEE Intelligent Transportation Systems Magazine* 7.1 (2015), S. 42–57. doi: 10.1109/MITS.2014.2360306.

[226] S. Nordbruch u. a. „Automated Valet Parking". In: *7. Tagung Fahrerassistenzsysteme*. Nov. 2015.

[227] M. Buechel u. a. „An Automated Electric Vehicle Prototype Showing New Trends in Automotive Architectures". In: *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. Sep. 2015, S. 1274–1279. doi: 10.1109/ITSC.2015.209.

[228] J. Ziegler u. a. „Making Bertha Drive - An Autonomous Journey on a Historic Route". In: *IEEE Intelligent Transportation Systems Magazine* 6.2 (2014), S. 8–20. issn: 1939-1390. doi: 10.1109/MITS.2014.2306552.

[229] C. Stiller, G. Färber und S. Kammel. „Cooperative Cognitive Automobiles". In: *Proceedings of the 2007 IEEE Intelligent Vehicles Symposium*. Juni 2007, S. 215–220.

[230] E. Bauer u. a. „PRORETA 3: An Integrated Approach to Collision Avoidance and Vehicle Automation". In: *at - Automatisierungstechnik* 60 (Dez. 2012), S. 755–765.

[231] M. Höll und M. Harrer. „Radikale Innovationen in der Mobilität (Track 2)". In: Proff, H., 2014. Kap. Ein Konzept zur integrativen Fahrdynamikregelung auf Basis einer praxisgerechten Fahrzustandsbeobachtung, S. 311–329.

[232] E. Donges. „Aspekte der Aktiven Sicherheit bei der Führung von Personenkraftwagen". In: *Automobil-Industrie* 27 (1982), S. 183–190.

[233] F. O. Flemisch u. a. „Towards cooperative guidance and control of highly automated vehicles: H-Mode and Conduct-by-Wire". In: *Ergonomics* 57.3 (2014). PMID: 24559139, S. 343–360. doi: 10.1080/00140139.2013.869355. url: http://dx.doi.org/10.1080/00140139.2013.869355.

[234] R. Matthaei und M. Maurer. „Autonomous Driving - a top-down-approach". In: *at - Automatisierungstechnik* 63.3 (März 2015), S. 155–167. doi: 10.1515/auto-2014-1136.

[235] B. Kaiser, B. Augustin und C. Baumann. „Von der Komponenten- zur Funktionsorientierten Entwicklung in der FFunktionale Sicherheit". In: *Elektronik im Fahrzeug (ELIV)*. 2013.

[236] H.-G. Krekels und R. Loeffert. „Zentrales Steuergerät für teilautomatisiertes Fahren“. In: *Fahrerassistenzsysteme und Effiziente Antriebe*. Springer, 2015, S. 62–68.

[237] *open robinos specification*. 1.0.1. Elektrobit Automotive GmbH. Juni 2016.

[238] J. Wendel. *Integrierte Navigationssysteme*. Oldenbourg Wissenschaftsverlag GmbH, 2007.

[239] S. Durekovic u. a. *ADASIS v2 Protocol*. 2.0.1.0. ADASIS Forum, c/o ERTICO. Brussels, Feb. 2011.

[240] J. Ludwig. „Elektronischer Horizont - Vorausschauende Systeme und deren Anbindung an Navigationseinheiten“. In: *Vernetztes Automobil* (2014), S. 223–229.

[241] A. Scheel u. a. „Multi-Sensor Multi-Object Tracking of Vehicles using High-Resolution Radars“. In: *2016 IEEE Intelligent Vehicles Symposium (IV)*. 2016.

[242] B. Ranft und C. Stiller. „The Role of Machine Vision for Intelligent Vehicles“. In: *IEEE Transactions on Intelligent Vehicles* 1.1 (März 2016), S. 8–19. issn: 2379-8858. doi: 10.1109/TIV.2016.2551553.

[243] L. Schneider u. a. „Semantic Stixels: Depth is Not Enough“. In: *2016 IEEE Intelligent Vehicles Symposium (IV)*. 2016.

[244] S. Rhode und Gauterin. „Vehicle Mass Estimation Using a Total Least-Squares Approach“. In: *2013 IEEE Intelligent Vehicles Symposium (IV)*. 2013.

[245] I. Szottka. „Particle Filtering for Lane-Level Map-Matching at Road Bifurcations“. In: *IEEE 16th International Conference on Intelligent Transportation Systems (ITSC)*. Okt. 2013, S. 154–159.

[246] M. Haberjahn und K. Kozempel. „Multi level fusion of competitive sensors for automotive environment perception“. In: *Information Fusion (FUSION), 2013 16th International Conference on*. IEEE. 2013, S. 397–403.

[247] J. Batista. „A drowsiness and point of attention monitoring system for driver vigilance“. In: *2007 IEEE Intelligent Transportation Systems Conference*. IEEE. 2007, S. 702–708.

[248] T. Bär u. a. „Probabilistic driving style determination by means of a situation based analysis of the vehicle data“. In: *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2011, S. 1698–1703.

[249] S. Fünfgeld u. a. „Driver state estimation for prediction of vehicle states within control systems“. In: *Intelligent Vehicles Symposium (IV), 2016 IEEE*. IEEE. 2016, S. 1386–1391.

[250] D. Töpfer u. a. „Efficient Scene Understanding for Intelligent Vehicles Using a Part-Based Road Representation“. In: *IEEE 16th International Conference on Intelligent Transportation Systems (ITSC)*. Okt. 2013, S. 65–70.

[251] J. Kramer u. a. „Connected efficiency–A paradigm to evaluate energy efficiency in tactical vehicle-environments“. In: *16. Internationales Stuttgarter Symposium*. Springer. 2016, S. 1451–1463.

[252] J. Nilsson u. a. „Predictive manoeuvre generation for automated driving“. In: *IEEE 16th International Conference on Intelligent Transportation Systems (ITSC)*. Okt. 2013, S. 418–423.

[253] A. Eidehall und D. Madås. „Real time path planning for threat assessment and collision avoidance by steering“. In: *IEEE 16th International Conference on Intelligent Transportation Systems (ITSC)*. Okt. 2013, S. 916–921.

[254] B. Paden u. a. „A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles“. In: *IEEE Transactions on Intelligent Vehicles* 1.1 (März 2016), S. 33–55.

[255] S. Sommer u. a. „Race: A centralized platform computer based architecture for automotive applications“. In: *Electric Vehicle Conference (IEVC), 2013 IEEE International*. IEEE. 2013, S. 1–6.

[256] D. K. Nilsson und U. E. Larson. „Secure firmware updates over the air in intelligent vehicles“. In: *ICC Workshops-2008 IEEE International Conference on Communications Workshops*. IEEE. 2008, S. 380–384.

[257] M. Mitschke und H. Wallentowitz. *Dynamik der Kraftfahrzeuge*. Bd. 5. Wiesbaden: Springer Vieweg, 2014.

[258] S. Moon, K. Yi und I. Moon. „Design, Tuning and Evaluation of Integrated ACC/CA Systems“. In: *17th World Congress of the International Federation of Automatic Control (IFAC 2008)*. Bd. 41. IFAC Proceedings Volumes. Juli 2008, S. 8546–8551.

[259] S. Bauer, R. Streiter und G. Wanielik. „Non-line-of-sight mitigation for reliable urban GNSS vehicle localization using a particle filter“. In: *Information Fusion (Fusion), 2015 18th International Conference on*. IEEE. 2015, S. 1664–1671.

[260] T. Radke. „Energieoptimale Längsführung von Kraftfahrzeugen durch Einsatz vorausschauender Fahrstrategien“. Diss. Karlsruher Institut für Technologie, 2013. isbn: 978-3-7315-0069-8.

[261] P. Schäfer, U. Reuter und H. C. „E-Mobility and driver assistance: Will driving dynamics and driving pleasure stay behind in the future?“ In: *16th Stuttgart International Symposium Automotive and Engine Technology*. Hrsg. von F. für Kraftfahrwesen und Fahrzeugmotoren Stuttgart. Bd. 2. Springer Fachmedien Wiesbaden GmbH, März 2016, S. 567–582.

[262] A. Geiger, P. Lenz und R. Urtasun. „Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite“. In: *Computer Vision and Pattern Recognition (CVPR),(Providence, USA)* (2012).

[263] M. Wiese u. a. „A method to control distributed cause-effect chains of electrical and electronic systems in the field of driving dynamics and driver assistance“. In: *16th Stuttgart International Symposium Automotive and Engine Technology*. Hrsg. von F. für Kraftfahrwesen und Fahrzeugmotoren Stuttgart. Bd. 2. Springer Fachmedien Wiesbaden GmbH, März 2016, S. 615–627.

[264] M. Zofka u. a. „Data-Driven Simulation and Parametrization of Traffic Scenarios for the Development of Advanced Driver Assistance Systems“. In: *IEEE 18th international conference on information fusion*. Juli 2015. doi: 10.1109/ITSC.2013.6728468.

[265] M. Bojarski u. a. *End-to-End Deep Learning for Self-Driving Cars*. 2016. url: https://devblogs.nvidia.com/parallelforall/deep-learning-self-driving-cars/ (besucht am 04. 06. 2017).

[266] A. Blumenstock, P. Glauner und M. Haueis. *Verfahren zur Ermittlung einer Fahrstrecke*. DE Patent App. DE201,410,003,973. Sep. 2014. url: https://www.google.com/patents/DE102014003973A1?cl=de.

[267] J. C. Spall. „Factorial design for efficient experimentation“. In: *IEEE control systems* 30.5 (2010), S. 38–53.

[268] J. Langner u. a. „Framework for using real driving data in automotive feature development and validation“. In: *8. Tagung Fahrerassistenz*. akzeptierter Konferenzbeitrag. München, Nov. 2017.

F Eigene Publikationen

[JB1] J. Bach, S. Otten und E. Sax. „A model-based scenario specification method to support development and test of automated driving functions“. In: *2016 IEEE Intelligent Vehicles Symposium (IV)*. Gotheburg, Juni 2016.

[JB2] J. Bach, S. Otten und E. Sax. „A Taxonomy and Systematic Approach for Automotive System Architectures - From Functional Chains to Functional Networks“. In: *3rd International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS)*. Porto, 2017.

[JB3] J. Bach u. a. „Control based driving assistant functions' test using recorded in field data“. In: *7. Tagung Fahrerassistenzsysteme*. 2015. url: https://mediatum.ub.tum.de/node?id=1285215.

[JB4] J. Bach u. a. „Reactive-Replay approach for verification and validation of closed-loop control systems in early development“. In: *SAE Technical Paper 2017-01-1671*. Detroit, 2017.

[JB5] J. Bach u. a. „Test Scenario Selection for System-Level Verification and Validation of Geolocation-Dependent Automotive Control Systems“. In: *23rd ICE IEEE International Conference on Engineering, Technology and Innovation*. Funchal, 2017.

[JB6] J. Bach u. a. „Data-Driven Development, A Complementing Approach for Automotive Systems Engineering“. In: *2017 IEEE International Symposium on Systems Engineering (ISSE)*. Wien, 2017.

G Betreute studentische Arbeiten

[S1] S. Frohn. „Entwicklung eines Datenloggers auf Basis eines Raspberry Pi“. Bachelorarbeit. Karlsruhe Institut für Technologie (KIT), 2013.

[S2] A. Widmann. „Verifikation von realen Netzwerken und Multibustestwerkzeugen“. Diplomarbeit. Karlsruhe Institut für Technologie (KIT), 2013.

[S3] D. Bachmann. „Konzeption und Evaluation von Map Matching Algorithmen auf Basis der Open Source Projekte Marble und OpenStreetMap“. Masterarbeit. Karlsruhe Institut für Technologie (KIT), 2013.

[S4] N. Goerke. „Design and Evaluation of Algorithms to Improve Position Tracking of Series Production Vehicle“. Bachelorarbeit. Karlsruhe Institut für Technologie (KIT), 2014.

[S5] M. Penzel. „Konzeptionierung des Frontends einer Verkehrsobjekte Simulation“. Bachelorarbeit. Karlsruhe Institut für Technologie (KIT), 2014.

[S6] M. Stang. „Demonstration einer durchgängigen Simulationskopplung für die Fahrerassistenzsystementwicklung mit kombinierten Programmierumgebungen und -sprachen“. Masterarbeit. Karlsruhe Institut für Technologie (KIT), 2015.

[S7] F. Weber. „Entwurf eines erweiterten Kalmanfilters zur INS-GNSS Integration auf Basis von kostengünstiger Sensorik“. Masterarbeit. Karlsruhe Institut für Technologie (KIT), 2015.

[S8] L. Becker. „Conceptual Study Regarding the Protection of Privacy in the Context of Fleet Management Systems and Customer-Oriented Vehicle Testing“. Bachelorarbeit. Karlsruhe Institut für Technologie (KIT), 2016.

[S9] C. King. „Entwicklung und Test eines automatisierten Fahrzeugmodells“. Masterarbeit. Karlsruhe Institut für Technologie (KIT), 2016.

[S10] B. Varga. „Modellprädiktive Regelung eines Demonstratorfahrzeuges“. Masterarbeit. Karlsruhe Institut für Technologie (KIT), 2017.

[S11] H. Schriefers. „Konzeption und Vergleich von Deep-Driving Ansätzen für die Planung und Regelung eines automatisierten Fahrzeugs“. Masterarbeit. Karlsruhe Institut für Technologie (KIT), 2017.

[S12] L. Ries. „Entwurf und Evaluation einer Methode zur Reduktion von Bewertungsaufwänden in der automobilen Funktionsentwicklung mittels Replicator Neural Networks“. Masterarbeit. Karlsruhe Institut für Technologie (KIT), 2017.

[S13] N. Brenner. „Entwicklung einer modularen Testumgebung für Fahrzeugregelungsfunktionen unter Verwendung aufgezeichneter Testdaten und der Reactive-Replay-Methode“. Masterarbeit. Karlsruhe Institut für Technologie (KIT), 2017.