



Universität Stuttgart

# Sicherheitsanalysen von Fail-Operational-Systemen für einen Nachweis nach ISO 26262

Von der Fakultät für Informatik, Elektrotechnik und Informationstechnik der  
Universität Stuttgart zur Erlangung der Würde eines Doktors der  
Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von  
Tobias Schmid, M.Sc.  
aus München

**Hauptberichter:** Prof. Dr. Stefan Wagner

**Mitberichter:** Prof. Dr. Andreas Rausch

**Tag der mündlichen Prüfung:** 20.12.2021

Institut für Software Engineering der Universität Stuttgart

2021



# ZUSAMMENFASSUNG

Der Übergang vom teil- auf das hochautomatisierte Fahren stellt eine Entlastung des Fahrers dar, da dieser das Verkehrsgeschehen nicht mehr permanent überwachen muss. Ein Fail-Silent-Verhalten ist im Fehlerfall kein Übergang in einen sicheren Zustand, weswegen Fail-Operational-Systeme für die funktionale Sicherheit notwendig sind. Fail-Operational-Fahrzeugführungen erfordern redundante Architekturen und neuartige Sicherheitskonzepte, um die Fehlertoleranz und eine geeignete Fehlerreaktion sicherzustellen. Einzelne Aspekte solcher Systeme wurden in der Literatur bereits diskutiert, allerdings fehlt bisher ein hinreichender Nachweis der funktionalen Sicherheit von Fail-Operational-Fahrzeugsystemen.

In dieser Arbeit wird eine hinreichende Argumentation der funktionalen Sicherheit gemäß des Industriestandards ISO 26262 für Fail-Operational-Fahrzeugsysteme vorgestellt. Basierend auf der Argumentation werden notwendige Sicherheitsanalysen inklusive derer Nachweisziele auf Systemebene identifiziert Vorgehen für die jeweiligen Analysen vorgestellt. Daraus ergeben sich zusätzlich die Schnittstellen von System- und Subsystemanalysen. Für die Analyse gemeinsamer Ausfälle und den Nachweis der Unabhängigkeit redundanter Elemente, werden, auf Basis einer Studie zur Identifikation relevanter Anforderungen, existierende Vorgehen adaptiert

und erweitert. Damit ergibt sich ein Vorgehen, das den Randbedingungen der Entwicklung eines Fail-Operational-Systems in der Automobilindustrie gerecht wird. Das Fail-Operational-Verhalten der Umschaltlogik, welche im Fehlerfall eine redundante Fahrzeugführung aktiviert, wird anhand eines Model-Checking-Ansatzes verifiziert. Durch die Qualifizierung des Werkzeugs wird die Konformität zur ISO 26262 sichergestellt. Für die Analyse der Fehlerpropagation und der Fehlertoleranzzeit wird der Ansatz entsprechend um den Softwareverbund erweitert. Implementierungs- und Rechenaufwand zeigen die Anwendbarkeit der Analysen. Darüber hinaus werden Fehlerbaummodelle aus der Luft- und Raumfahrt für den quantitativen Nachweis von Fail-Operational-Systemen adaptiert und mittels Markov-Modellen validiert. Durch eine Sensitivitätsanalyse erfolgt die Identifikation von Optimierungsansätzen zur Minimierung der Ausfallwahrscheinlichkeit.

# ABSTRACT

The transition from semi-automated to highly automated driving relieves the driver, since traffic does not have to be monitored constantly. Fail-Operational systems are necessary for functional safety, since fail-silent behavior does not constitute a transition to a safe state. Fail-Operational vehicle control systems require redundant architectures and novel safety concepts in order to ensure fault tolerance and a suitable fault reaction. Individual aspects of such systems have already been discussed, but so far no sufficient proof of the functional safety of fail-operational vehicle systems has been provided.

In this thesis a sufficient argumentation of functional safety according to the industry standard ISO 26262 for fail-operational vehicle systems is presented. Based on the argumentation, necessary safety analyses including their verification objectives on system level are identified and procedures for the respective analyses are presented. This also results in interfaces of system and subsystem analyses. For the analysis of common failures and the proof of independence of redundant elements, existing procedures are adapted and extended based on a study to identify relevant requirements. This results in a procedure that meets the boundary conditions during the development of a fail-operational system. The fail-operational behaviour of the switching logic, which activates redundant vehicle control in case of a

failure, is verified by means of a model-checking approach. The qualification of the tool ensures conformity with ISO 26262. For the analysis of the failure-propagation and the fault-tolerance time, the approach is extended accordingly by the software network. Implementation and calculation efforts show the applicability of the analyses. In addition, fault tree models from the aerospace industry are adapted for the quantitative detection of fail-operational systems and validated using Markov models. A sensitivity analysis is used to identify optimization approaches to minimize the probability of failure.

# DANKSAGUNGEN

Ohne den Beitrag und die Unterstützung von vielen Seiten, wäre diese Arbeit nicht möglich gewesen.

Ein besonderer Dank gilt meinem Doktorvater Prof. Dr. Stefan Wagner, der die Arbeit und meine Forschung ermöglicht, mich bei meiner Arbeit stets unterstützt und diese durch wertvolle Anregungen und Rückfragen bereichert hat.

Außerdem gilt mein Dank Prof. Dr. Andreas Rausch für die Übernahme des Zweitgutachtens.

Mein Dank gilt ebenso Dr. Stefanie Schraufstetter, die mir während meiner Arbeit, wissenschaftlich und darüber hinaus, zur Seite stand und mich stets motiviert hat. Dr. Christian Wimmer danke ich für sein stets offenes Ohr und die vielen hilfreichen Ratschläge.

Während meiner Arbeit durfte ich die Unterstützung vieler weiterer Kollegen bei BMW erfahren. Sowohl fachliche als auch persönliche Gespräche haben meine Doktorandenzeit bereichert. In diesem Zusammenhang möchte ich auch den von mir betreuten Studenten, insbesondere Gerald Michl, danken. Besonders dankbar bin ich auch Herrn Dominik Hellhake für Rat, Spaß und die Einführung in Großversuche. Auch der Software Engineering Gruppe

und insbesondere Jonas Fritsch möchte ich für die gute Zeit und die Diskussionen über die formale Verifikation danken.

Simon Schmid und Christoph Willibald danke für ihre Anmerkungen zu dieser Arbeit. Ich hoffe, dass ich euch auch entsprechend bei euren Arbeiten unterstützen kann.

Außerdem danke ich meiner Familie, die mir meinen bisherigen Weg über das Studium, die Auslandserfahrungen und die Dissertation ermöglicht und zu jeder Zeit unterstützt haben.

Meiner Freundin Claudia danke ich dafür, dass sie mich über diese Zeit geduldig ertragen und unterstützt hat.

München am 21.12.2021



# INHALTSVERZEICHNIS

1	Einleitung	11
1.1	Fail-Operational-Systeme für das hochautomatisierte Fahren	11
1.2	Problemstellung und Beitrag der Arbeit . . . . .	14
1.2.1	Problemstellung . . . . .	14
1.2.2	Beitrag der Arbeit . . . . .	14
1.3	Veröffentlichungen im Rahmen der Arbeit . . . . .	15
1.4	Aufbau der Arbeit . . . . .	17
2	Grundlagen	21
2.1	Funktionale Sicherheit . . . . .	21
2.1.1	Der Industriestandard ISO 26262 . . . . .	22
2.1.2	Begriffsdefinitionen für Sicherheitsbetrachtungen . . . . .	24
2.1.3	Der Sicherheitslebenszyklus nach ISO 26262 . . . . .	29
2.2	Sicherheitsargumentation und Safety Case . . . . .	32
2.3	Fail-Operational-Fahrzeugsysteme . . . . .	33
2.4	Sicherheitsanalysen . . . . .	39
2.4.1	Klassische Analysen . . . . .	40
2.4.2	Analyse der Fehlerpropagation . . . . .	45
2.4.3	Formale Methoden . . . . .	47

3	Stand der Wissenschaft	53
3.1	Relevante Sicherheitsargumentationen . . . . .	54
3.1.1	Sicherheitsargumentationen für Fail-Operational-Systeme	54
3.1.2	Sicherheitsargumentationen mittels Goal Structuring Notation gemäß ISO 26262 . . . . .	58
3.2	Analyse gemeinsamer Ausfälle . . . . .	61
3.3	Formale Verifikation mittels Model-Checking . . . . .	67
3.3.1	Model-Checking in der Automobilindustrie . . . . .	67
3.3.2	Optimierungsansätze des Model-Checkings . . . . .	72
3.4	Analyse der Fehlerpropagation . . . . .	75
3.5	Quantitative Analyse von mehrkanaligen Systemen . . . . .	78
3.6	Zusammenfassung des Forschungsstands . . . . .	82
4	Sicherheitsargumentation gemäss ISO 26262	85
4.1	Sicherheitsrelevante Anforderungen nach ISO 26262 . . . . .	87
4.2	Modellierung des Fail-Operational-Systemverhaltens . . . . .	91
4.3	Ableitung von Fehlermodellen . . . . .	94
4.4	Sicherheitsargumentation mittels Goal Structuring Notation	99
5	Analyse gemeinsamer Ausfälle bei Fail-Operational-Systemen	109
5.1	Studie zur Identifikation von Anforderungen an die Analyse gemeinsamer Ausfälle . . . . .	110
5.1.1	Ziel der Studie . . . . .	110
5.1.2	Aufbau, Durchführung und Auswertung der Studie . . . . .	111
5.1.3	Ergebnisse der Studie . . . . .	113
5.1.4	Zusammenfassung und Diskussion der Ergebnisse . . . . .	118
5.2	Erweiterung der Vorgehensweise und Ableitung eines Datenmodells . . . . .	120
5.3	Validierung des Vorgehens anhand einer Fallstudie . . . . .	125
5.4	Quantifizierung von gemeinsamen Ausfällen . . . . .	131
6	Formale Verifikation der Umschaltlogik	137
6.1	Modellierung des Systems und formale Spezifikation . . . . .	139

6.1.1	Modellierung der Umschaltlogik . . . . .	139
6.1.2	Spezifikation der Sicherheitsanforderungen . . . . .	143
6.2	Identifikation relevanter Fehlerfälle und Implementierung .	148
6.3	Validierung und Werkzeugqualifizierung . . . . .	152
6.3.1	Validierung des Modells . . . . .	152
6.3.2	Werkzeugqualifizierung nach ISO 26262 . . . . .	154
6.4	Ergebnisse und Anwendbarkeit des Model-Checkings . . . .	157
7	Formaler Nachweis des Verhaltens im Systemverbund	165
7.1	Modellierung des Systemverbunds . . . . .	167
7.1.1	Software-Komponenten als modulare Fehlerbäume . . . .	168
7.1.2	Modellierung der Architektur und Kommunikation . . . .	171
7.2	Analyse der Umschaltung im Verbund . . . . .	174
7.3	Validierung und Qualifizierung . . . . .	180
7.4	Ergebnisse und Diskussion des Ansatzes . . . . .	183
7.4.1	Anwendbarkeit der Methode . . . . .	183
7.4.2	Optimierung der Systemrobustheit auf Basis der Analyse	186
7.4.3	Diskussion des Ansatzes . . . . .	188
8	Quantitative Analyse von Fail-Operational-Systemen	191
8.1	Modellierung mehrkanaliger Systeme mittels Fehlerbaum .	193
8.2	Validierung der Analyse mit Markov-Modellen . . . . .	200
8.3	Systemoptimierung auf Basis einer Sensitivitätsanalyse . . .	209
9	Schluss	215
9.1	Zusammenfassung und Beiträge . . . . .	215
9.2	Diskussion und Ausblick . . . . .	218
	Literaturverzeichnis	221
	Abbildungsverzeichnis	243
	Tabellenverzeichnis	247
	Abkürzungsverzeichnis	249



# KAPITEL 1

## EINLEITUNG

### 1.1 Fail-Operational-Systeme für das hochautomatisierte Fahren

Der technologische Fortschritt im Bereich der Sensorik und Algorithmik und zugleich immer leistungsstärkere Rechenplattformen ermöglichen die Entwicklung von Fahrerassistenzfunktionen hin zum automatisierten und autonomen Fahren [Koc18, WHLS15]. Diese Entwicklung stellt neben der Elektrifizierung, Vernetzung und der Bereitstellung von Dienstleistungen ein Zukunftsfeld der Automobilindustrie dar [Dor18]. Dem Kunden bieten automatisierte Fahrfunktionen einen Gewinn an Komfort, Effizienz und Sicherheit durch die Entlastung des Fahrers (vgl. [Koc18, WRM+19]), weswegen im Jahr 2040 mit 10 bis 20% automatisierten Fahrzeugen im Verkehr gerechnet wird [AKM18].

Automobilhersteller arbeiten an der zunehmenden Automatisierung von Fahrfunktionen [Dor18, Ver15a]. Die Entwicklung bis hin zum autonomen Fahren erfolgt dabei evolutionär. Heutzutage können Fahrzeugsysteme die Fahraufgabe in einzelnen Verkehrssituationen übernehmen; eine Über-

wachung und Unterstützung des Fahrers ist jedoch notwendig [Koc18]. Bezüglich der Automatisierung werden verschiedene Stufen unterschieden, die sich im Anteil der Fahraufgabe, welche Fahrer und System übernehmen, unterscheiden. Abbildung 1.1 zeigt die Stufen des automatisierten Fahrens inklusive der Fahraufgabe von Fahrer und System nach dem Verband der deutschen Automobilindustrie (VDA) [Ver15a].



	Stufe 0 nur Fahrer	Stufe 1 Assistiert	Stufe 2 Teil- Automatisiert	Stufe 3 Hoch- Automatisiert	Stufe 4 Voll- Automatisiert	Stufe 5 Fahrerlos
	Quer- und Längsführung	Quer- oder Längsführung	Dauerhafte Überwachung	Keine dauerhafte Überwachung	-	-
	-	Jeweils andere Fahraufgabe	Quer- und Längsführung im Anwendungsfall	Quer- und Längsführung im Anwendungsfall, Erkennt Systemgrenzen	Beherrscht im Anwendungsfall alle Situationen	Übernimmt Fahraufgabe vollumfänglich

Abbildung 1.1: Definition der Stufen des automatisierten Fahrens nach Aufgabe des Fahrers und des Systems gemäß VDA [Ver15a]

Aktuelle Systeme ermöglichen den teilautomatisierten Betrieb durch Funktionen wie einen Stauassistenten [Ver15a], bei welchem das System, begrenzt auf spezifische Anwendungsfälle, die Quer- und Längsführung übernehmen kann. Anwendungsfälle werden dabei durch Umweltbedingungen, Straßen und die Geschwindigkeit definiert. Der Fahrer muss das System zu jedem Zeitpunkt überwachen und gegebenenfalls die Fahraufgabe übernehmen. Der Übergang zum hochautomatisierten Fahren, der Stufe 3 des automatisierten Fahrens, erlaubt dem Fahrer sich zeitweise abzuwenden. Das Fahrzeug kann für längere Strecken, beispielsweise auf der Autobahn, selbständig fahren [Har20]. Bei erkannten Systemgrenzen oder im Fehlerfall wird der Fahrer zur Übernahme aufgefordert und muss nach einem angemessenen Zeitraum die Fahraufgabe übernehmen. [Ver15a]

In teilautomatisierten Systemen stellt ein Abschalten von assistierenden Systemen eine sichere Reaktion dar, da der Fahrer über mechanische Rückfallebenen die Fahrzeugführung unmittelbar übernehmen kann. Das Verhalten im Fehlerfall wird daher als Fail-Silent bezeichnet. Beim hochautomatisierten Fahren dagegen erfolgt durch das Abschalten kein Übergang in einen sicheren Zustand, da nicht von einer unmittelbaren Fahrerübernahme ausgegangen werden kann. Das erfordert ein fehlertolerantes, sogenanntes Fail-Operational, Verhalten, bei welchem die Fahrzeugführung auch im Fehlerfall fortgeführt wird. [Sch16][Ise16] Fail-Operational-Systeme basieren auf redundanten Systemen für die Fahrzeugführung [Sch16][KST+19]. Durch den Übergang von Fail-Silent- zu Fail-Operational-Systemen ergeben sich bezüglich der funktionalen Sicherheit neue Herausforderungen. Das Erreichen des sicheren Zustands, im Falle von Fail-Operational-Anforderungen die Fortführung der Fahrzeugführung, ist komplexer und erfordert bezüglich der funktionalen Sicherheit und deren Nachweis gemäß dem Industriestandard ISO 26262 eine Anpassung und Erweiterung bisheriger Sicherheitsaktivitäten.

## 1.2 Problemstellung und Beitrag der Arbeit

### 1.2.1 Problemstellung

Für die Freigabe von Fahrzeugsystemen ist die funktionale Sicherheit des Systems und deren Nachweis unabdingbar [WHL15]. Fail-Operational-Fahrzeugsysteme für das hochautomatisierte Fahren stellen eine neue Technologie dar und unterscheiden sich durch redundante Architekturen und das Sicherheitskonzept, die Reaktion auf einen Fehlerfall, von herkömmlichen Fail-Silent Systemen. Auch wenn Fail-Operational-Systeme beispielsweise in der Luft- und Raumfahrt bereits verfügbar sind, ist die Übertragbarkeit durch diversitäre Randbedingungen nur eingeschränkt gegeben. Untersuchungen zu einzelnen Aspekten der funktionalen Sicherheit von Fail-Operational-Systemen für das hochautomatisierte Fahren sind vorhanden. Eine hinreichende Sicherheitsargumentation unter Berücksichtigung der Norm für funktionale Sicherheit in der Automobilindustrie, der ISO 26262 [Int18], wurde für Fail-Operational-Systeme bisher nicht erbracht. Diese ist jedoch für die Einführung hochautomatisierter Fahrfunktionen unbedingt notwendig. Die Identifikation notwendiger Sicherheitsanalysen muss im Rahmen der Argumentation erfolgen.

### 1.2.2 Beitrag der Arbeit

Die vorliegende Arbeit schließt diese Lücke durch die Herleitung und Beschreibung einer hinreichenden Sicherheitsargumentation und liefert damit einen Beitrag zur funktionalen Sicherheit von Fail-Operational-Fahrzeugführungen für das hochautomatisierte Fahren. Für die aus der Sicherheitsargumentation identifizierten notwendigen Nachweise werden Methoden für entsprechende Analysen auf Systemebene präsentiert, die den Vorgaben der Norm für funktionale Sicherheit in der Automobilindustrie ISO 26262 entsprechen. Darüber hinaus werden Analysemethoden für die relevanten Nachweisziele auf Systemebene vorgestellt.



Im Detail liefert die Arbeit folgende Beiträge:

- Hinreichende Sicherheitsargumentation für eine Fail-Operational-Fahrzeugführung gemäß ISO 26262 unter Verwendung der Goal Structuring Notation,
- Vorgehen und Datenstruktur für die Analyse gemeinsamer Ausfälle redundanter Elemente für Fail-Operational-Systeme auf Basis einer Studie,
- Methodik und Anwendung von formaler Verifikation mittels Model-Checking zum Nachweis des Fail-Operational-Verhaltens von Umschaltlogiken,
- Methodik und Anwendung von Model-Checking für den Nachweis des Fail-Operational-Verhaltens und der Einhaltung der Fehlertoleranzzeit im Systemverbund unter Berücksichtigung von Fehlerpropagation und zeitlichen Aspekten,
- Methodik für eine quantitative Analyse von Fail-Operational-Systemen mit dynamischen Redundanzen und verschiedenen Betriebsphasen und eine Systemoptimierung bezüglich der Ausfallwahrscheinlichkeit.

### 1.3 Veröffentlichungen im Rahmen der Arbeit

Folgende Publikationen, in chronologischer Reihenfolge aufgelistet, wurden im Rahmen der Dissertationsarbeit erstellt und leisten einen Beitrag zu der vorliegenden Arbeit:

- T. Schmid. „Safety Analysis for Highly Automated Driving“. In: 29th IEEE International Symposium on Software Reliability Engineering workshops. Hrsg. von S. Ghosh. Piscataway, NJ: IEEE, 2018, S. 154–157. doi: 10.1109/ISSREW.2018.000-7.

- T. Schmid, S. Schraufstetter, S. Wagner. „An Approach for Structuring a Highly Automated Driving Multiple Channel Vehicle System for Safety Analysis“. In: 2018 3rd International Conference on System Reliability and Safety. Piscataway, NJ: IEEE, 2018, S.362–367. doi: 10.1109/ICSRs.2018.8688859.
- T. Schmid, S. Schraufstetter, S. Wagner, D. Hellhake. „A Safety Argumentation for Fail-Operational Automotive Systems in Compliance with ISO 26262“. In: 2019 4th International Conference on System Reliability and Safety. Piscataway, NJ: IEEE, 2019, S. 484–493. doi: 10.1109/ICSRs48664.2019.8987656.
- T. Schmid, S. Schraufstetter, T. Hagler, D. Krepis. „Quantitative Analyse von Fail-Operational Hardwarearchitekturen“. In: Safe.Tech - Funktionale Sicherheit in der Bahntechnik, Automatisierung und Automobiltechnik. 2020.
- J. Fritsch, T. Schmid, S. Wagner. „Experiences from Large-Scale Model Checking: Verification of a Vehicle Control System.“ In: IEEE International Conference on Software Testing, Verification and Validation 2021. 2021. url: <https://arxiv.org/abs/2011.10351>.
- T. Schmid, S. Schraufstetter, J. Fritsch, D. Hellhake, G. Koelln, S. Wagner. „Formal Verification of a Fail-Operational Automotive Driving System“. In: IEEE Transactions on Dependable and Secure Computing. 2020-11-23 (eingereicht). url: <https://arxiv.org/abs/2101.07307>.

## 1.4 Aufbau der Arbeit

Im Folgenden wird der Aufbau der Arbeit, gezeigt in Abbildung 1.2, vorgestellt.

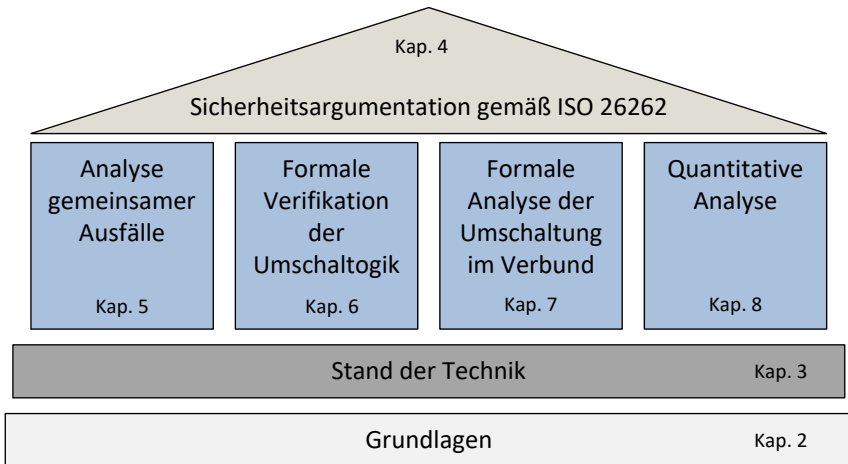


Abbildung 1.2: Aufbau der Arbeit

In **Kapitel 2** werden die für das Verständnis der Arbeit relevanten Grundlagen vermittelt. Das beinhaltet die Grundbegriffe der funktionalen Sicherheit und die ISO 26262 als Industrienorm für die funktionale Sicherheit in der Automobilindustrie. Darauf aufbauend werden der Sicherheitslebenszyklus, inklusive der wichtigsten Arbeitsprodukte, und Sicherheitsargumentationen vorgestellt. Der Grundlagenteil vermittelt auch einen Überblick über Fail-Operational-Fahrzeugsysteme und deren Sicherheitsaspekte auf Basis relevanter Sicherheitsziele. Grundlagen bezüglich Sicherheitsanalysen inklusive formaler Methoden werden am Ende des Kapitels adressiert.

Der Stand der Wissenschaft wird in **Kapitel 3** vorgestellt und Forschungslücken werden identifiziert. Entsprechend des Aufbaus der Arbeit erfolgt die Vorstellung von Arbeiten zu Sicherheitsargumentationen für Fail-Operational-Systeme und im Kontext der ISO 26262, der Analyse gemeinsamer Ausfälle,

formaler Verifikation als Sicherheitsnachweis und zur quantitativen Analyse. Im Rahmen der Diskussion des aktuellen Forschungsstands der formalen Verifikation, werden die Anwendung im Automobilbereich und relevante Optimierungen der Methodik adressiert.

In **Kapitel 4** wird eine Sicherheitsargumentation für Fail-Operational-Fahrzeugsysteme präsentiert. Aus den Umfängen von Sicherheitsanforderungen gemäß der Industrienorm ergeben sich relevante Sicherheitsaspekte. Zusammen mit Fehlermodellen des Fail-Operational-Systems erfolgt die Konsolidierung zu einer Sicherheitsargumentation unter Verwendung der Goal Structuring Notation als graphische Notationsform. Entsprechend der ISO 26262 werden sowohl die Identifikation der Fehler als auch die Verifikation der Sicherheitsanforderungen abgedeckt, wodurch sich eine hinreichende Argumentation der funktionalen Sicherheit ergibt. Aus der Argumentation lassen sich die notwendigen Analysen auf System- und Subsystemebene ableiten.

Ansätze der identifizierten Analysen auf Systemebene werden in den **Kapiteln 5 mit 8** vorgestellt. Diese adressieren die Unabhängigkeit gemeinsamer Ausfälle, die Verifikation der Umschaltlogik und die Analyse der Fehlerpropagation sowie den quantitativen Nachweis einer ausreichend geringen Wahrscheinlichkeit von Sicherheitszielverletzungen.

**Kapitel 5** umfasst die Analyse gemeinsamer Ausfälle. Um die Grenzen bisheriger Vorgehensweisen und relevante Anforderungen zu identifizieren, wurde eine Studie durchgeführt. Basierend auf dieser Studie und der Literatur erfolgt die Erweiterung der allgemeinen Vorgehensweise und dessen Validierung anhand einer Fallstudie. Zusätzlich wird die Anwendung quantitativer Methoden zur Berücksichtigung von abhängigen Fehlern in der Automobilentwicklung diskutiert.

Die Verifikation der Umschaltlogik anhand eines Model-Checking Ansatzes wird in **Kapitel 6** präsentiert. Hierfür erfolgt die Modellierung der Um-

schaltlogik und der relevanten Architektur sowie die formale Spezifikation der Sicherheitsanforderungen. Die Umsetzung des Model-Checkings unter Berücksichtigung der ISO 26262, die Validierung und die Qualifizierung des Werkzeugs wird ebenfalls beschrieben. Die Ergebnisse inklusive einer Diskussion der Anwendbarkeit stellen den Schlussteil des Kapitels dar.

Für die Analyse der Fehlerpropagation unter Berücksichtigung von variablen Latenzzeiten wird der Model-Checking Ansatz in **Kapitel 7** um den Softwareverbund und relevante Signale erweitert. Zur Verifikation wird ein Tiefensuchalgorithmus vorgestellt. Ein Minimalbeispiel dient der Validierung. Die Anwendung wird anhand einer repräsentativen Fahrwerksfunktion untersucht. Mittels der Analyse werden Systemeigenschaften für eine höhere Robustheit des Verhaltens identifiziert.

Die quantitative Analyse, der Nachweis eines ausreichend geringen Risikos durch zufällige Hardwareausfälle, erfolgt in **Kapitel 8**. Dazu wird die Modellierung eines Fehlerbaums unter Berücksichtigung verschiedener Betriebsphasen vorgestellt. Die Validierung des Fehlerbaumes inklusive der Abstraktionen erfolgt mittels einer Markov-Analyse. Darüber hinaus werden Systemparameter zur Optimierung der Ausfallwahrscheinlichkeit durch eine Sensitivitätsanalyse identifiziert.

Die Arbeit schließt mit der Zusammenfassung, der Diskussion der Ergebnisse und einem Ausblick auf weiteren Forschungsbedarf und mögliche nächste Schritte in **Kapitel 9**.



# KAPITEL 2

## GRUNDLAGEN

Im folgenden Kapitel werden relevante Grundlagen für das Verständnis der weiteren Arbeit vermittelt. Hierfür wird die funktionale Sicherheit inklusive relevanter Begrifflichkeiten und dem Standard der Automobilindustrie, ISO 26262, eingeführt. Fail-Operational-Systeme im Fahrzeug und deren sicherheitsrelevante Aspekte werden vorgestellt. Außerdem werden die relevanten Grundlagen von Sicherheitsargumentationen und Sicherheitsanalysen präsentiert.

### 2.1 Funktionale Sicherheit

Die Sicherheit stellt eine grundlegende Anforderung an Fahrzeugsysteme dar. Im Allgemeinen wird unter dem Begriff die Freiheit von unakzeptablen Risiken verstanden [SH19]. Sicherheitskritische elektronische Systeme können sowohl einen Beitrag zur aktiven als auch zur passiven Sicherheit der Fahrzeuge leisten [SH19]. Unter aktiver Sicherheit wird die Vermeidung von Unfällen, unter der passiven Sicherheit die Minderung der Unfallschwere verstanden [PS16]. Auch für elektronische Systeme muss das Risiko in einem akzeptablen Bereich liegen, was durch die Zunahme an Automatisierung

und des Funktionsumfangs immer bedeutender wird [WHLS15]. Der Bereich des akzeptablen Risikos wird durch den Stand der Technik definiert [WHLS15]. Dabei müssen nach Schnieder et al. [SH19] für elektronische Systeme die Sicherheit der Sollfunktion (engl. Safety of intended Function), die Angriffssicherheit (engl. Cyber-Security) und die funktionale Sicherheit (engl. functional Safety) berücksichtigt werden.

Im Gegensatz zu der Sicherheit der Sollfunktion, der Sicherheit im fehlerfreien Zustand, beschäftigt sich die funktionale Sicherheit mit der Sicherheitsbetrachtung von Fehlern elektrischer und elektronischer Systeme (Elektrik und Elektronik (E/E)) [SH19]. Ziel ist es, mögliche Fehler zu identifizieren, deren Auswirkungen zu bewerten und durch geeignete Maßnahmen das Risiko auf ein akzeptables Maß zu beschränken [SH19]. Der Industriestandard für funktionale Sicherheit von Personenkraftwagen, ISO 26262 [Int18], definiert funktionale Sicherheit als die Freiheit von unakzeptablen Risiken, ausgelöst durch Fehlfunktionen von E/E-Systemen, welche zu Gefährdungen führen. Im Schadensfall ist der Hersteller verpflichtet Nachweise vorzulegen und damit zu beweisen, dass die Systeme bezüglich der funktionalen Sicherheit entsprechend dem Stand der Technik entwickelt wurden [WHLS15].

### 2.1.1 Der Industriestandard ISO 26262

Die Norm für funktionale Sicherheit im Automobilbereich (engl. road vehicles - functional Safety) ist die ISO 26262 [Int18]. Die erste Version wurde 2011 veröffentlicht und umfasst die Betrachtung von Personenkraftwagen [Int18]. In der zweiten Version, welche 2019 veröffentlicht wurde, werden auch Lastkraftwagen, Busse und Motorräder berücksichtigt. Die ISO 26262 basiert dabei auf der Norm für funktionale Sicherheit sicherheitsbezogener elektrischer/elektronischer/programmierbarer elektronischer Systeme, der IEC 61508 [Deu01] und stellt den bereits angesprochenen Stand der Technik bei der Berücksichtigung funktionaler Sicherheit dar. Fokus der ISO 26262 sind dabei elektrische und elektronische Systeme in Straßenfahrzeugen. Entsprechend der Definition werden dabei Fehler von elektronischen und elektrischen Komponenten, einschließlich deren Zusammenwirken, betrach-



tet. Rein mechanische Bauteile stehen, ebenso wie Oxidation, Feuer, Rauch und andere äußere Einflüsse, sofern sie nicht durch Fehler elektronischer oder elektrischer Komponenten verursacht werden, nicht im Fokus. In der Norm werden Richtlinien und Anforderungen für die Entwicklung von sicherheitskritischen E/E Systemen über den kompletten Sicherheitslebenszyklus vorgegeben. Die Norm umfasst zwölf Teile, den Aufbau zeigt Abbildung 2.1.

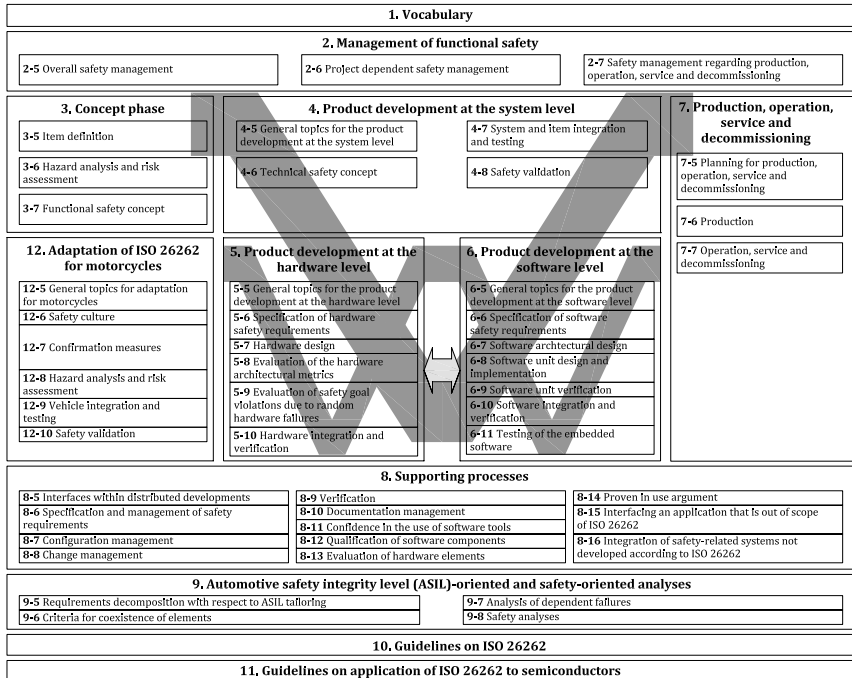


Abbildung 2.1: Aufbau und Umfang des Industriestandards ISO 26262 [Int18]

Die einzelnen Teile werden, inklusive derer spezifischen Kapitel in der Abbildung dargestellt. Der Aufbau orientiert sich dabei an einem Entwicklungsvorgehen entsprechend des V-Modells. Im Folgenden wird eine kurze Übersicht der einzelnen Teile gegeben. Arbeitsprodukte im Rahmen des Sicherheitslebenszyklus werden in Kapitel 2.1.3 vorgestellt.

In Teil 1 werden grundlegende Begriffe der ISO 26262 definiert. Teil 2 beschreibt Anforderungen an die Organisation und das Management zur Unterstützung einer sicherheitsgerichteten Entwicklung. Das sicherheitsgerichtete Management umfasst sowohl allgemeine Umfänge, wie das Etablieren einer Sicherheitskultur, als auch projektspezifische Umfänge, wie die Definition und Planung der Sicherheitsaktivitäten. Die Konzeptphase in Teil 3 behandelt die Definition und Abgrenzung der Betrachtungseinheit, die Gefahren und Risikobewertung mit der Definition der Sicherheitsziele sowie die Ableitung des funktionalen Sicherheitskonzeptes. Die Entwicklungsphase auf Systemebene wird in Teil 4 beschrieben. Im Rahmen der Produktentwicklung werden, abhängig von der Sicherheitseinstufung, Vorgaben und Empfehlungen gegeben. Die Norm legt die Ableitung des technischen Sicherheitskonzeptes sowie dessen Integrations- und Testumfänge dar. Auch die Validierung, der Nachweis der Wirksamkeit des Sicherheitskonzeptes, wird behandelt. In Teil 5 und 6 finden sich Entwicklungs- und Verifikationsvorgaben auf Hardware- und Softwareebene. Produktphasen nach der Entwicklung, wie Produktion, Instandhaltung und Wiederverwertung, werden in Teil 7 berücksichtigt. Teil 8 stellt übergreifende Aspekte dar. Dazu gehören die Beschreibung der sicherheitsrelevanten Anforderungen, Dokumentation, Änderungsprozesse, Werkzeugqualifikation oder Argumentationen basierend auf der Betriebserfahrung von Systemen. Teil 9 beschreibt zusätzlich Aspekte der Sicherheitskonzepte, wie die Dekomposition und Sicherheitsanalysen inklusive der Analyse abhängiger Fehler. Der Teil 10 gibt eine Übersicht und zusätzliche Erklärungen zum Sicherheitslebenszyklus. Anwendungsspezifische Vorgaben für Halbleiterelemente sowie Motorräder finden sich in den Teilen 11 und 12.

### 2.1.2 Begriffsdefinitionen für Sicherheitsbetrachtungen

Im Folgenden werden für die Arbeit relevante Begriffe der Sicherheitsbetrachtung eingeführt. Die Definitionen gemäß ISO 26262 erfordern eine Übersetzung aus dem Englischen. Arbeitsprodukte im Rahmen des Sicherheitslebenszyklus werden nicht beschrieben (vgl. Kapitel 2.1.3).

## Fehler

Der Fehlerbegriff ist in der deutschen Sprache nicht eindeutig und wird für verschiedene Aspekte verwendet. Es wird zwischen der Fehlerursache (engl. fault), dem Fehlerzustand (engl. error) und der Fehlerauswirkung oder dem Ausfall (engl. failure) unterschieden [Hil12][Int18]. Abbildung 2.2 zeigt dabei die Wirkkette ausgehend von der Fehlerursache zum Versagen.

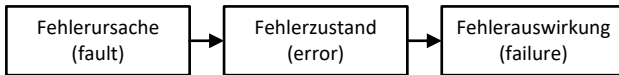


Abbildung 2.2: Fehlerbegriffe und Fehlerwirkkette entsprechend ISO 26262 [Int18]

Eine Fehlerursache ist gemäß ISO 26262 ein vom gewünschten Verhalten abweichendes Verhalten, welches zu einem Versagen führen kann. Eine Fehlerursache führt zu einem Fehlerzustand, einer Abweichung von tatsächlichen Werten gegenüber spezifizierten oder korrekten Werten. Dieser resultiert in einer Auswirkung oder einem Versagen, einem Zustand, in dem das System die Funktion nicht mehr erfüllt. [Int18] Eine Fehlerauswirkung auf Komponentenebene kann dabei eine Fehlerursache auf Systemebene darstellen [Sch16]. Sicherheitsmechanismen ermöglichen die Unterbrechung der Fehlerwirkkette. Eine Diagnose und ein Sicherheitsmechanismus, beispielsweise die Verwendung von Ersatzwerten, verhindern eine Abweichung außerhalb einer Toleranz und damit die Fehlerauswirkung.

Zusätzlich werden in der ISO 26262 verschiedene Fehlerarten unterschieden [Int18]. Diese werden in Abbildung 2.3 dargestellt.

Es wird zwischen **sicheren Fehlern**, **Einfachfehlern** und **Mehrfachfehlern** differenziert. **Sichere Fehler** tragen nicht zu einer Verletzung von Sicherheitszielen bei. **Einfachfehler** führen zu einer potentiellen Sicherheitszielverletzung. Für solche Fehler werden im Rahmen der Sicherheitsanalysen Maßnahmen abgeleitet. Restfehler (engl. residual faults) sind Einfachfehler bei denen die Maßnahmen die Verletzung von Sicherheitszielen nicht vollständig verhindern [Int18]. **Mehrfachfehler** führen nur in Kombination zu Verletzungen von Sicherheitszielen. Solange diese vorliegen, führt das

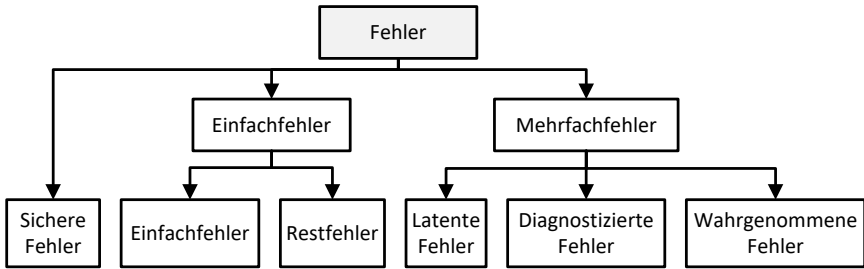


Abbildung 2.3: Fehlerarten nach ISO 26262 [Int18]

Auftreten weiterer Fehler zu einer Verletzung von Sicherheitszielen. Ein vorliegender, nicht erkannter Erstfehler stellt einen latenten Fehler dar. Wenn Fehler durch Diagnosemechanismen oder den Fahrer erkannt werden, handelt es sich um diagnostizierte beziehungsweise um wahrgenommene Fehler. [Int18]

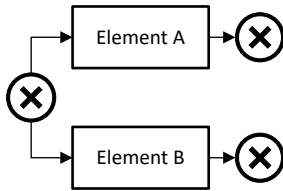
### Abhängige Fehler

Fehler, deren auftreten nicht unabhängig voneinander ist, werden abhängige Fehler (engl. dependent failures) genannt. Für unabhängige Ereignisse ist die Wahrscheinlichkeit des Ausfalls mehrerer Elemente, zum Beispiel Komponenten, gleich des Produktes der Wahrscheinlichkeiten der Einzelereignisse ( $P(E_A), P(E_B)$ ). Sofern diese Unabhängigkeit nicht gegeben ist, steigt die resultierende Ausfallwahrscheinlichkeit durch die Wahrscheinlichkeit, die einem gemeinsamen Ausfall (engl. common-cause failure) entspricht [Deu01, Int18].

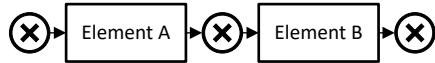
$$P(E_A \wedge E_B) > P(E_A) \cdot P(E_B).$$

Der Ausfall erfolgt dabei nicht notwendigerweise zum gleichen Zeitpunkt oder in geringem zeitlichem Abstand. In der Norm ISO 26262 werden zwei Arten von abhängigen Fehlern unterschieden: gemeinsame Ausfälle (engl. common-cause failures) und kaskadierende Fehler (engl. cascading failures). Abbildung 2.4 zeigt die Wirkketten beider Fehlerarten.

Bei gemeinsamen Ausfällen führt eine Ursache direkt zu Ausfällen mehrerer Elemente. Kaskadierende Fehler treten auf, wenn eine Fehlerursache in der



(a) gemeinsamer Ausfall



(b) Ausfall durch kaskadierenden Fehler

Abbildung 2.4: Zusammenhang der Fehlerursache und Fehlerauswirkung bei abhängigen Fehlern [Int18]

Folge zu einem Ausfall weiterer Elemente führt [Int18]. Dabei wird von Fehlerpropagation oder Fehlerfortpflanzung gesprochen. Schnellbach [Sch16] merkt an, dass die Definitionen in [Int11] nicht eindeutig sind, insbesondere wenn sich ein Fehler über mehrere Komponenten hinweg propagiert, und dann zum Ausfall mehrerer Elemente durch die gleiche Ursache führt. Daher definiert Schnellbach gemeinsame Ausfälle als parallele, direkte Ausfälle einer Ursache und kaskadierende Ausfälle als sequentielle Ausfälle, was in der zweiten Version der ISO 26262 [Int18] berücksichtigt wurde. [Sch16]

### Integrität

Die Sicherheitsintegrität (kurz Integrität) bezieht sich auf eine Sicherheitsfunktion eines Elements. Diese kann als eine Wahrscheinlichkeit der Erfüllung einer bestimmten Risikoreduktion interpretiert werden [Ros16]. In der ISO 26262 erfolgt die Einstufung in verschiedene Kategorien mittels des Automotive Safety Integrity Level (ASIL). Das Risiko bestimmt sich durch die Auftretenswahrscheinlichkeit, die Schwere eines Schadens oder einer Verletzung, sowie die Kontrollierbarkeit [Int18].

### sicherer Zustand

Der sichere Zustand ist nach ISO 26262 [Int18] ein Zustand ohne unakzeptable Risiken. In einem sicheren Zustand liegt entweder kein Risiko vor oder die Sicherheitsintegrität entspricht diesem. Dabei gibt es verschiedene Verhalten bezüglich der Fehlerreaktionen. Von Fail-Silent wird gesprochen,

wenn der Übergang in den sicheren Zustand durch Passivieren erfolgt. Dabei ist eine Beeinflussung weiterer Komponenten zu verhindern (engl. freedom-from-interference [Int18]). Wird der sichere Zustand durch Abschalten nicht erreicht, muss der Fehler toleriert und die relevanten Funktionen aufrechterhalten werden. Ein solches Verhalten wird als Fail-Operational bezeichnet. [Ise16]

### Fehlertoleranzzeit

Die Fehlertoleranzzeit (engl. fault-tolerant-time) umfasst das Zeitintervall vom Auftreten eines Fehlers bis dieser zu einer Sicherheitszielverletzung führt und ist ein Attribut eines Sicherheitsziels. Um eine Gefährdung zu verhindern, müssen die Sicherheitsmechanismen daher innerhalb dieser Zeitspanne wirksam werden. Abbildung 2.5 zeigt die zeitlichen Zusammenhänge.

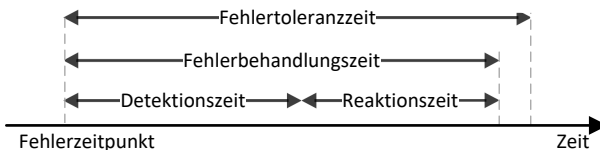


Abbildung 2.5: Fehlertoleranzzeit und Fehlerbehandlungszeit nach ISO 26262 [Int18]

Die Fehlerbehandlungszeit, die sich aus der Detektionszeit und der Reaktionszeit zusammensetzt, muss kleiner der Toleranzzeit sein. Diese endet mit dem Erreichen des sicheren Zustands [Int18].

### Ausfallwahrscheinlichkeit

Die Ausfallwahrscheinlichkeit beschreibt die Wahrscheinlichkeit, dass eine Komponente zu einem Zeitpunkt  $t$  das funktionsgerechte Verhalten nicht bereitstellen kann. In der ISO 26262 wird eine exponentielle Fehlerverteilung angenommen. Damit ergibt sich die Ausfallwahrscheinlichkeit  $Q$  als

$$Q(t) = 1 - e^{-\lambda t}$$

mit der Ausfallrate  $\lambda$ . Die **Zuverlässigkeit** (engl. reliability) ist das Komplementäre der Ausfallwahrscheinlichkeit  $1 - Q$  und beschreibt demnach die Wahrscheinlichkeit, dass eine Funktion zum Zeitpunkt  $t$  nicht ausgefallen ist. Die Ausfälle in einem infinitesimalen Zeitintervall bezogen auf die intakten Einheiten definieren die Ausfallrate.

Die **Verfügbarkeit** (engl. availability) beschreibt nach ISO 26262 die Wahrscheinlichkeit, dass eine Funktion bei Anforderung bereitgestellt werden kann [Int18]. Die Verfügbarkeit bezieht sich auf den Betriebszustand, wohingegen die Zuverlässigkeit über die Lebensdauer betrachtet wird. Für nicht reparierbare Systeme sind diese identisch [BL04]. Die quantitative Zielgröße für die Wahrscheinlichkeit von Sicherheitszielverletzungen in der ISO 26262 [Int18] ist eine mittlere Ausfallwahrscheinlichkeit pro Stunde, über die Lebensdauer berechnet.

### 2.1.3 Der Sicherheitslebenszyklus nach ISO 26262

Der Sicherheitslebenszyklus beschreibt die einzelnen Aktivitäten und Arbeitsprodukte im Rahmen einer sicherheitsgerichteten Entwicklung nach ISO 26262 [Int18]. Der Sicherheitslebenszyklus umfasst dabei alle Teile der ISO 26262 entsprechend Abbildung 2.1. Abbildung 2.6 zeigt den Sicherheitslebenszyklus, wobei Managementprozesse und Aktivitäten nach dem Produktionsstart nicht dargestellt werden.

Dargestellt sind die Konzeptphase und die Produktentwicklungsphase. Der erste Schritt ist die Item Definition. Dabei wird der Betrachtungsumfang inklusive der Betriebsbedingungen, Schnittstellen, Einflussfaktoren und des Verhaltens auf Fahrzeugebene beschrieben [Ros16]. Durch die Impact Analyse wird bei Weiterentwicklungen die Auswirkungen von Änderungen bewertet. Diese Analyse entfällt bei Neuentwicklungen. Ein Kernaspekt der ISO 26262 ist die Gefahren- und Risikoanalyse, die in der Formulierung von Sicherheitszielen und deren ASIL-Bewertungen resultiert. Dabei werden Fehlerauswirkungen auf Fahrzeugebene in relevanten Fahrsituationen bezüglich der Kontrollierbarkeit (engl. controllability) und der Schadensschwere (engl. severity) bewertet. Außerdem wird die Auftretenswahrscheinlichkeit (engl.

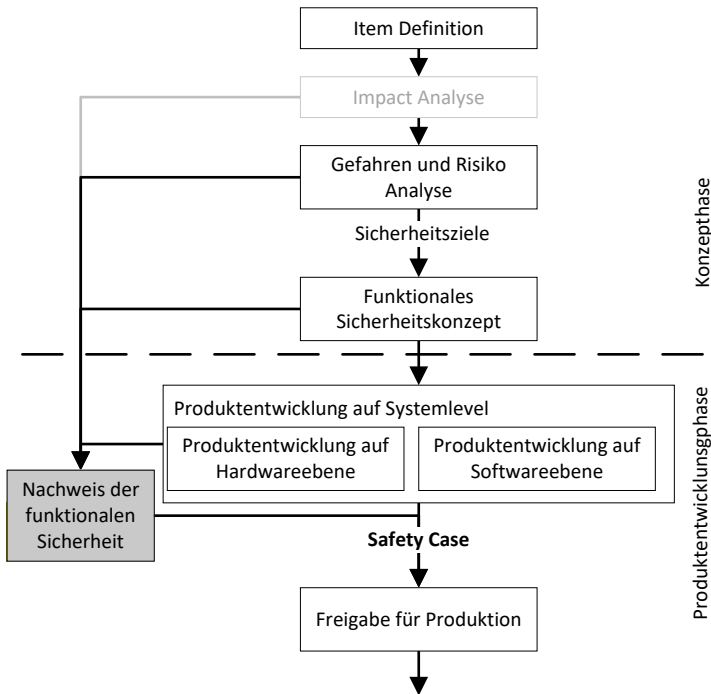


Abbildung 2.6: Phasen des Sicherheitslebenszyklus nach ISO 26262 [Int18]

exposure) der Fahrsituation, im Frequenzbereich für diskrete Ereignisse und im Zeitbereich für kontinuierliche Ereignisse, berücksichtigt [Ros16]. Das Verhindern der Gefährdung wird als Sicherheitsziel, einer Anforderung auf Systemebene, formuliert. Aus der Bewertung ergibt sich ein Maß für das Risiko, welches in der Ableitung eines Integritätslevels (ASIL) resultiert. Die nächste Phase stellt das funktionale Sicherheitskonzept dar. Dabei werden Sicherheitsanforderungen aus den Sicherheitszielen abgeleitet und auf funktionale Elemente des Systems allokiert. Dabei können neben den Architekturelementen des Systems auch externe Maßnahmen berücksichtigt werden. [Int18] Im Rahmen der Produktentwicklung erfolgt die Allokation der funktionalen Sicherheitsanforderungen auf die technische Systemarchitektur. Das technische Sicherheitskonzept berücksichtigt Hardware und



Softwareumfänge, sowie deren Schnittstellen. Abbildung 2.7 zeigt die Entwicklungsschritte anhand des V-Modells nach ISO 26262 am Beispiel der Softwareentwicklung [Int18].

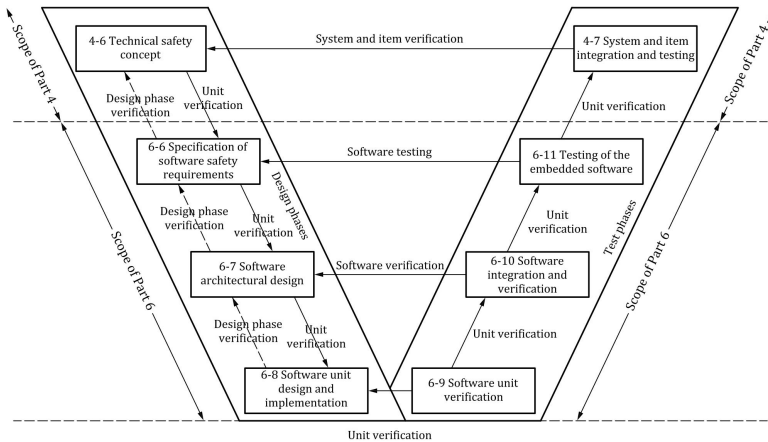


Abbildung 2.7: Schritte der sicherheitsgerichteten Produktentwicklung anhand des V-Modells nach ISO 26262 am Beispiel der Softwareentwicklung [Int18]

Im Rahmen der Produktentwicklung werden ausgehend von technischen Sicherheitsanforderungen die Anforderungen an die Software formuliert, die Architektur entsprechend abgeleitet und die einzelnen Softwarebausteine entwickelt und implementiert. Der Nachweis der anforderungsgemäßen Umsetzung erfolgt mittels Verifikation (Design phase verification). Auf der rechten Seite erfolgen die Verifikation und Integration inklusive dem Testen. Das Vorgehen gilt in analoger Weise für die Produktentwicklung auf Komponenten- und Systemebene. Neben den Verifikationsschritten ist auch eine Validierung, der Nachweis der Zweckerfüllung, notwendig. Im Rahmen dieser werden Prämissen und Annahmen für die Einstufung des Risikos und die Angemessenheit der Maßnahmen überprüft. Die Nachweise der einzelnen Arbeitsprodukte sowie Verifikations- und Validierungsergebnisse werden in einem Safety Case zu einer gesamtheitlichen Sicherheitsargumen-

tation zusammengefasst (vgl. Abbildung 2.6). Dieser stellt die Grundlage für die Produktionsfreigabe dar.

## 2.2 Sicherheitsargumentation und Safety Case

Gemäß ISO 26262 ist die Aufgabe des Safety Cases die belastbare und nachvollziehbare Argumentation der Erfüllung funktionaler Sicherheit. Dabei stellt die Argumentation den Zusammenhang zwischen den Sicherheitszielen und Sicherheitsanforderungen und den einzelnen Arbeitsprodukten, wie Sicherheitskonzepten, Analysen und Validierungen, dar. [Int18] Der Safety Case ist die Basis für die Freigabe des Produktes und ein zentrales Arbeitsprodukt des Sicherheitslebenszyklus. Sicherheitsaktivitäten wie Analysen und Nachweise unterstützen die Argumentation. [Ros16] Der Safety Case muss konsistente und adäquate, sowie gemäß den Zielen und Anforderungen, vollständige Nachweise der Sicherheit liefern [Hil12].

Die Dokumentation des Safety Cases kann nach ISO 26262 narrativ, graphisch oder in tabellarischer Form erfolgen. Auch Kombinationen sind möglich. Die ISO 26262 [Int18] schlägt die Goal Structuring Notation als eine graphische Form der Argumentation vor. Die Goal Structuring Notation ist eine von Kelly [Kel98] vorgestellte graphische Darstellung einer Sicherheitsargumentation in einer nachvollziehbaren und verständlichen Form. Das Konzept und die wichtigsten Elemente werden im Folgenden vorgestellt. Abbildung 2.8 führt die im Rahmen der Goal Structuring Notation verwendeten Symbole entsprechend der Definition von Kelly [Kel98] ein.

Ausgangspunkt der Sicherheitsargumentation sind Ziele. Das sind sowohl Sicherheitsziele als auch Sicherheitsanforderungen, die das System erfüllen muss. Mittels Strategien werden Ziele auf Unterziele herunter gebrochen. Lösungen stellen Nachweise der Zielerfüllung dar. Das können Analysen, Testberichte, Experteneinschätzungen oder Ähnliches sein. Mit Kontext-Elementen werden einzelne Elemente in Beziehung zu Argumenten oder Randbedingungen außerhalb der eigentlichen Argumentation gesetzt. Be-

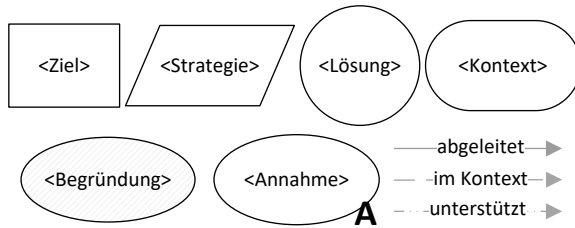


Abbildung 2.8: Symbole der Goal Structuring Notation nach Kelly [Kel98]

gründungen erklären Zusammenhänge und dienen der Nachvollziehbarkeit. Darüber hinaus sind Annahmen notwendig. Diese stellen beispielsweise Prämissen an Schnittstellen oder Vorgaben und Richtlinien dar. Auch die Relation zwischen den Elementen verdeutlichen die Struktur und damit die Argumentation. Dabei wird zwischen Relationen für die Ableitungen von Zielen, Relationen, die den Kontext aufzeigen und Relationen, welche die Argumentation durch Begründung oder Nachweise unterstützen, unterschieden. [Kel98]

## 2.3 Fail-Operational-Fahrzeugsysteme

Wie bereits einleitend erläutert (vgl. Kapitel 1.1) erfordert hochautomatisiertes Fahren Fail-Operational-Fahrzeugführungen. Im Folgenden wird das Fail-Operational-Verhalten definiert und im Anschluss ein mehrkanaliges System mit einer Betriebsstrategie und relevanten Sicherheitszielen vorgestellt.

Fail-Operational wird von Isermann [Ise16] als die Fähigkeit eines Systems nach einem Fehler weiterhin betriebsfähig zu sein beschrieben. Ein solches Verhalten ist erforderlich, wenn unmittelbar nach dem Ausfall kein sicherer Zustand möglich ist. In der ISO 26262 wird der Begriff Fehlertoleranz analog als die Fähigkeit der Funktionsbereitstellung in Gegenwart eines oder mehrerer Fehler beschrieben [Int18]. Nach Schnellbach [Sch16] bedeutet Fail-Operational, eine definierte Funktion, für eine definierte Zeit,

auch in Gegenwart einer bestimmten Menge an Fehlern, bereit zu stellen. Das fehlertolerante Verhalten muss daher nicht für eine unbegrenzte Zeit und nicht in gleichem Umfang wie im fehlerfreien Zustand sichergestellt werden. Fail-Silent Verhalten liegt bei einer Passivierung des fehlerhaften Elementes vor, bei der weitere Elemente nicht beeinflusst werden. Von einem Fail-Safe Verhalten wird gesprochen, sofern ein sicherer Zustand, aktiv oder passiv, erreicht wird [Ise16, Sch16].

Fehlertolerante Systeme basieren auf Redundanzen, um im Fehlerfall ein funktionsfähiges System nutzen zu können [KST+19, Sch16]. Dabei wird zwischen dynamischer und statischer Redundanz unterschieden. Bei einer statischen Redundanz sind alle Systeme permanent aktiv und über einen Vergleich wird das Ausgangssignal festgelegt. Um Fehler zu identifizieren, sind demnach mindestens drei Pfade notwendig. Dynamische Redundanz erfordert eine geringere Anzahl an redundanten Systemen oder Wirkketten und stellt daher den kosteneffizienteren Ansatz für die Automobilindustrie dar [KKS+06]. Dabei ist ein System oder eine Wirkkette im Eingriff, während weitere Systeme in einem Standby-Betrieb sind. Wenn das aktive System einen Fehler detektiert, wird auf ein Rückfallsystem umgeschaltet. Daher sind zwei Pfade notwendig. Für den Standby-Betrieb wird zwischen einem Cold-Standby, wenn das Rückfallsystem zuerst hochfahren muss, und einem Hot-Standby, bei dem auch das Rückfallsystem in einem kontinuierlichen Betrieb und daher schneller übernahmebereit ist, unterschieden. [Ise16]

Die Systemarchitektur, auf der diese Arbeit basiert, wurde von Kron et al. [KST+19] vorgestellt. Abbildung 2.9 zeigt das zwei-kanalige System mit dynamischer Redundanz.

Das System besteht aus zwei Fail-Silent Kanälen, einem Nominalkanal und einem Rückfallkanal. Jeder Kanal stellt dabei eine Wirkkette dar, welche in der Lage ist die Fahrzeugführung eigenständig zu übernehmen. Ein solcher Kanal besteht aus Sensorik, der Bestimmung der Fahrstrategie, der Fahrzeugregelung und den Aktoren, Bremse und Lenkung. Der Antrieb ist nicht redundant und wird nur aus dem Nominalkanal angesteuert. Der Nominalkanal ist

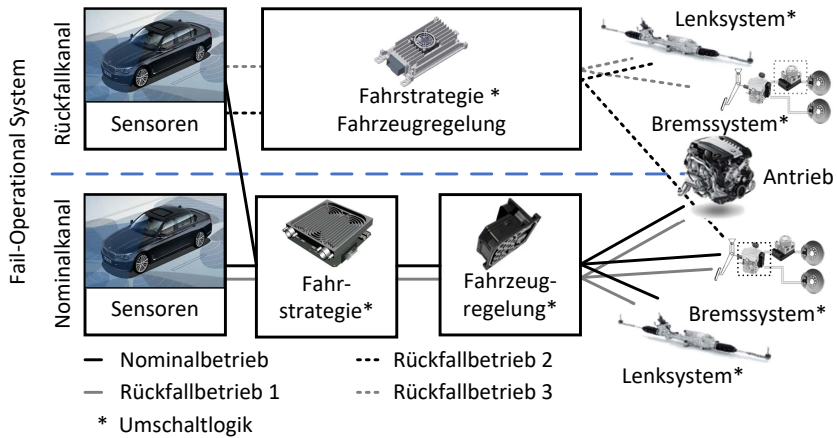


Abbildung 2.9: Fail-Operational-Fahrzeugführung nach Kron et al. [KST+19]

im fehlerfreien Betrieb und bei einem Fehler auf dem Rückfallkanal aktiv. [KST+19] Die Komponenten im Rückfallkanal sind auf den Rückfallbetrieb ausgelegt. Das bedeutet der Funktionsumfang ist geringer, da beispielsweise Komfort eine untergeordnete Rolle spielt. Entsprechend werden der Rückfallbetrieb auf dem Nominalkanal, Rückfallbetrieb 1, und der Rückfallbetrieb auf dem Rückfallkanal, Rückfallbetrieb 3, unterschieden. Zusätzlich kann die primäre Aktorik, in Abbildung 2.9 das primäre Bremssystem, aus dem Rückfallkanal angesteuert werden. Die Ansteuerung kann direkt oder über das sekundäre Bremsmodul erfolgen und sowohl durch die Umschaltlogik als auch auf Komponentenebene umgesetzt werden. In Abbildung 2.9 ist der Betriebsmodus als Rückfallbetrieb 2 dargestellt. Die Kommunikation, sowie die Energieversorgung ist für jeden Kanal unabhängig, wobei auch eine Kommunikation zwischen den Kanälen erfolgt. Zwischen den Kanälen werden Signale für die Diagnosen und eine sprunglose Umschaltung ausgetauscht. Ein Konzept einer Überwachung und Diagnose basierend auf einer redundanten Architektur, einer Sicherheitsschicht, in dem die Trajektorien von Primär- und Sekundärkanal verglichen und auf Korrektheit geprüft wer-

den, stellen Mehmed et al [MSC20]. Hierfür ist ein Hot-Standby Zustand des Rückfallkanals notwendig. Eine mit der Arbeit von Kron et al. [KST+19] vergleichbare Architektur wird auch von Wolf und Oertel [WO20] sowie Niedballa und Reuss [NR20] vorgeschlagen. Letztere untersuchen zusätzlich ausfallsichere Netzwerktopologien. Die Umschaltung wird mittels einer Umschaltlogik umgesetzt. [KST+19] Diese ist dezentral partitioniert, um einen Ausfall durch Einzelfehler zu verhindern. Abhängig vom Zustand der einzelnen Komponenten bestimmt die Umschaltlogik die Systemkonfiguration und aktiviert, beziehungsweise deaktiviert die entsprechenden Komponenten. [Kue18] Die Einschränkung der möglichen Systemkonfigurationen auf den Nominal- und die Rückfallbetriebe limitieren mögliche Varianten und damit den Absicherungsaufwands. Umschaltlogiken können nach Friese [Fri21] innerhalb der Komponenten, auf Element-Ebene, auf Komponenten- oder Systemebene umgesetzt werden. Das System nach Kron et al. [KST+19] und Kümmel [Kue18] stellt dabei eine Umschaltung auf System- und auf Komponentenebene, implementiert die Priorisierung des Bremssystems, dar. Abbildung 2.10 zeigt diese für die Fail-Operational-Fahrzeugführung.

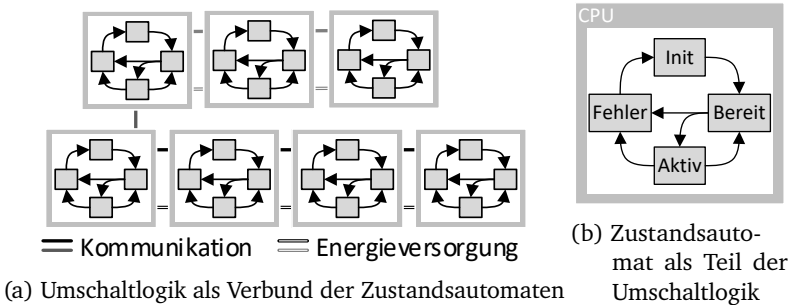


Abbildung 2.10: Umschaltlogik für eine Fail-Operational-Fahrzeugführung nach Kron et al. [KST+19]

Auf der linken Seite ist die Umschaltlogik als Verbund der Zustandsautomaten gezeigt. Die Zustandsautomaten sind auf die Komponenten des Systems (vgl. Abbildung 2.9) verteilt und durch das gegenseitige Einlesen der Zustände gekoppelt. Abhängig vom Zustand der eigenen Komponente,

externen Aktivierungs- beziehungsweise Deaktivierungsanforderungen und den Zuständen der anderen Zustandsautomaten, erfolgen die Transitionen der Automaten. Abbildung 2.10b zeigt die vereinfachte Darstellung eines Zustandsautomaten. Dieser besitzt einen *Init* Zustand und die Zustände *Bereit*, *Aktiv* und *Fehler*. Nach dem Aufstarten der Komponente befindet sich der Automat in *Init*; liegt kein Fehler vor wechselt er in *Bereit*. Erfolgt eine Aktivierung der hochautomatisierten Fahrfunktion, liegt kein Fehler vor und die anderen Zustandsautomaten signalisieren ebenfalls *Bereit* wechselt der Zustand zu *Aktiv*. Im Fehlerfall wird der Zustand *Fehler* aktiviert. Ein erneutes Aktivieren der Fahrfunktion ist nur nach einem Klemmenwechsel möglich.

Für den Rückfallbetrieb in einem Fail-Operational relevanten Fahrmodus zeigt die ISO 26262 eine mögliche Betriebsstrategie, um das Risiko und damit die notwendige Integrität im Rückfallbetrieb zu reduzieren. Diese wird im Folgenden erklärt und dient als Basis für die Formulierung der Sicherheitsziele. Im fehlerfreien Nominalbetrieb wird das System im Rah-

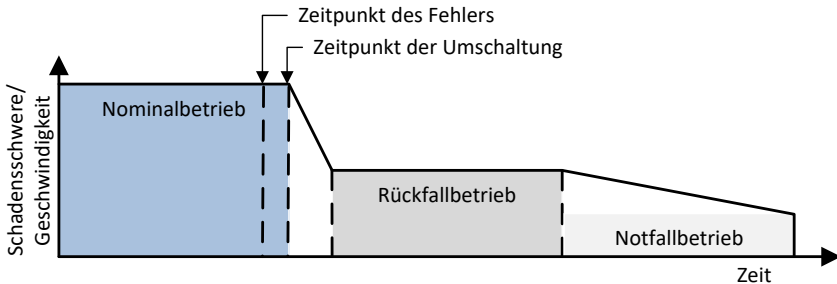


Abbildung 2.11: Fail-Operational-Betriebsstrategie nach ISO 26262 [Int18]

men der Auslegungsgrenzen betrieben. Dies erfolgt bis zu einer maximalen Geschwindigkeit, wodurch sich eine entsprechende Schadensschwere ergibt. Im Fehlerfall ist eine gewisse Zeit zur Diagnose und Umschaltung auf die Rückfallkonfiguration notwendig. Das Zeitintervall muss dabei geringer als die Fehlertoleranzzeit sein. Die ISO 26262 ermöglicht eine Umsetzung der einzelnen Kanäle mit geringerer Integrität, sofern eine Verzögerung zur Limi-

tierung der Schadensschwere im Rückfallbetrieb erfolgt. Bedingung hierfür ist, dass die Verzögerung zum Erreichen des Systemzustands entsprechend der ursprünglichen ASIL Einstufung umgesetzt ist und die Übergangszeit klein ist. [Int18] Im Notfallbetrieb erfolgt eine weitere Verzögerung.

Basierend auf dieser Betriebsstrategie und den Arbeiten von Kölbl und Leue [KL18] und Stolte et al. [SBRM06] werden relevante Sicherheitsziele in Tabelle 2.1 präsentiert. Diese stellen eine Auswahl ohne den Anspruch auf Vollständigkeit dar. Die Tabelle zeigt neben dem Sicherheitsziel auch den Betriebsmodus und die ASIL Einstufung. Die Risikobewertung erfolgt entsprechend der Literatur.

Im Rahmen der Gefahren- und Risikobewertung ergibt sich für die hochautomatisierte Fahrfunktion die geringste Kontrollierbarkeit, da nicht von einem Fahrereingriff ausgegangen werden kann. Auch die Auftretenswahrscheinlichkeit wird im Sinne der Gefahren und Risikobewertung als maximal angenommen, da die Fehler in üblichen Fahrsituationen, beispielsweise einer Autobahnfahrt, relevant sind. Die Schadensschwere ist geschwindigkeitsabhängig. Daher ergibt sich für Sicherheitsziele, die einen Verlust der Fahrzeugführung im Nominalbetrieb zur Folge haben, eine Integrität von ASIL D. Das umfasst sowohl die Umschaltlogik als auch das Folgen der Trajektorie. Für den Rückfallbetrieb wird aufgrund der Verzögerung entsprechend der ISO 26262 ein ASIL B abgeleitet. Die Verzögerung auf die entsprechende Geschwindigkeit muss jedoch, wie beschrieben, mit ASIL D umgesetzt werden. Gefahren, die zu einer Degradation inklusive der fehlerhaften Aktivierung führen, werden mit ASIL B bewertet.

Für das Fail-Operational-System, wie in Abbildung 2.9 gezeigt, ergibt sich daher eine Umsetzung der Rückfallebene, mit Ausnahme der Verzögerung, mit ASIL B. Für den Nominalbetrieb und den entsprechenden Nominalkanal ist eine Integrität bezüglich der genannten Sicherheitsziele mit ASIL D erforderlich. Da die Dauer des Rückfallbetriebes so bemessen ist, dass durch die Wahrscheinlichkeit der Fahrerübernahme die Kontrollierbarkeit vollständig gegeben ist, besitzt der Notfallbetrieb keine Integritätsanforderung.



Tabelle 2.1: Sicherheitsziele des Fail-Operational-Systems

Betriebsmodus	Sicherheitsziel	ASIL
Nominalbetrieb	Der Betrieb darf nicht aktiviert werden, wenn die Bereitschaft aller Komponenten nicht gegeben ist.	B
Nominalbetrieb	Der Betrieb darf nicht fälschlicherweise deaktiviert werden.	D
Nominalbetrieb	Eine Kollision durch Verlassen der Trajektorie muss verhindert werden.	D
Nominalbetrieb	Die Umschaltlogik muss nach einem Fehler einen funktionsfähigen Rückfallbetrieb aktivieren.	D
Rückfallbetrieb	Das System muss nach einer Umschaltung auf eine Geschwindigkeit kleiner $v$ verzögern.	D
Rückfallbetrieb	Der Betrieb darf nicht fälschlicherweise deaktiviert werden.	B
Rückfallbetrieb	Eine Fahrerübernahme muss bei Erreichen der Systemgrenzen oder eines Fehlers erfolgen.	B
Rückfallbetrieb	Eine Kollision durch Verlassen der Trajektorie muss verhindert werden.	B
Notfallbetrieb	Eine Kollision durch Verlassen der Trajektorie muss verhindert werden.	QM

## 2.4 Sicherheitsanalysen

Ziel der Sicherheitsanalysen ist nach ISO 26262 der Nachweis eines ausreichend geringen Risikos der Verletzung von Sicherheitszielen. Zu diesem Zweck müssen mögliche Fehler, Ursachen und Auswirkungen, insbesondere Gefährdungen, identifiziert, bewertet und Sicherheitsmaßnahmen abgeleitet werden. Darüber hinaus werden Analysen genutzt um Design und Sicherheitskonzepte zu verifizieren und damit den Nachweis für deren Eignung zu führen. [Int18]

Im Folgenden werden grundlegende Methoden der Sicherheitsanalysen eingeführt. Dabei wird zwischen klassischen, etablierten Methoden, Methoden zur Analyse von Fehlerpropagationen und formalen Methoden unterschieden.

#### 2.4.1 Klassische Analysen

In der ISO 26262 werden deduktive und induktive Sicherheitsanalysen empfohlen. Bei der Deduktion wird vom Allgemeinen auf das Individuelle geschlossen. Das bedeutet, bei deduktiven Analysen, wie beispielsweise der Fehlerbaumanalyse (engl. Fault-Free Analysis (FTA)), wird der Betrachtungsrahmen auf die allgemeinen Prämissen, zum Beispiel Gefahren, eingeschränkt. Umgekehrt wird bei der Induktion aus einem speziellen Zusammenhang, beispielsweise einer Beobachtung, auf allgemeine Zusammenhänge geschlossen. Bei einer induktiven Analyse ist das Ergebnis, beispielsweise die mögliche Gefährdung, nicht vorab eingeschränkt. Dadurch entstehen allerdings, insbesondere bei der Analyse komplexer Systeme, große Betrachtungsumfänge. Eine induktive Methode ist die Fehlermöglichkeits- und Einflussanalyse (FMEA). Markov-Analysen können sowohl vom Start-, induktiv, als auch vom Endzustand aus, deduktiv, konstruiert werden. [Eri16]

Im Folgenden werden die Fehlerbaumanalyse, Markov-Analysen und die Fehlermöglichkeits und Einflussanalyse als in der funktionalen Sicherheit etablierte Methoden vorgestellt. Darüber hinaus wird kurz auf weitere Methoden eingegangen.

#### **Fehlerbaumanalyse**

Die Fehlerbaumanalyse wurde Mitte des 20. Jahrhunderts zur Analyse von Raketensystemen entwickelt. Heute werden Fehlerbäume in vielen Bereichen, unter anderem auch in der Automobilindustrie, eingesetzt. [ESH15] Durch die logische Verknüpfung eines Hauptereignisses mit Ursachen und deren weiterer Detaillierung, entsteht ein Fehlermodell für das definierte Hauptereignis. Ein unerwünschtes Hauptereignis stellt beispielsweise

die Verletzung eines Sicherheitsziels dar. [ESH15] Zur Verknüpfung der Ereignisse werden logische Gatter verwendet [Eri16]. Dabei können sowohl Einfach- als auch Mehrfachfehler berücksichtigt werden [ESH15]. Die graphische Darstellung unterstützt das Verständnis für System- und Fehlerzusammenhänge. Außerdem kann durch die Berücksichtigung von Wahrscheinlichkeiten der Basisereignisse das Auftreten der Folgeereignisse und des Hauptereignisses quantitativ bestimmt werden. [Eri16] In Abbildung 2.12 werden die wichtigsten logischen Verknüpfungen, auch Gatter genannt, gezeigt.

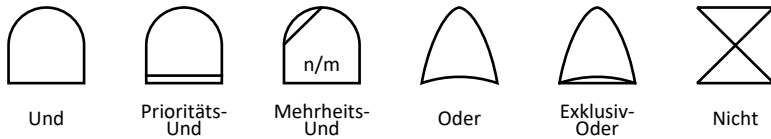


Abbildung 2.12: logische Gatter bei der Fehlerbaumanalyse nach Edler [ESH15]

Das Und-Gatter symbolisiert ein logisches *Und*. Bei einer quantitativen Analyse werden die Wahrscheinlichkeiten der Basisereignisse multipliziert. Sofern die Reihenfolge eine Rolle spielt wird dies durch das *Prioritäts-Und* dargestellt. Mehrheitsgatter werden bei durch das Eintreten von  $n$  aus  $m$  Basisereignissen erfüllt. Das logische *Oder* führt beim Eintreten mindestens eines Basisereignisses zu einem Fehler. Wenn das Eintreten von genau einem Ereignis zum Fehler führt, wird das *Exklusiv-Oder* verwendet. Eine Negierung eines Ereignisses wird durch das Nicht-Symbol modelliert. Abbildung 2.13 zeigt ein einfaches Beispiel eines Fehlerbaums, anhand dessen die Modellierung erklärt wird.

Die Konstruktion erfolgt schrittweise, ausgehend vom Hauptereignis. Für jedes Ereignis werden im Rahmen der Analyse die Ursachen identifiziert und diese logisch verknüpft. Für Basisereignisse können die quantitativen Daten für die Wahrscheinlichkeiten definiert werden. Im Beispiel in Abbildung 2.13 wird das Sicherheitsziel durch zwei Ereignisse verletzt. Zum einen führt ein Fehler im Bremssystem und zum anderen ein Fehler in der Spannungsver-

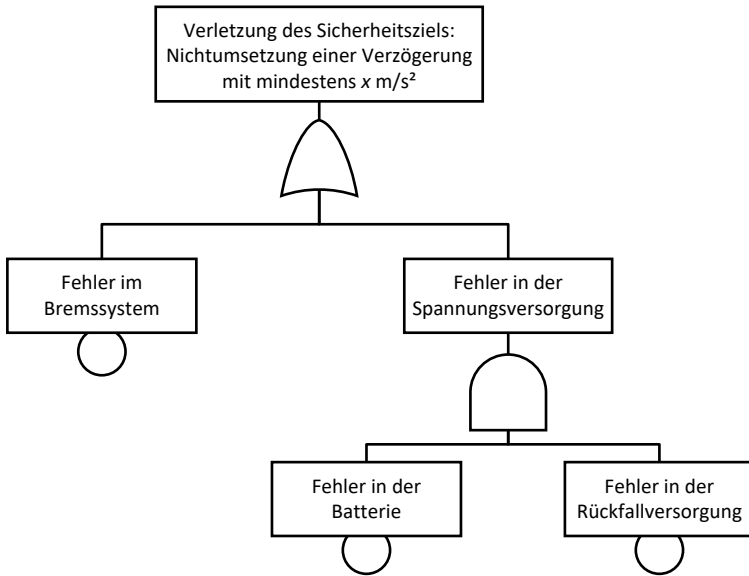


Abbildung 2.13: Beispiel eines Fehlerbaumes nach Edler [ESH15]

sorgung zu einem Ausfall. Die Verknüpfung erfolgt dabei durch ein logisches *Oder*. Der Fehler im Bremssystem stellt ein Basisereignis dar, ein Ereignis, welches im Rahmen der Analyse nicht weiter detailliert wird. Der Fehler in der Spannungsversorgung tritt durch die Kombination eines Fehlers der Batterie und eines Rückfall-Systems, wie beispielsweise Kapazitäten, ein. Diese Ereignisse sind durch ein logisches *Und* verknüpft. Beide Ereignisse müssen eintreten damit ein Energieausfall auftritt. [ESH15]

In nicht-kontinuierlichen Fehlerbäumen wird die Wahrscheinlichkeit aller Ereignisse zu einem diskreten Zeitpunkt, meist zum Ende der Lebensdauer berechnet. Minimalschnitte stellen dabei die kleinste Menge der Ereignisse dar, die zu einem Ausfall führen. [Eri16] Die Analyse erfolgt durch die Berechnung der Einzelereignisse und Kombination entsprechend der booleschen Logik für die Minimalschnitte. [Eri16] Auch kontinuierliche Wahrscheinlichkeiten können durch Integration über die Zeitintervalle zu diskreten Zeitpunkten ausgewertet werden. [RS15, VGRH81]

Vorteile der Methode sind unter anderem die strukturierte und intuitive Analyse, unterstützt durch die visuelle Darstellung. Es können systematische und zufällige Fehler, Einfach- und Mehrfachfehler berücksichtigt werden. Herausforderungen stellen große und komplexe Systeme dar, da die Modelle unter Umständen unübersichtlich werden. Außerdem kann die Modellierung von zeitlichen Aspekten kompliziert sein. [Eri16][ESH15]

### Markov-Analysen

Eine weitere Möglichkeit der quantitativen Fehleranalyse bieten Markov-Modelle. Diese gehen auf Andrei Markov und seine Erkenntnisse zur Beschreibung zufälliger Prozesse gegen Ende des 19. Jahrhunderts zurück [Eri16]. Markov-Analysen werden vorwiegend für die Bestimmung der Zuverlässigkeit und der Verfügbarkeit genutzt. [Eri16] Die Markov-Analyse basiert dabei auf der Beschreibung von Zuständen und Transitionen. Es werden Initialzustände, der Zustandsraum und die Transitionen mit den zugehörigen Übergangswahrscheinlichkeiten definiert [Eri16]. Abbildung 2.14 zeigt ein beispielhaftes Markov-Modell mit den genannten Elementen.

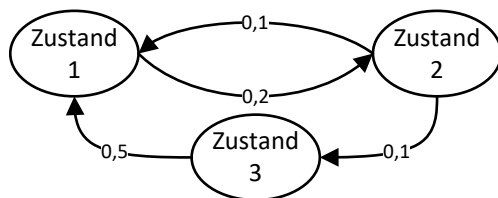


Abbildung 2.14: Beispiel eines Markov-Modells

Dargestellt ist ein System mit drei Zuständen und entsprechenden Transitionen inklusive deren Übergangswahrscheinlichkeiten zwischen 0 und 1. Berechnet werden die Wahrscheinlichkeiten jedes Zustands in diskreten Zeitschritten. Der Folgezustand hängt dabei, nach der Markov Eigenschaft, nur vom derzeitigen Zustand und den Übergangswahrscheinlichkeiten ab. Die statistische Wahrscheinlichkeit in einem bestimmten Zustand zu sein wird durch Differentialgleichungen beschrieben. Der Zusammenhang lautet  $\frac{dP(t)}{dt} = T \cdot P(t)$  wobei  $t$  die Zeit,  $P(t)$  die Matrix der Wahrscheinlichkeiten für

jeden Zustand und  $T$  die Transitionsmatrix mit den Übergangswahrscheinlichkeiten darstellt [Eri16]. Bei stationären Systemen tritt ein Zeitpunkt ein, ab dem die Wahrscheinlichkeiten der Zustände konstant bleiben.

Markov-Modelle bieten die Möglichkeit einer exakten Bestimmung von Wahrscheinlichkeiten auch für dynamische und stochastische Prozesse [Eri16]. Allerdings werden die Modelle für komplexe Systeme umfangreich und damit unübersichtlich, schwer nachvollziehbar und daher fehleranfällig weswegen Dugan [DBB93] diese zur Analyse solcher Systeme für ungeeignet hält.

### Fehlermöglichkeits- und Einflussanalyse

Die FMEA (engl. Failure Mode and Effect Analysis) stellt eine induktive Analyse­methode dar. Die Methode stammt ursprünglich aus dem Militärbereich und wurde Mitte des 20. Jahrhunderts entwickelt [Eri16]. Abbildung 2.15 zeigt die einzelnen Schritte anhand der jeweiligen Betrachtungsumfänge.

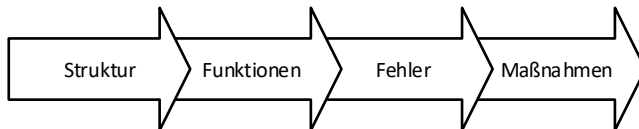


Abbildung 2.15: Vorgehen bei der Fehlermöglichkeits- und Einflussanalyse nach dem VDA [Ver17]

Die Modellierung der Systemstruktur bildet die Ausgangsbasis. Den Strukturelementen werden daraufhin Funktionen zugewiesen und diese zu einem Funktionsnetz anhand von Ursachen und Folgen verknüpft. Für die Funktionen werden mögliche Fehlfunktionen identifiziert und basierend auf dem Funktionsnetz ebenfalls vernetzt. So lässt sich die Auswirkung eines Fehlers im System nachvollziehen und beurteilen. [Wer12] Basierend auf der Fehleranalyse erfolgt die Risikoanalyse. Dabei wird das Risiko eines Fehlers anhand der Bedeutung, der Auftretenswahrscheinlichkeit und der Entdeckungswahrscheinlichkeit bewertet. Im Anschluss werden adäquate Maßnahmen zur Entdeckung, Vermeidung und Behebung abgeleitet, die in der Bewertung berücksichtigt werden. Die FMEA kann dabei sowohl entwicklungsbegleitend

als auch zur Verifikation auf Produkt- und Prozessebene verwendet werden, ist allerdings auf die Analyse von Einfachfehlern beschränkt. [Ver17]

### Weitere Analysemethoden

Weitere Ansätze stellen die Hazard and Operability Analysis (HAZOP) und Hierarchically Performed Hazard Origin and Propagation Studies (Hip-Hops) dar. Bei der HAZOP werden Leitworte genutzt, um mögliche Fehlermodi eines Systems zu identifizieren. Dafür werden Systemparameter und Variablen mit Abweichungen kombiniert und die Kritikalität evaluiert. Beispielsweise würde ein *zu hohes Geschwindigkeitssignal* eine solche Kombination darstellen. Durch die Verwendung der Leitworte, wie *zu hoch*, ist kein detailliertes technisches Verständnis für die Analyse notwendig, die Leitworte müssen aber die relevanten Fehlermodi beinhalten. [Eri16]

Hip-Hops stellen eine Kombination von FTA und FMEA sowie einer funktionalen Fehleranalyse dar. Dabei wird Letztere auf Fahrzeugebene durchgeführt. Durch eine FMEA, die auf Schnittstellen fokussiert ist, werden Fehlermodelle für die jeweiligen Komponenten erstellt. Die funktionale Analyse dient als Ausgangsbasis zur Konstruktion von Fehlerbäumen, die FMEAs werden genutzt, um die Fehlerbäume zu detaillieren. Damit können in Hip-Hops auch Mehrfachfehler berücksichtigt werden. [PMSH01]

## 2.4.2 Analyse der Fehlerpropagation

Neben den bereits vorgestellten Methoden, sowie den in Kapitel 2.4.3 adressierten formalen Methoden, gibt es Ansätze explizit für die Analyse der Fehlerpropagation. Dabei werden in der Literatur vor allem Modell- und Graphen-basierte Ansätze vorgeschlagen. Das folgende Kapitel stellt eine kurze Übersicht dar.

Systems Engineering beschäftigt sich mit der Entwicklung komplexer Systeme und deren eindeutiger Beschreibung. Hierfür werden modellbasierte Ansätze unter Verwendung spezifizierter Modellierungssprachen, die die notwendigen Entwicklungsschritte unterstützen, entwickelt. [BB19] Modellbasierte Sicherheitsanalysen nutzen die im Rahmen der Entwicklung

erstellten Modelle als Basis für Fehlermodelle [JMHW05]. Insbesondere wird die System Modeling Language (SysML) und die Architecture Analysis and Design Language (AADL), beziehungsweise Electronics Architecture and Software Technology - Architecture Description Language (EAST-ADL), als Beschreibungssprache genutzt [BB19, Sar19].

SysML stellt eine Erweiterung der Unified Modeling Language (UML) dar und ist eine graphische Beschreibungsmöglichkeit von Systemen. Dazu werden verschiedene Diagramme oder Systemsichten wie Verhaltensdiagramme, Anforderungsdiagramme oder Blockdiagramme genutzt. [FMS12] Neben der anforderungsbasierten Entwicklung, welche in der SysML unterstützt wird, wurden Erweiterungen der SysML um Gefahren und Risiken eingeführt [BSK16]. Weitere Ansätze stellen die Ableitung von Fehlermodellen aus den Diagrammen dar [WBE+15]. So werden aus Block- und Flussdiagrammen anhand der Schnittstellen im Modell Fehlerbäume generiert [MNKC04]. Eine weitere, verbreitete Modellierungssprache ist AADL. Die AADL dient der Beschreibung der Architektur von Software und Hardwaresystemen, Schnittstellen und zeitlichen Aspekten. Für Automobilsysteme wird vor allem die EAST-ADL genutzt. Diese basiert auf einem Ebenenkonzept zur Beschreibung verschiedener Detaillierungen inklusive der Fahrzeugebene und funktionaler Beschreibungen. [JMHW05] Auch Sicherheitskonzepte werden mittels EAST-ADL beschrieben. Die Integration der Sicherheitsbetrachtung erfolgt dabei analog, durch Erweiterung der Beschreibung und Ableitung von Fehlermodellen aus der modellbasierten Beschreibung. [Sar19]

Darüber hinaus werden Graphen-basierte Ansätze zur Analyse von Fehlerpropagation genutzt. Die wichtigsten Ansätze werden anhand der Modellbeschreibungen kurz vorgestellt.

Petri-Netze stellen solche Graphen-basierten Ansätze dar. Dabei sind insbesondere Timed Petri Nets (TPNs) für die Untersuchung der Fehlerpropagation relevant [BH08]. Übergänge sind nur bei Erfüllung von Vorbedingungen, die durch Token repräsentiert werden und unter Umständen eine globale Zeit beinhalten, möglich [Eri16]. Eigenschaften von Petri-Netzen werden durch temporale Logiken beschrieben [BH08]. Die Auswertung erfolgt durch



die Lösung eines abgeleiteten Gleichungssystems [Eri16].

Eine weitere Modellierungsmöglichkeit stellen Bayesische Netze dar [TTS98]. Dies sind gerichtete Graphen, bei denen Knoten Systemvariablen darstellen. Zusätzlich wird jedem Wert der Variable eine Wahrscheinlichkeit zugeordnet. Durch den gerichteten Graphen ergibt sich eine Wahrscheinlichkeit für jede Variablenkombination, gegen die Sicherheitsanforderungen geprüft werden können. [Nyb13]

Bei der Failure Propagation and Transformation Notation (FPTN) werden Komponenten mittels Knoten modelliert und deren Ein-Ausgangsverhalten, das Fehlerpropagationsverhalten, durch boolesche Logik, entsprechend aufgebaut. Durch die Methode können auch komplexe, modulare Systeme analysiert werden [FH94].

Timed Failure Propagation (TFP) Graphen berücksichtigen zwei Arten von Knoten, Fehlerursachen (engl. root causes) und Abweichungen. Durch die Kanten des gerichteten Graphen werden zeitliche Aspekte wie Verzögerungen durch Intervalle modelliert. [BBC17, MSU+92] Die Evaluation erfolgt mittels Suchverfahren [MSU+92]. Auch Model-Checking Ansätze werden zur Verifikation genutzt [BBC17].

### 2.4.3 Formale Methoden

Neben klassischen, in der funktionalen Sicherheit etablierten Analysen, sind formale Methoden und insbesondere das Model-Checking für diese Arbeit relevant. Deren Grundlagen werden daher an dieser Stelle vorgestellt.

Formale Methoden stammen aus der Softwaretechnik und werden seit den 1970er Jahren kontinuierlich weiterentwickelt. Ziel ist ein Beweis der Korrektheit, der vollständigen Erfüllung einer Eigenschaft. Im Gegensatz zum Testen, kann mittels Beweisen die Fehlerfreiheit von Systemen gezeigt werden. [Kle09]

Formale Methoden werden dabei in axiomatische Methoden und modellbasierte Methoden unterteilt [O'R17]. Erstere basieren auf der Definition von formalen Spezifikationen und der mathematischen Beschreibung der

Eigenschaften. Durch logische Beweisführung, wie Umformungen und Schlussfolgerungen, wird die Erfüllung der Spezifikation analysiert. Ansätze werden dabei entsprechend der Probleme, beispielsweise das first-order Theorem Proving für Systeme erster Ordnung, eingeteilt. [BK08] Axiomatische Methoden sind für komplexe Systeme bisher nur begrenzt anwendbar, da die Lösungsfindung von der formalen Beschreibung abhängt [O'R17]. Darüber hinaus gibt es modellbasierte Ansätze, die auf der Prüfung eines abstrahierten Modelles beruhen [O'R17]. Im Folgenden wird auf das Model-Checking eingegangen. Die einzelnen Schritte des Verfahrens werden in Abbildung 2.16 dargestellt.

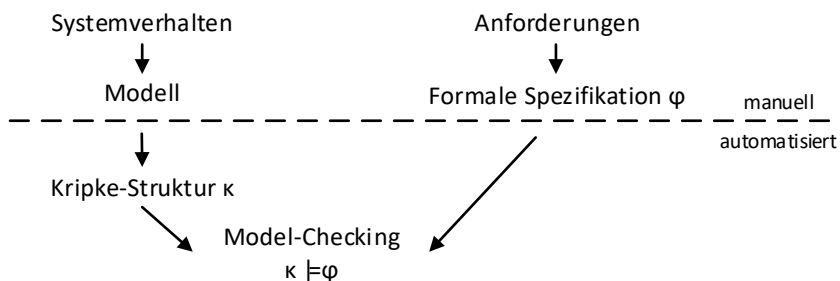


Abbildung 2.16: Schritte des Model-Checkings nach Clarke [CHVB18]

Sowohl Anforderungen an das Systemverhalten, beispielsweise funktionale- oder Sicherheitsanforderungen, als auch das Systemverhalten werden formal beschrieben. Das Systemverhalten wird in einer zustandsorientierten Darstellung modelliert. Model-Checking Programme übersetzen dies in eine Kripke-Struktur, eine sprachenunabhängige Darstellung des Zustandsraumes, bei der Übergangsbeziehungen mittels temporaler Logik, auf die am Ende des Kapitels eingegangen wird, berücksichtigt werden. Analog werden die Anforderungen an das System formal spezifiziert. [CHVB18] Bezüglich der Verfahren wird zwischen strukturellem und symbolischem Model-Checking, sowie Abstraktionsmethoden unterschieden. [CHVB18] Strukturelle Methoden, wie das explizite Model-Checking, basieren auf einer Formulierung des expliziten Zustandsraumes in einer Kripke-Struktur. Abstraktionsme-

thoden nutzen Reduktionen der Kripke-Struktur, um den Zustandsraum zu minimieren. Die Kripke-Struktur berücksichtigt bei symbolischen Verfahren nicht jeden Zustand explizit, das Verhalten wird stattdessen durch eine symbolische Repräsentation, die mit einer Zusammenfassung einhergeht, dargestellt. Lösungsverfahren für das symbolische Model-Checking basieren auf binären Entscheidungsdiagrammen und Lösungen von Erfüllbarkeitsproblemen. Mittels Entscheidungsdiagrammen wird der Zustandsraum, beispielsweise systematisch anhand einer Baumstruktur, durchsucht. [O'R17] Abbildung 2.17 zeigt einen Zustandsbaum anhand dessen mögliche Suchstrategien dargestellt werden.

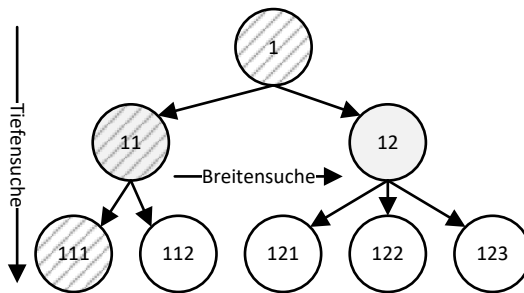


Abbildung 2.17: Suchstrategien in einem Zustandsbaum

Unterschieden wird dabei zwischen der Breitensuche (engl. Depth-First Search (DFS)) und der Tiefensuche (engl. Breadth-First Search (BFS)). Bei der Tiefensuche wird ein Ast (in Abbildung 2.17 gestreift dargestellt) bis zur Basis durchsucht, bevor der Nächste durchsucht wird. Bei der Breitensuche dagegen wird eine Ebene erschöpfend durchsucht, bevor die Prüfung der Ebene darunter erfolgt. [CHVB18] Auch Kombinationen der Suchstrategien werden genutzt. Gegenbeispiele, Verletzung von Spezifikationen, können mit der Tiefensuche unter Umständen schneller gefunden werden. Die Breitensuche findet in jedem Fall die kürzeste Zustandssequenz, die zu einer Verletzung führt. [BK08] Beim Durchsuchen des Zustandsraumes werden die Anforderungen für jeden Zustand geprüft. Der jeweilige Zustand wird mit den bereits untersuchten verglichen. Sofern der Zustand bereits gespei-

chert ist, werden Folgezustände nicht nochmals untersucht. [BK08] Beim On-The-Fly Model-Checking wird der Zustandsbaum beziehungsweise der Folgezustand erst zur Laufzeit bestimmt (vgl. Kapitel 3.3.2.2) [CHVB18]. Beim bounded-Model-Checking (vgl. auch Kapitel 3.3.2.1) wird bis zu einer definierten, endlichen Tiefe nach Gegenbeispielen in einer Kripke-Struktur gesucht. Ist ein Solches bis zur definierten Tiefe nicht erreichbar, gilt die Spezifikation als erfüllt. Das bedeutet, es erfolgt kein umfassender Nachweis. Probleme sind in der Anwendung jedoch meist auf relevante Zeitintervalle beschränkbar. [CHVB18] Lösungsmethoden basieren auf Satisfiability (SAT) Solvern, bestimmten Lösungsansätzen für Entscheidungsprobleme. Solche Ansätze sind, im Vergleich zu binären Entscheidungsdiagrammen, für die Lösung deutlich größere Probleme geeignet. [BCC+09, Bry86] Das Entscheidungsproblem ist gleichbedeutend mit der Suche nach einem Pfad, einer Sequenz von Zuständen  $T(s_i, s_{i+1})$  ausgehend von Initialzuständen  $I(s_0)$ , der die Spezifikationen  $p$  verletzt. Zur Lösung werden die Formulierungen der temporalen Logik in Darstellungen der Sequenzen überführt. [Bie09]

$$\exists s_0, \dots, s_k \quad I(s_0) \wedge \bigwedge_{i=0}^{k-1} T(s_i, s_{i+1}) \wedge \neg p(s_k)$$

Zur eigentlichen Lösung werden Algorithmen zur Pfad-Planung genutzt [Bie09]. Diese basieren auf der rekursiven Variation der Variablen [Bie09]. [BCC+09] Dabei spannen die Freiheitsgrade an jedem Knoten den möglichen Zustandsraum auf. [BCC+09] Es gibt verschiedene Ansätze der Lösungssuche [ES03]. Relevant für diese Arbeit ist insbesondere der Ansatz von Eén und Sörensson [ES03]. Implementierte Optimierungen stellen dabei gelernte Sequenzen dar, welche zu einem bekannten Zustand führen. Diese werden den Suchkriterien als Abbruchbedingungen hinzugefügt. Darüber hinaus werden Heuristiken und weitere Optimierungen genutzt, um eine Reihenfolge bei der Variation der Variablen zu treffen. Diese basieren beispielsweise auf der Häufigkeit der Aufrufe einzelner Variablen. [ES03] Für eine Übersicht zu aktuellen SAT Solvern wird auf Gong und Zhou [GZ17] verwiesen.

Tabelle 2.2: relevante Ausdrücke der linear temporalen Logik

<b>Operator</b>	<b>Bedeutung</b>
X	nächster
Y	vorheriger
F	schließlich
G	global
U	bis
H	historisch
O	einmalig

Wie in Abbildung 2.16 dargestellt, wird temporale Logik sowohl für die Formulierung der Kripke-Struktur als auch für die formale Spezifikation der Anforderungen genutzt. Die Beschreibung bietet die Möglichkeit zeitliche Abfolgen formal zu spezifizieren und stellt damit die Beschreibung einer Verbindung zwischen verschiedenen Rechenschritten dar. Dabei werden verschiedene logische Sprachen unterschieden. Linear temporale Logik geht von einem eindeutigen Nachfolger jedes Zustandes aus. Computational Tree Logic (CTL) bildet dagegen die Logik in Baumstrukturen ab. Ein Zustand kann mehrere Folgezustände haben. Die relevanten Formulierungen in Linear Temporal Logic (LTL) werden in Tabelle 2.2 vorgestellt. Dabei gelten die booleschen Variablen, wie *Und* (&), *Oder* (|) *exklusives Oder* (*xor*), Implikationen ( $\rightarrow$ ) oder *Ungleich* ( $\neq$ ). Die Definitionen werden in [CCJ+10] und [CHVB18] vorgestellt und entsprechend in dieser Arbeit verwendet.

Die logischen Beschreibungen folgen einer festen Syntax. Ein Beispiel ist der logische Ausdruck  $G(\phi \rightarrow X\psi)$ , wobei  $\phi$  der Ausgangszustand und  $\psi$  der Folgezustand ist. Der Ausdruck bedeutet: Wenn  $\phi$  eingetreten ist, muss immer  $\psi$  der nächste Zustand sein. Auch Intervalle für deren Gültigkeit können definiert werden. [CCJ+10]

Model-Checking stellt eine effektive Möglichkeit der Verifikation sicherheitskritischer Anwendungen dar [O'R17]. Allerdings können durch den Modellierungsschritt und die formale Spezifikation der Anforderungen Fehler induziert werden, die aufgrund der Automatisierung unentdeckt bleiben. [Kle09] Der limitierende Faktor bei der Anwendung formaler Methoden ist jedoch die Größe des Zustandsraums, woraus sich das Problem der Zustandsraumexplosion (engl. state-space explosion problem). [CHVB18] Dies führt zu hohem Speicherbedarf und zu langen Rechenzeiten und unter Umständen dazu, dass die Algorithmen nicht terminieren oder aufgrund der Speicherauslastung ungewollt abbrechen [Kle09].

# KAPITEL 3

## STAND DER WISSENSCHAFT

In diesem Kapitel wird anhand publizierter Literatur der Stand der Technik und der Stand der Wissenschaft vorgestellt. Auf diesem Wissen wird in den darauffolgenden Kapiteln aufgebaut. Dabei werden die Grenzen und Lücken der Literatur diskutiert.

Entsprechend dem Aufbau der Arbeit werden Veröffentlichungen zu Sicherheitsargumentationen, insbesondere im Kontext von Fail-Operational-Verhalten und der Norm ISO 26262, sowie Argumentationen mittels der Goal Structuring Notation diskutiert. Im Anschluss wird der Stand der Wissenschaft bezüglich Analysen gemeinsamer Ausfälle vorgestellt. Die Anwendung formaler Verifikation bei sicherheitskritischen Systemen als Nachweis der Korrektheit und für die Untersuchung der Fehlerpropagation wird darauffolgend präsentiert. Am Ende des Kapitels werden Arbeiten mit Relevanz für die quantitative Analyse dynamischer und mehrkanaliger Systeme diskutiert.

## 3.1 Relevante Sicherheitsargumentationen

Sicherheitsargumentationen und die Erstellung von Safety-Cases wurden in der Wissenschaft ausgiebig diskutiert. Im Folgenden wird zuerst auf Arbeiten, die die funktionale Sicherheit von Fail-Operational-Systemen betreffen, eingegangen. Darüber hinaus werden Sicherheitsargumentationen sicherheitskritischer Systeme vorgestellt. Der Fokus liegt dabei auf der Anwendung der Goal Structuring Notation im Kontext der ISO 26262. Arbeiten, die eine Sicherheitsargumentation auf Basis der Goal Structuring Notation für Fail-Operational-Systeme adressieren, finden sich in der Literatur bisher nicht.

### 3.1.1 Sicherheitsargumentationen für Fail-Operational-Systeme

Sicherheitsaspekte stellen einen Kernaspekt von Fail-Operational-Systemen dar und werden daher in Publikationen zu solchen Systemen behandelt. Eine hinreichende Sicherheitsargumentation wird in der Literatur nicht gegeben.

Die umfassendste Arbeit zu Fail-Operational-Fahrzeugsystemen stammt von Schnellbach [Sch16] aus dem Jahr 2016. Eine Literaturrecherche des Autors zeigt, dass Fail-Operational-Aspekte insbesondere in der Luft und Raumfahrt identifiziert worden sind, aber in der Automobilindustrie bis dato keine Betrachtung auf Systemebene erfolgte. In einem ersten Schritt analysiert Schnellbach die Erstversion der ISO 26262 von 2011 und identifiziert Erweiterungen der Aspekte Fehlertoleranz, Degradation, Rückfallbetriebszeit und die Integrität nach einem Erstfehler für Fail-Operational-Aspekte. Diese wurden in der ISO 26262:2018 [Int18] berücksichtigt und in Kapitel 2.3 eingeführt. Schnellbach schlägt außerdem die getrennte Berücksichtigung des Nominal- und Rückfallbetriebs in der Gefahren- und Risikoanalyse vor, um Degradationskonzepte wie Fahrerwarnung und Verzögerung zu berücksichtigen (vgl. Kapitel 2.3). Es wird vorgeschlagen, einhergehend mit dem sicheren Zustand, die Rückfallbetriebszeit nur zu begrenzen, wenn die



Integrität des Systems geringer als das Risiko ist. Darüber hinaus wird ein Modell für die Analyse der Unabhängigkeit vorgestellt, welches in der ISO 26262:2018 berücksichtigt wurde. Ein weiterer Beitrag ist die Analyse von Hardwarearchitekturen und die Ableitung einer solchen, inklusive notwendiger Redundanzen, basierend auf einem Optimierungsproblem. Dabei zeigt Schnellbach zum einen, dass dynamische Redundanz aufgrund der Kosten und ähnlicher Verfügbarkeit sinnvoll für den Einsatz in Fahrzeugen ist, und zum anderen vollständige Redundanz im Sinne der ISO 2626 nicht zwingend notwendig ist, um Fehlermetriken zu erfüllen. Die um Fail-Operational relevante Aspekte erweiterten Definitionen, wie Fehlertoleranz, wurden in der Überarbeitung der ISO 26262 von 2018 berücksichtigt, welche als Basis für diese Arbeit verwendet wird. Schnellbach [Sch16] hat relevante Aspekte für die Sicherheitsbetrachtung von Fail-Operational-Systemen identifiziert. Allerdings wurde keine hinreichende Sicherheitsargumentation unter Berücksichtigung der Analysen aufgezeigt und eine Umschaltlogik nicht adressiert. [Sch16]

Sari [Sar19] stellt in seiner Arbeit ein Vorgehen für die Entwicklung einer Fail-Operational-Architektur für das hochautomatisierte Fahren basierend auf Redundanzen vor. Die Betrachtung erfolgt für die Sensorik und die Verarbeitung bis hin zur Ansteuerung der Aktuatorik anhand eines Sicherheitskonzeptes gemäß der ISO 26262. Für hochautomatisiertes Fahren wird eine zwei-kanalige Architektur als ausreichend angesehen, da der Rückfallbetrieb begrenzt wird. Für diese wird eine Dekomposition der Sicherheitsziele vorgestellt. Im zweiten Teil der Arbeit wird ein Ansatz der modellbasierten Umsetzung des Sicherheitslebenszyklus präsentiert. Diese nutzt eine Beschreibung mittels Architecture Description Language (ADL). Die erzeugten Modelle und die Relationen der modellierten Objekte dienen als Basis für die Analyse abhängiger Fehler. Die Modelle werden dazu regelbasiert bezüglich Konsistenz und Unabhängigkeit untersucht. Sari [Sar19] beschreibt ein Degradationskonzept für ein mehrkanaliges System inklusive einer Kanalschaltung auf Systemebene, deren Analyse jedoch nicht detailliert vorgestellt wird. Der Fokus der Arbeit liegt auf der modellbasierten Beschreibung aus der Sicherheitskonzepte abgeleitet werden. Der

Nachweis des Fail-Operational-Verhaltens wird dabei vorwiegend durch die Betrachtung unabhängiger Fehler auf Basis der Architektur berücksichtigt. Eine Generalisierung und die Bewertung der einzelnen Unabhängigkeiten werden nicht thematisiert. [Sar19]

Oszwald [Osz21] präsentiert verschiedene Rekonfigurationskonzepte auf Bauteil-, Komponenten- und Systemebene vorgestellt. Konzepte auf Systemebene basieren dabei vor allem auf zentralen Ansätzen. Detailliert wird im Rahmen des AutoKonf-Projektes ein Konzept, dass auf der Mehrfachverwendung von aktornahen ECUs, in dem Fall dem Brems- und dem Lenksystem, basiert. Dazu werden auf Softwareebene verschiedene Schichten inklusive einer Rekonfigurationsschicht, die bei Diagnose eines Fehlers das redundante Systeme aktiviert, vorgestellt. Die Rekonfiguration kann dabei sowohl dezentral als auch zentral erfolgen um das Fail-Operational Verhalten darzustellen. Anforderungen, wie zeitliche Latenzen unterhalb der Fehlertoleranzzeit, werden adressiert und, ebenso wie die Ausfallwahrscheinlichkeit, mittels Simulation untersucht. Eine hinreichende Sicherheitsargumentation erfolgt im Rahmen der Arbeit nicht, wengleich Aspekte der Sicherheitsbetrachtung relevant für die Konzeptphase berücksichtigt werden. [Osz21]

Kölbl und Leue [KL18] beschreiben die Notwendigkeit einer Fail-Operational-Architektur für das automatisierte Fahren. Basierend auf der Beschreibung des Verhaltens und der ISO 26262 stellen die Autoren Sicherheitsziele für ein Fail-Operational-System vor. Diese berücksichtigten die Diagnose, die Umschaltung und den Rückfall- beziehungsweise Notbetrieb. Nach einem Fehler muss der Rückfallbetrieb erreicht werden und für eine bestimmte Zeit verfügbar sein. Darüber hinaus erstellen die Autoren ein funktionales Modell basierend auf Zuständen für das Fahrzeugsystem. [KL18] Dessen Analyse erfolgt mittels Model-Checking und wird in Kapitel 3.3.1 vorgestellt. Weitere Publikationen umfassen Einzelaspekte von Fail-Operational-Systemen, die sich vorwiegend auf die Hardware oder Softwareebene beziehen [Sch16]. Konzepte für Fail-Operational-Komponenten werden von Sinha [Sin11], Hayama et al. [HHK+10] und Ebner et al. [EGZ20] vorgestellt.

Sinha stellt basierend auf einer einfachen Gefahren- und Risikoanalyse und einem daraus identifizierten ASIL D Sicherheitsziel, eine Architektur eines Brake-by-Wire Systems, bestehend aus zwei Fail-Silent Kanälen, vor. Der Nachweis der Zuverlässigkeit wird mit einer Fehlerbaumanalyse geführt. Außerdem wird die Unabhängigkeit der Pfade und die Degradation auf Systemebene thematisiert. Da das System nicht im Kontext einer Fail-Operational-Fahrzeugführung betrachtet wird, sondern das Lenksystem als solches aufgrund der fehlenden mechanischen Rückfallebene Fail-Operational-Verhalten aufweisen muss, sind die Anforderungen vergleichbarer mit denen eines Fail-Operational-Fahrzeugsystems. Ein Steer-by-Wire System wird von Hayama et al. [HHK+10] präsentiert. Von einer Zustands-basierten Betrachtung der Fehlermodi wird eine redundante Architektur abgeleitet. Nachweise in Form von Sicherheitsanalysen werden nicht geführt. Ebner et al. stellen eine Methodik zur Analyse der Zuverlässigkeit verschiedener Antriebstopologien vor. Durch stochastische Petri-Netze werden Fehler- und Degradationsverhalten modelliert. Eine Auswertung im Rahmen einer Fallstudie hat dabei gezeigt, dass eine redundante Auslegung der Antriebs-einheit inklusive der Kupplungen zu einer höheren Zuverlässigkeit führt und sich entsprechend für Fail-Operational-Anwendungen eignet. Kohn et al. [KKS+06] stellen einen redundanten Aufbau eines Steuergerätes für Fail-Operational-Anwendungen dar. Dabei wird basierend auf funktionalen Aspekten die Fehlertoleranz betrachtet. Sicherheitsanalysen werden nicht betrachtet [KKS+06]. Alcaide Portet et al. [AKHA20] untersuchen Redundanzen auf Software Ebene. Analog zu der Arbeit von Kohn et al. [KKS+06] werden individuelle Kerne verwendet. Deren Diversität wird durch einen Versatz der Rechenzyklen erzeugt. Eine Implementierung zeigt einen geringen Anstieg der Rechenzeit im Vergleich zu einer zweifachen Redundanz durch notwendige Vergleiche. Limitierungen einer reinen Software-basierten Diversität stellen gemeinsam genutzte Hardware und kommunale Elemente, wie der Scheduler dar. Software-Diversität kann demnach als zusätzliche Maßnahme für eine Unabhängigkeit in Fail-Operational-Systemen genutzt werden.

Becker et al. [BS15] und Schleiss et al. [SDWB17] gehen jeweils von fle-

xiblen Fahrzeugplattformen aus, auf denen im Fehlerfall eine funktionsfähige Systemkonfiguration aktiviert wird. Beide Arbeiten stellen Algorithmen vor, die verschiedene sicherheitsrelevante Randbedingungen berücksichtigen. [BS15, SDWB17] Eine solche Architektur stellt in der Serienfertigung jedoch unter Umständen nicht das Kostenoptimum dar. Für große Stückzahlen ist der Einsatz bedarfsgerechter Systeme, wie beispielsweise von Kohn et al. [KKS+06] oder Niedballa und Reuss [NR20] beschrieben, wirtschaftlicher. In der vorliegenden Arbeit werden daher solche Ansätze nicht betrachtet.

Relevante Aspekte der Sicherheit von Fail-Operational-Systemen wurden in der Literatur identifiziert. Diese sind insbesondere die Verfügbarkeit und die Degradation inklusive eines spezifizierten Rückfallbetriebs. Sicherheitsanalysen adressieren vor allem die Ausfallwahrscheinlichkeit und konzeptuelle Aspekte, wie die Degradation und die Notfallbetriebsdauer. Die Umschaltlogik wurde nur von Sari [Sar19] berücksichtigt, der modellbasierte Ansatz stellt aber eine Einschränkung bezüglich der Generalisierbarkeit dar, da eine entsprechende Datengrundlage erforderlich ist.

### 3.1.2 Sicherheitsargumentationen mittels Goal Structuring Notation gemäß ISO 26262

Wie in Kapitel 2.2 beschrieben, fordert der Industriestandard ISO 26262 einen Safety Case, in dem auf Basis der Sicherheitsnachweise eine Argumentation hinreichender funktionaler Sicherheit erfolgt. Entsprechende, hinreichende Sicherheitsargumentation für Fail-Operational-Systeme mittels der Goal Structuring Notation wurden in der Literatur bisher nicht behandelt (vgl. 3.1.1). Im Folgenden werden Arbeiten vorgestellt, die eine Sicherheitsargumentation unter Berücksichtigung der ISO 26262 adressieren. Die Untersuchung sicherheitskritischer Systeme im Kontext anderer Normen oder Domänen, beispielsweise der Luft und Raumfahrt (vgl. [Nen07]), ist aufgrund der divergenten Randbedingungen nicht relevant.

Ridderhof et al. [RGD07] beschreiben ein Vorgehen zur Erstellung einer Sicherheitsargumentation unter Verwendung der Goal Structuring Notation als Basis für einen Safety Cases. Dabei werden ausgehend von einem Ziel auf oberster Ebene durch Verfeinerung Unterziele formuliert. Das erfolgt iterativ, bis zu einem Ziel, das vollständig nachgewiesen wird. Im nächsten Schritt werden die ausgehend davon notwendige Nachweise identifiziert. Die Rückverfolgbarkeit der Ziele wird durch eine Zuordnung zu Sicherheitszielen und Anforderung dargestellt. [RGD07]

Ähnlich gehen auch Birch et al. [BRH+13] vor. Die Detaillierung der Ziele in der Argumentation erfolgt dabei auf Basis möglicher Fehlerursachen, analog zu einem Fehlerbaum, wobei auch die Implementierung und die Validierung berücksichtigt wird. [BRH+13] Die ISO 26262 wird sowohl in [RGD07] als auch [BRH+13] berücksichtigt.

Beckers et al. [BCF+14] verwenden in ihrem Ansatz darüber hinaus zusätzliche Elemente der Goal Structuring Notation, wie eine Unterscheidung der Betrachtungsebene für Ziele, um die Nachvollziehbarkeit zu gewährleisten. Der Fokus der Arbeit liegt auf der Ableitung von Sicherheitsanforderungen und deren Beschreibung durch relevante Attribute. Für die Auswahl der Attribute wurden relevante Aspekte der ISO 26262, wie der Rückfall- oder Notfallbetrieb oder Dekompositionen, identifiziert. [BCF+14]

Habli et al. [HIRK10] nutzen zur Erstellung der Goal Structuring Notation SysML. Für die Detaillierung und Argumentation sind dabei insbesondere Verhaltensdiagramme relevant. Strukturdiagramme dienen der Definition des Kontexts. Eine Fehleranalyse wird nicht behandelt. [HIRK10]

Mit Mustern der Sicherheitsargumentation beschäftigen sich die Arbeiten von Wagner et al. [WSPK10] und Palin [PH10]. Wagner et al. haben solche mittels einer Studie im Industrieumfeld identifiziert. Unter anderem wurden die Verwendung von Fehlermodellen, logische Vereinfachungen, und die Aufteilung gemäß der Architektur als Muster für die Detaillierung von Zielen im Rahmen der Goal Structuring Notation identifiziert. Insbesondere die Verwendung der Fehlermodelle deckt sich dabei mit der bereits vorgestellten Literatur (vgl. [BRH+13, RGD07]). [WSPK10] Palin

et al. [PH10] stellen Muster für die Argumentation auf Fahrzeugebene vor. Dabei werden Randbedingungen, wie rechtliche Aspekte, Produktion und Betrieb, sowie die Systemsicherheit berücksichtigt. Bei der Verfeinerung wird zwischen physischen, wie dem Verbauort, und funktionalen Zielen unterschieden. Außerdem werden Unterziele für Gefahren und Fehler in das Eliminieren der Ursache, Vermeidung und Vermindern aufgeteilt. Das Vermeiden (engl. Mitigation) wird dabei in Diagnose und Degradation unterteilt. [PH10] Die Muster dienen zum einen zur Ableitung der Argumentation und zum anderen zur Prüfung der Vollständigkeit.

Die Anwendung der Sicherheitsargumentation auf Softwaresysteme wurde von Braun et al. [BPSW09] und Hocking et al. [HKAS14] untersucht. Braun et al. identifizieren dabei zwei Hauptargumente für Sicherheit. Zum einen müssen Gefährdungen durch das Sicherheitskonzept und die Analysen adressiert werden. Ebenso ist es notwendig, mögliche Fehler des Systems zu betrachten und Maßnahmen abzuleiten. Eine individuelle Betrachtung der Software ist nicht hinreichend, stattdessen müssen Umweltbedingungen und Hardwareaspekte in der Analyse berücksichtigt werden. Funktionale Modelle dienen dabei der Ableitung der Argumentation. [BPSW09] Hocking et al. [HKAS14] identifizieren die Argumente analog. Demnach müssen die Anforderungen im Kontext der Sicherheitskonzepte umgesetzt und Fehler verhindert werden. Darüber hinaus sind auch formale Richtlinien und Standards einzuhalten. [HKAS14]

Ein Vorgehen basierend auf iterativen Detaillierungen der Ziele, bis zu einer Ebene, für die ein Nachweis erfolgt und anschließender Identifikation notwendiger Nachweise wird in allen Arbeiten vorgeschlagen. Zur Verfeinerung werden insbesondere funktionale Modelle, Verhaltensmodelle und Fehlermodelle genutzt. Einige Arbeiten stellen darüber hinaus eine Erfüllung der ISO 26262 durch die Identifikation relevanter Attribute, wie Fehlertoleranz, und deren Berücksichtigung in der Sicherheitsargumentation dar.

## 3.2 Analyse gemeinsamer Ausfälle

Fehlertoleranz durch redundante Systeme und Dekompositionen, die Aufteilung von Sicherheitslasten auf mehrere Elemente, setzen Unabhängigkeit voraus [Sch16]. Die ISO 26262, wie auch die IEC 61508, fordern daher eine Analyse abhängiger Fehler (engl. Dependent Failure Analysis (DFA)) [Deu01, Int18]. Die Analyse dient dem Nachweis der Unabhängigkeit bezüglich gemeinsamer Ausfälle (engl. common-cause failures) sowie kaskadierenden Fehlern (engl. cascading failures) und dem Zweck Sicherheitsmaßnahmen zu definieren und zu bewerten [Int18]. Im Folgenden steht die Analyse gemeinsamer Ausfälle im Fokus (engl. Common-Cause Analysis (CCA)).

Die Analyse gemeinsamer Ausfälle wurde in der Literatur über einen langen Zeitraum und über verschiedene Industrien hinweg untersucht. Im Folgenden erfolgt die Beschreibung eines generischen Vorgehens, wie es in der Literatur aufgezeigt wird. Der Fokus liegt dabei auf qualitativen Analysen. Im Anschluss werden die einzelnen Arbeitsschritte detailliert und in diesem Zusammenhang relevante Literatur beschrieben. Dabei werden wiederum insbesondere Arbeiten mit Bezug zur Automobilindustrie und im Kontext der ISO 26262 berücksichtigt und damit der Stand der Wissenschaft aufgezeigt.

Das generische Vorgehen der Analyse gemeinsamer Ausfälle wird industrieübergreifend analog beschrieben. Relevante Arbeiten stammen aus Industrien mit sicherheitskritischen Aspekten, wie der Nukleartechnik, der Luft und Raumfahrt, der Medizintechnik und der Automobilindustrie. Die grundlegenden Schritte einer Analyse werden sowohl in der ISO 26262 [Int18] als auch den Publikationen von Ericson [Eri16], Johnston [Joh87], Balakrishnan [Bal15] und Schnellbach [Sch16] beschrieben.

Das Vorgehen umfasst dabei folgende Schritte:

1. Analyse der Systemlogik,
2. Identifikation der notwendigen Unabhängigkeit,
3. Identifikation der Abhängigkeiten und Koppelfaktoren,
4. Bewertung der Abhängigkeiten,
5. Ableitung von Maßnahmen und
6. Dokumentation und Review der Analyse.

Die einzelnen Schritte der Analyse und der jeweilige Stand der Wissenschaft wird im Folgenden, unter Berücksichtigung der Vorgaben der ISO 26262, anhand relevanter Veröffentlichungen aufgezeigt.

### **Analyse der Systemlogik und Identifikation der notwendigen Unabhängigkeiten**

Für die Analyse der Systemlogik und der Identifikation der notwendigen Unabhängigkeit empfiehlt die ISO 26262 Sicherheitsanalysen [Int18]. Deduktive Analysen, wie die FTA eignen sich, da diese Mehrfachfehler berücksichtigen und somit Anforderungen der Unabhängigkeit aufzeigen. [Bal15, Eri16] Nach Schnellbach [Sch16] ist die Berücksichtigung von Redundanzen, inklusive Dekompositionen, Sicherheitsmechanismen und Umfängen unterschiedlicher Integrität, für einfache Systeme hinreichend. Komplexe Umschalt Szenarien und Rekonfigurationen nach Fehlern wurden dabei nicht berücksichtigt.

### **Identifikation einzelner Koppelfaktoren, deren Bewertung und die Ableitung von Maßnahmen**

Im nächsten Schritt erfolgt die Identifikation einzelner Koppelfaktoren (engl. coupling factors), deren Bewertung und die Ableitung von Maßnahmen. In der ISO 26262 wird gefordert, dass in der Analyse folgende Themenbereiche abgedeckt werden:



- Zufällige Hardwarefehler,
- Fehler in der Entwicklung,
- Fertigungsfehler,
- Montagefehler,
- Wartungsfehler,
- Umweltfaktoren,
- Fehler gemeinsamer externer Ressourcen und
- Belastung und Alterung.

Schnellbach [Sch16], Moestl und Ernst [ME15] schlagen eine getrennte Analyse von Hardware-, Funktions- und Software-Umfängen vor. Die Identifikation und Bewertung kann auf Basis von Sicherheitsanalysen, wie einer FMEA oder FTA, oder mittels Checklisten durchgeführt werden. In der Literatur werden darüber hinaus modellbasierte Vorgehen vorgeschlagen. Checklisten werden am häufigsten genannt, da diese auch im Falle großer komplexer Systeme praktikabel sind und werden und im Folgenden adressiert. Sowohl die IEC 61508 und die ISO 26262, als auch die Arbeiten von Ericson [Eri16], Mosleh et al. [MRDM99], Leeman et al. [LDC09], Moestl und Ernst [ME15], sowie, im Rahmen einer Common Mode Analyse, einer Analyse gleicher Fehlermodi, Wang [Wan17] und Balakrishnan [Bal15] geben Beispiele solcher Checklisten. Schnellbach [Sch16] konsolidierte einige dieser Checklisten. Dabei ergeben sich gegenüber der ISO 26262 Detaillierungen bezüglich der Entwicklungsfehler. Unter anderem die Anforderungsspezifikation, aber auch identische Systeme, der Verbauort, Softwarefehler und die unerwünschte Beeinflussung durch Daten und Timing wurden zusätzlich als Quellen für Fehler in der Entwicklung identifiziert, wobei der Detaillierungsgrad gegenüber ISO 26262 nur geringfügig steigt [Sch16]. Auf Basis der Checklisten erfolgt die Analyse der Unabhängigkeit, respektive der Systeme. Kriterien für die Bewertung gemäß der ISO 2626 sind dabei die Integrität, der Grad der Unabhängigkeit, die Komplexität, die Technologie und die Belastung des Systems. Der Detaillierungsgrad muss dabei ausreichend für eine Argumentation der

Sicherheit sein. [Int18] Sofern nach Analyse oder Experteneinschätzung keine ausreichende Unabhängigkeit gegeben ist, erfolgt die Definition von Maßnahmen, wie die Eliminierung und Kontrolle der Fehlerursache oder die Minderung der Koppelmechanismen. [Sch16] Ein analoges qualitatives Vorgehen wird in der IEC 61508 für die Analyse von Industrieanlagen beschrieben [Deu01, MRDM99]. In der Luftfahrt und Medizintechnik wird die Common Mode Analysis um die Particular Risk Analyse und die Zonal Safety Analyse ergänzt [SAE96, Wan17]. Erstere berücksichtigt dabei externe Fehlerursachen, die basierend auf Erfahrungswerten als kritisch angesehen werden. Die Zonal Safety Analyse adressiert Gefahren unter Berücksichtigung des Verbauorts und dessen Umgebung [SAE96]. Schnellbach kritisiert die Anwendbarkeit der Checklisten aufgrund der allgemeinen Formulierung und der daraus bedingten fehlenden Struktur [Sch16]. Daher wird in [Sch16] und übernommen in den Anhang der ISO 26262 Band 9 [Int18] die Anwendung eines Abhängigkeitsmodells vorgeschlagen. Dieses wird in Abbildung 3.1 gezeigt.

Schnellbach schlägt eine Untersuchung anhand des Modells zur Identifikation von Koppelmechanismen (engl. Common-Cause Initiator (CCI)) und Abhängigkeiten vor. Darin berücksichtigte Koppelmechanismen sind beispielsweise geteilte Ressourcen, wie die Energieversorgung. Im Anschluss werden die identifizierten Abhängigkeiten mittels einer Checkliste weiter untersucht. Beispiele der Checkliste für verschiedene Betrachtungsebenen (System-, Hardware- und Softwareebene) werden dabei sowohl in [Sch16] als auch der ISO 26262 [Int18] gegeben. Auf Systemebene stellt beispielsweise die Energieversorgung eine geteilte Ressource dar, auf Softwareebene können Bibliotheken ein Initiator gemeinsamer Ausfälle sein. Schnellbach [Sch16] weist daraufhin, dass ein Katalog aufgrund der Domänenabhängigkeit nicht allgemein anwendbar ist. Das Abhängigkeitsmodell stellt damit eine Ergänzung eines rein Checklisten-basierten Verfahrens dar und ermöglicht damit im Vergleich ein systemspezifisches und zielgerichteteres Vorgehen. Außerdem erfolgt durch die Identifikation der Abhängigkeiten eine Prüfung der Vollständigkeit. Die Untersuchung der Unabhängigkeit auf Software- und Halbleiterebene adressieren Moestl und Ernst [ME15],

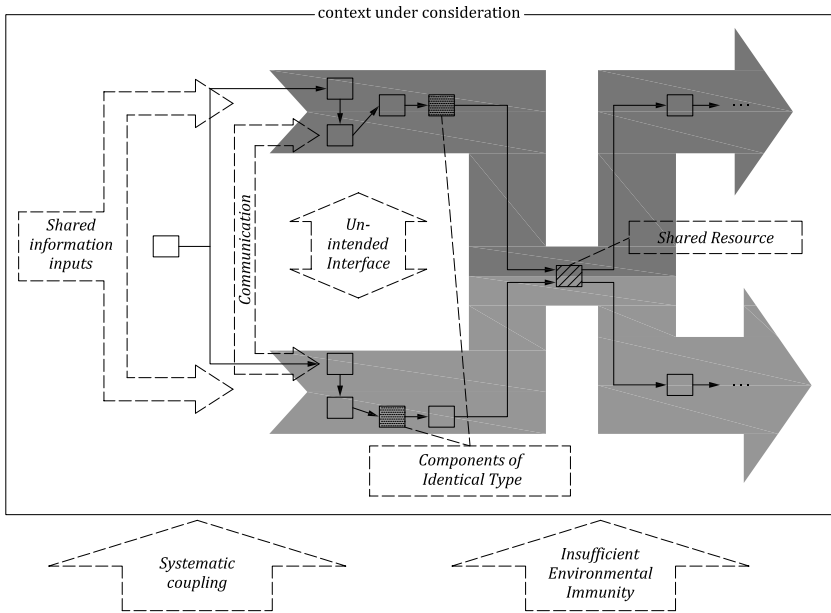


Abbildung 3.1: Abhängigkeitsmodell nach Schnellbach [Sch16] (vgl. [Int18])

sowie Prashant-Reddy und Leburu [PL20] mittels einer Erweiterung der Checklisten zu einer Matrix-Darstellung. Das Vorgehen ist insbesondere für komplexe Zusammenhänge und zum Nachweis der Unabhängigkeit innerhalb eines Elements geeignet [PL20]. Wie zu Beginn des Abschnitts erklärt, wird in der Literatur, von Ericson [Eri16] und Balakrishnan [Bal15], neben der Verwendung von Checklisten für prozessuale Fehler auch ein Vorgehen auf Basis von Sicherheitsanalysen, wie der FTA und FMEA, beschrieben. Dabei werden Fehlerursachen identifiziert. Durch deren Abgleich erfolgt die Identifikation von Common-Cause Ausfällen. Allerdings erfordert dies eine identische Beschreibung und Notation der analysierten Fehlerfälle. Aufgrund der hohen Anzahl an Fehlerursachen und notwendigen Unabhängigkeit in einem komplexen System ist eine Analyse basierend auf Sicherheitsanalysen

umfangreich [Eri16]. Die ISO 26262 empfiehlt daher nur einen unterstützenden Einsatz von Sicherheitsanalysen. Weitere Analyseverfahren stellen modellbasierte Verfahren, wie beispielsweise von Sari [Sar19] und Dropmann et al. [DTT+18] präsentiert, dar. Diese basieren auf der Modellierung der Systemzusammenhänge inklusive deren Abhängigkeiten. Die Analyse erfolgt durch eine regelbasierte Überprüfung der Modelle. [Sar19] Solche Verfahren erfordern jedoch eine entsprechende modellbasierte Entwicklung, um effizient eingesetzt werden zu können.

### **Finalisierung der Dokumentation**

Im letzten Schritt der Analyse erfolgt die Dokumentation. Anforderungen an diese ergeben sich aus der ISO 26262 [Int18]. Der Industriestandard gibt dabei kein explizites Format vor, allerdings soll die Dokumentation präzise, klar strukturiert, verständlich und wartbar sein [Int18]. Balakrishan [Bal15] erklärt, entsprechend des Vorgehens basierend auf Sicherheitsanalysen, dass eine Dokumentation analog zu der einer FMEA erfolgen kann. Diese muss Ziel und Zweck der Analyse, die Beschreibung der Vorgehensweise und deren Grenzen beinhalten. Außerdem müssen die Ergebnisse dokumentiert sein und referenzierte Analysen und Dokumente zugänglich sein. [Bal15]

Der Stand der Wissenschaft zeigt, dass das generische Vorgehen Industrieübergreifend analog beschrieben wird. Die Umsetzung der einzelnen Schritte dagegen variiert. Die Systemanalyse kann, wenn die Komplexität es erfordert, über Sicherheitsanalysen erfolgen. Bezüglich der Analyse der Unabhängigkeit werden insbesondere Checklisten-basierte Vorgehen vorgeschlagen. Das Abhängigkeitsmodell nach Schnellbach [Sch16] stellt dabei einen ergänzenden systematischen Ansatz dar. Kataloge für Initiatoren sind domänenspezifisch, weswegen in der Literatur nur Beispiele gegeben werden. Erfahrungswerte oder Felddaten bezüglich Ausfällen in der Automobilindustrie werden für die Analyse gemeinsamer Ausfälle gemäß der Literatur nicht berücksichtigt. Ein Prozess zur Erstellung oder Wartung solcher Listen wird nicht vorgestellt. Die präsentierte Literatur beschränkt die Untersuchungen auf klar abgegrenzte Systeme, eine Analyse von kompletten Wirkketten

wird nicht adressiert. Auf zusätzliche Randbedingungen, wie eine verteilte Entwicklung und Effizienz, wird in der Literatur ebenfalls nicht eingegangen.

### 3.3 Formale Verifikation mittels Model-Checking

Formale Methoden werden in der Literatur ausgiebig behandelt. In der Automobilindustrie und der funktionalen Sicherheit sind diese bisher nicht etabliert; geforscht wurde zu dieser Thematik jedoch. Im Folgenden werden Model-Checking Ansätze im Kontext der ISO 26262 respektive der Automobilindustrie diskutiert. Darüber hinaus wird der Forschungsstand bezüglich der Verifikation sicherheitskritischer Systeme mittels Model-Checking behandelt und anhand relevanter Arbeiten vorgestellt.

#### 3.3.1 Model-Checking in der Automobilindustrie

Einige Arbeiten adressieren die Verwendung formaler Methoden im Kontext der Automobilindustrie. Der Fokus der Literaturrecherche liegt auf der generellen Anwendbarkeit, um Vorteile, Probleme und Erkenntnisse bezüglich der Implementierung zu identifizieren. Es werden Arbeiten betrachtet, die sich mit der Verifikation des funktionalen Verhaltens beschäftigen. Die Verwendung spezifischer Werkzeuge wird in diesem Kapitel nicht diskutiert.

Die Arbeit von Kölbl und Leue [KL18] wurde in Kapitel 3.1.1 bereits eingeführt. Basierend auf einer funktionalen Beschreibung mit SysML Zustandsdiagrammen wird das Systemverhalten gegen die formal spezifizierten Sicherheitsziele verifiziert. Dies beinhaltet Fail-Operational-Verhalten inklusive eines zeitlich begrenzten Rückfallbetriebs. Die Formulierung der Sicherheitsziele erfolgt dabei durch logische Kombination einzelner Zustände von Komponenten. Durch die Berücksichtigung von Fehlerraten und stochastischem Model-Checking werden die Wahrscheinlichkeiten für Verletzungen von Anforderungen bestimmt und damit grundlegende Architekturentscheidungen abgeleitet, für die Rechenzeit und Speicherbedarf beim Model-Checking handhabbar sind [KL18].

Die Anwendbarkeit formaler Methoden haben Nyberg et al. [NGL+18] im Rahmen der Entwicklung von Lastkraftwagen bei Scania<sup>1</sup> anhand verschiedener Studien untersucht. Hierfür haben die Autoren unter anderem ein einfaches Mathworks, Simulink<sup>2</sup> Modell mittels Model-Checking verifiziert. Eine detaillierte Implementierung wird nicht beschrieben. Nyberg et al. [NGL+18] kommen zu dem Schluss, dass formale Methoden eine effiziente Methode der Verifikation in bestimmten Anwendungsfällen, abhängig vom notwendigen Abstraktionsgrad, sind. Die methodische Komplexität und die Diversität der Architekturen und der Software erschweren die Anwendung in der Automobilindustrie jedoch. [NGL+18]

Auch Nellen et al. [NRW+18] untersuchen die Anwendung des Model-Checkings für die Verifikation einer Fahrzeugfunktion, in diesem Fall einer Parkfunktion, in Mathworks, Simulink. Dabei wurden Verletzungen der Spezifikationen entdeckt, jedoch konnten aufgrund der Laufzeiten und Nicht-linearitäten nicht alle Anforderungen verifiziert werden. Einfluss auf die Laufzeit hatten dabei variierende Parameterwerte und die Größe des Modells sowie das relevante Analyseintervall. Die Autoren sehen in der formalen Verifikation einen Mehrwert und eine einfache und effiziente Analysemöglichkeit und schlagen die Verwendung ergänzend zu Sicherheitsanalysen vor. Zusätzlich zu den bisher vorgestellten Arbeiten wird von Nellen et al. [NRW+18] eine Qualitätssteigerung der Anforderungen durch die formale Spezifikation, da Inkonsistenzen und Unvollständigkeiten erkannt werden, positiv bewertet. In der Arbeit wird außerdem die Erstellung und Rückverfolgbarkeit von Anforderungen adressiert. [NRW+18]

Todorov et al. [TBT18] beschäftigen sich mit der Anwendung formaler Methoden zur Beherrschung der Komplexität automatisierter Systeme. Für eine Geschwindigkeitsregelung wurde ein nicht näher beschriebenes Modell und formale Spezifikationen formuliert. Beim inkrementellem Model-Checking, bei dem bereits erbrachte Nachweise für Weitere genutzt werden, ergaben sich lange Rechenzeiten für Anforderungen, die lange Zeitintervalle umfassen. Bezüglich der Anwendung wurden ähnliche Herausforderungen wie

---

<sup>1</sup>Original Equipment Manufacturer (OEM) für Nutzfahrzeuge (<https://www.scania.com/>)

<sup>2</sup><https://de.mathworks.com/products/simulink.html>

von Nyberg et al. identifiziert. Aber auch Todorov et al. sehen durch Korrektheitsnachweise Vorteile in der Verifikation komplexer Systeme mittels formaler Methoden. [TBT18] Auf die ISO 26262 wurde von Nyberg et al. [NGL+18], Todorov et al. [TBT18] und Nellen et al. [NRW+18] anhand der Notwendigkeit einer Analyse eingegangen.

Die Verifikation einer Motorregelung im Schleppfall wurde von Villa et al. [VWTB+98] bereits 1998 untersucht. Für die Anwendung von Model Checking wurde dafür ein Zustandsmodell explizit erstellt, ein Schritt, der durch die Anwendung von mittlerweile gängigen Werkzeugen nicht mehr notwendig ist. Transitionen ergeben sich durch Bestimmung der Freiheitsgrade der Motorschwingungen. Mittels On-the-fly Model-Checking wurde das System verifiziert. Villa et al. weisen allerdings bereits auf die Notwendigkeit von Abstraktionen zur Vermeidung der Zustandsexplosionen und der Minimierung des Speicherbedarfes hin. [VWTB+98]

Der Ansatz von Kim et al. [KLN+15] nutzt die Kombination von symbolischem und statistischem Model-Checking. Letzteres dient der Auswahl einer Menge an zu prüfenden Zustandssequenzen. Mit der Methode konnten Kim et al. die effiziente Verifikation eines Blinker-Systems demonstrieren. Die Laufzeit ist dabei insbesondere vom Konfidenzlevel abhängig, das den Umfang des statistischen Model-Checkings beeinflusst. Auch Kim et al. weisen auf die Herausforderung der formalen Spezifikation hin. [KLN+15]

Aniculaesei und Zhang [AVZR21] nutzen eine Kombinatorik von Model-Checking und datenbasierter Modellierung für die Verifikation sicherheitskritischer Funktionen. Sicherheitsanforderungen werden formalisiert. Die zu verifizierenden Funktionen werden auf Basis von Messdaten als Ein-Ausgangsfunktionen abstrahiert beschrieben. Zudem erfolgt die Definition von Grenzwerten durch physikalische Zusammenhänge um das Prüfen nicht-linearer Zusammenhänge zu ermöglichen. Durch modulare Beschreibungen der Funktionen und Teilfunktionen ist der Ansatz skalierbar und bietet daher insbesondere bei komplexen Funktionen, eine zusätzliche, von der ISO 26262 geforderte, formale Verifikationsmöglichkeit. Dies wurde im Rahmen von Fallbeispielen, wie einer ACC-Funktion (Adaptive Cruise Control), bei einem OEM, gezeigt. Für den Ansatz sind allerdings Messdaten notwendig,

was insbesondere in frühen Entwicklungsphasen unter Umständen nicht gewährleistet ist [AVZR21]

Das Sicherheitskonzept eines Elektrofahrzeuges wurde in [BDP+17] vergleichend mit mehreren formalen Methoden verifiziert. Im Gegensatz zu statischen Methoden und Testen konnten mittels Model-Checking alle Fehlerinjektionen entdeckt werden. Die Autoren schließen daher auf das Potential von Model-Checking zur Kostenreduktion und Effizienzsteigerung durch eine frühzeitige vollständige Verifikation. Analog zu [NRW+18] wurde die Steigerung der Anforderungsqualität hervorgehoben. [BDP+17]

Bahig et al. [BEK17] zeigen eine semi-formale Verifikation der Automotive Open System Architecture (Autosar)<sup>1</sup> Watchdog-Komponente. Für dessen Verifikation wurden die Spezifikationen formal beschrieben und das Verhalten mittels UML Zustandsdarstellung modelliert. Geprüft wurden dabei neben Spezifikationsgrenzen auch Zustände, die Sackgassen darstellen. Die Prüfung beinhaltet allerdings keine Betrachtung von Sequenzen in Bedingungen, weswegen kein Korrektheitsnachweis erfolgt, bietet aber eine Unterstützung bei Entwicklung. [BEK17]

Neben der funktionalen Systemanalyse und der Softwareanalyse wird Model-Checking auch zur Verifikation von Hardwareaspekte genutzt. Beispielhaft wird die Arbeit von Beyer et al. [BBG+05] vorgestellt, in der die Verifikation einer Kommunikationsschicht adressiert wird. Dabei wird das Zeitverhalten der Kommunikation inklusive Latenzen und Phasenversatz formal beschrieben und verifiziert. Der Algorithmus basiert auf einer Referenzzeit. Die Arbeit zeigt die Anwendbarkeit der formalen Beschreibung. [BBG+05]

Model-Checking wird neben der direkten Verifikation auch zur Ableitung von Testfällen vorgeschlagen. Aniculaesei, Vorwald und Rausch [AVR19a, AVR19b] präsentieren in mehreren Arbeiten ein Vorgehen zur automatisierten Testfallgenerierung auf Basis formaler Anforderungen. Beim Model-Checking werden prinzipiell Verletzungen der Spezifikationen gesucht, daher negieren die Autoren die Anforderungen und formulieren sogenannte *trap conditions* zur Generierung von Verletzungen im Sinne des

---

<sup>1</sup>Zusammenschluss von Industriepartner zur Entwicklung einer Standardisierten Software Architektur in der Automobilindustrie (<https://www.autosar.org/>)



Model-Checkings. In [AVR19b] erfolgt die Formulierung der Spezifikationen unter Berücksichtigung verschiedener Metriken, wie auch von Whalen et al. [WRHM06] vorgeschlagen, um das spezifizierte Systemverhalten zu beschreiben. Genutzte Metriken sind eine Anforderungsabdeckung, eine explizite Berücksichtigung der Vorbedingungen und des Endzustands und die separate Analyse jedes Freiheitsgrades. Aus den Traces der identifizierten Verletzungen, den Systemzuständen bei der Prüfung der *trap conditions*, werden die Eingangsparameter für eine Simulation extrahiert. Mit diesen wird der entsprechenden Endzustand bestimmt, wodurch sich ein vollständiger Testfall ergibt. Durch die Analyse einer Logik für eine Zentralverriegelung wird das Vorgehen demonstriert [AVR19a]. In [AVR19b] erfolgt zudem die Validierung des Vorgehens, bei der die genannten Metriken zur Ableitung der Spezifikation aus den Anforderungen genutzt wurden, mittels eines ACC-Systems auf Ebene der Kundenfunktion. [AVR19b] In [AVR19b] und [AVR19a] wird dabei das Werkzeug SCADE <sup>1</sup> verwendet. In einer weiteren Arbeit von Aniculaesei et al. [AHDR18] wurde für das ACC-System, unter Verwendung von NuSMV <sup>2</sup>, die Testfallabdeckung durch die generierten Anforderungen anhand eines Vergleichs mit mutationsbasierter Testfallgenerierung für die genannten Metriken evaluiert. Dabei konnte gezeigt werden, dass die Test-Sets unter Berücksichtigung der jeweiligen Metrik die durch Mutationen generierten Testfälle zu über 90 % abdecken. Auch wenn der Anwendungsfall nicht der Komplexität einer Automobilfunktion auf Softwareebene entspricht, konnte gezeigt werden, dass durch die Formulierung der negierten Spezifikation das Systemverhalten verifiziert werden kann. Unter der Prämisse eines korrekten Modellverhaltens kann das Vorgehen auch zur Validierung der Spezifikationen genutzt werden. [AHDR18, AVR19b]

Publikationen zu formalen Methoden im Automobilkontext zeigen deren grundlegende Anwendbarkeit. Der Modellierungsaufwand stellt keine Limitierung dar, zudem gibt es Ansätze um die Äquivalenz von Code und Modell sicherzustellen (vgl. [TAR20]). Stattdessen wird vor allem die for-

---

<sup>1</sup><https://www.ansys.com/products/embedded-software/ansys-scade-suite>

<sup>2</sup><http://nusmv.fbk.eu/>

male Spezifikation als Herausforderung gesehen. Allerdings wird auch der Mehrwert durch den Korrektheitsnachweis und die Steigerung der Anforderungsqualität hervorgehoben. Limitierungen sind vor allem bei Prüfung von Anforderungen über große Intervalle durch lange Laufzeiten aufgetreten. Weitere genannte Einschränkungen haben sich vor allem auf einzelne Werkzeuge bezogen. Die gezeigten Anwendungsfälle stellen abstrahierte Systeme dar und spiegeln die Komplexität einer Fahrwerksfunktion nicht wieder. Dennoch stellen Laufzeit und Speicherprobleme Hindernisse dar. Optimierungen der Vorgehensweise wurden in den vorgestellten Arbeiten nicht behandelt, solche werden im folgenden Kapitel adressiert. Zudem wird in der Literatur die Ableitung von Testfällen adressiert durch die Negierung von Spezifikation adressiert womit das Systemverhalten oder auch die Spezifikation verifiziert werden können.

### 3.3.2 Optimierungsansätze des Model-Checkings

In der Literatur wurden neben der Anwendung des Model-Checkings verschiedene Optimierungsansätze behandelt, um Laufzeit- und Speicherprobleme zu minimieren. Relevante Arbeiten und Konzepte werden vorgestellt. Dabei werden die Strategien nach Such- und Speicherverfahren gegliedert.

#### 3.3.2.1 Suchverfahren

Relevante Optimierungen und Erweiterungen von Suchstrategien sind Tiefenbegrenzungen (bounded Model-Checking), eingebettete Tiefensuche, Parallelisierung und zielgerichtetes Model-Checking. Die Ansätze werden im Folgenden anhand relevanter Literatur beschrieben.

Die Tiefenbegrenzung (engl. bounded Model-Checking) wurde bereits in Kapitel 2.4.3 eingeführt. Ein solches Vorgehen verhindert, dass Schleifen, die zu einem zyklischen Zurückspringen führen, beliebig oft berechnet werden und stellt so eine Terminierung sicher [BCC+09, Hol08]. Algorithmen zur Transformation in Erfüllbarkeitsprobleme, welche bei einer höheren Anzahl möglicher Variablen effizienter sind als Entscheidungsdiagramme sind, wer-

den beispielsweise von Biere et al. [BCC+09] oder Cimatti et al. [CGP+02] vorgestellt. Udupa et al. [UDR11] stellen eine Erweiterung vor, bei dem die Beschränkung iterativ aufgeweitet wird, wenn innerhalb der Beschränkung kein Gegenbeispiel gefunden wird. Für Systeme, die den Anforderungen vollständig entsprechen, ergibt sich keine Laufzeitverbesserung [Hol08, UDR11]. Eine Erweiterung der Tiefensuche um Lebendigkeit-Eigenschaften, Eigenschaften, die durch infinite Zustandssequenzen erfüllt werden, zu verifizieren, stellt die eingebettete Tiefensuche dar. Courcoubetis et al. [CVWY92] stellen einen Algorithmus vor, bei dem in jedem Zustand geprüft wird, ob eine Schleife zu diesem vorhanden ist, die eine Lebendigkeitsbedingung, eine Bedingung, dass ein bestimmter Zustand erhalten wird, erfüllt. Sofern dies der Fall ist wird die Schleife in der Tiefensuche berücksichtigt. [CVWY92] Eine Optimierung der Breitensuchen wird vor allem angewandt, um die Berechnungen zu parallelisieren. Ein solches Vorgehen wird von Barnat et al. [BBC03] vorgestellt. Dabei werden einzelne Tiefensuchen nach einer Breitensuche Recheneinheiten zugeteilt. Damit kann die Suche parallelisiert und verfügbare Ressourcen können besser genutzt werden. [BBC03] Neben den genannten Optimierungsverfahren wurden in der Literatur zielgerichtete Suchverfahren (engl. directed Model-Checking) vorgestellt. Deren Konzept basiert auf Heuristiken mit dem Ziel relevante Zustände früher zu untersuchen [ELLL04]. Yang und Dill [YD98] stellen verschiedene Ansätze vor. Eine Möglichkeit, um Verletzungen früher zu erkennen, ist ein Abbruch bei Zuständen, die bekannterweise direkt zu einem Zustand führen, der die Spezifikationen verletzt. Die Effektivität des Ansatzes variiert dabei mit dem untersuchten System. [YD98] In Kombination mit einer Priorisierung der Zustände auf Basis der Hamming-Distanz wurden weitere Verbesserungen der Laufzeit erreicht. Eine deutliche Effizienzsteigerung konnte durch die Auswahl einer Untermenge der Zustände erzielt werden. Dies stellt jedoch eine Approximation dar, mit der unter Umständen nicht alle Verletzungen der Systemanforderungen gefunden werden können. Einen vollständigen Nachweis der Korrektheit, kann die Methode dadurch nicht liefern. [YD98] In der Arbeit von Edelkamp et al. [ELLL04] wird eine Kostenfunktion vorgeschlagen, um basierend auf einer Breitensuche den zu

verfeinernden Knoten zu wählen. Wehrle et al. [WKP09] verfolgen einen Ansatz basierend auf Zustandsübergängen. Dabei werden nur Zustände explizit bestimmt, die einen Einfluss auf die Distanz zu einer Verletzung haben. Geänderte Variablen, welche keine Auswirkungen auf den Systemzustand haben, werden beispielsweise nicht betrachtet. [WKP09]

Die Literatur zeigt verschiedene Ansätze der Optimierung auf. Begrenzungen der Suchtiefe und Parallelisierungen werden allgemein vorgeschlagen, sofern der Anwendungsfall dies ermöglicht. Die Effizienz weiterer Optimierungen, wie zielgerichtete Suchen, variieren dabei mit dem System und sind insbesondere bei schneller Detektion von Verletzungen hilfreich. Für die Verifikation der Korrektheit bieten diese im Allgemeinen keine Effizienzsteigerung.

### 3.3.2.2 Speicheroptimierung

Weitere Optimierungsstrategien, die in der Literatur adressiert werden, betreffen den Speicherbedarf. An dieser Stelle werden das On-the-Fly Model-Checking, Caching und Kompressions-Strategien vorgestellt.

On-the-Fly Model-Checking beschreibt ein Verfahren, bei dem der Zustandsraum zur Laufzeit bestimmt wird [BK08, CHVB18]. McMillan [McM93] und Holzmann et al. [Hol08] stellen Algorithmen vor, bei dem erst im jeweiligen Zustand die Folgezustände bestimmt werden. Dadurch wird der Speicherbedarf minimiert, da im Falle einer Anforderungsverletzung nicht der komplette Zustandsraum bestimmt wird. [Hol08] Wiederum ergibt sich keine Minimierung des Speicherbedarfs bei einem korrekten System, da besuchte Zustände, um mehrmaliges Prüfen zu verhindern, nach wie vor abgelegt werden.

Beim Caching werden Zustände gespeichert, bis die Speicherkapazität erreicht ist. Sofern dies eintritt, werden Zustände aus dem Speicher durch neue ersetzt. Verschiedene Ersetzungsstrategien, insbesondere die Auswahl auf Basis von Abdeckungsgraden, werden von Behrmann et al. [BLP03] vorgestellt.

Kompressionen der Zustände minimieren den Speicherbedarf. Holzmann et al. [Hol08] stellen ein Hashing-Verfahren vor. Die Ersetzung der Zustände ist dabei allerdings nicht eindeutig und kann zu Kollisionen führen. Pelánek et al. [PRŠ08] stellen weitere Verfahren der Codierung vor. Darüber hinaus weisen genannte Arbeiten darauf hin, nur minimal notwendige Zustandsinformationen zu speichern. [BK08, CHVB18, WKP09]

Die Literatur zeigt diverse Ansätze der Optimierung des Speicherbedarfes auf. Eine Kombination dieser ist mitunter möglich.

### 3.4 Analyse der Fehlerpropagation

Zur Untersuchung der Fehlerpropagation werden neben klassischen Analysemethoden ein breites Feld an weiteren Ansätzen in der Wissenschaft vorgestellt. Unabhängig von der Methode ist eine Abbildung des Systems erforderlich. Die zu Grunde liegenden Modell- und Graphen-basierten Ansätze wurden in Kapitel 2.4.2 eingeführt. Relevant für diese Arbeit sind dabei insbesondere modulare Fehlerbäume, Model-Checking Ansätze und simulationsbasierte Methoden. Der Fokus liegt dabei auf der Abbildung des Fehlerverhaltens und des Modellierungsumfangs.

Klassische Methoden (vgl. Kapitel 2.4.1) stoßen bei der Analyse komplexer Systeme an ihre Grenzen, daher werden in der Literatur Erweiterungen dieser vorgeschlagen.

Kaiser [Kai06] stellt in seiner Arbeit eine Erweiterung klassischer Fehlerbäume in Form von modularen Fehlerbäumen vor. Das Konzept basiert auf der modularen Darstellung einzelner Einheiten, beispielsweise Komponenten, und deren Verknüpfung mit Hauptereignissen und anderen Elementen. [Kai06] Das ermöglicht die Abbildung der Architektur und eine einfache Anpassung der Fehlerlogik bei lokalen Änderungen. In herkömmlichen Fehlerbäumen ergeben sich insbesondere durch zyklische Berechnungen und Rückwirkungen lange Fehlerketten. Diese können in modularen Fehlerbäumen übersichtlich dargestellt werden. Die Fehlerlogik basiert auf der

booleschen Logik. Die Berechnung der modularen Fehlermodelle erfolgt durch binäre Entscheidungsdiagramme. [Kai06] Kaiser et al. [KSA+18] erweitern das Konzept in einer späteren Arbeit um Sequenzen, abgebildet durch Markov-Ketten, und Möglichkeiten der automatischen Generierung. [KSA+18] Die Integration quantitativer Betrachtungen erfolgt durch Roth et al. [RL15].

Zustands/Event Fehlerbäume stellen eine zusätzliche Erweiterung der modularen Fehlerbäume dar. Kaiser [Kai06] integriert Zustandsautomaten deren Übergänge durch Events getriggert werden, die analog der Fehlerlogik im Modell verknüpft werden. Um zeitliche Aspekte zu modellieren werden darüber hinaus Verzögerungselemente eingeführt. [Kai06] Zustand/Event-Fehlerbäume werden simuliert. Die Rechenzeit kann dabei die Anwendbarkeit einschränken. [Kai06] Jiang et al. [JZW19] stellen einen Ansatz vor, bei dem die Zustands-/Event Fehlerbäume in zeitgesteuerte Automaten umgewandelt und mittels Model-Checking analysiert werden. Der Ansatz bietet die Möglichkeit einer effizienten Verifikation, beschränkt sich aber auf qualitative Untersuchungen. [JZW19]

Darüber hinaus werden in der Literatur formale Ansätze unter der Verwendung von Model-Checking diskutiert, um Fehlerpropagationen zu untersuchen.

Albore et al. [ADI+17] stellen einen Ansatz basierend auf der formalen Modellierungssprache AltaRica<sup>1</sup> vor. Dabei werden Funktionen durch Zustände und Transitionen definiert und erweitern die Spezifikation um zeitliche Bedingungen. Die Autoren berücksichtigen dabei Zeitintervalle für die Detektion, Fortpflanzung und Auswirkung der Fehler auf funktionaler Ebene und diskrete Fehlermodi. Die Übersetzung der Beschreibungssprache für symbolisches Model-Checking wird von Bozzano [BCL+11] gezeigt. Analysiert wird das System mittels einem On-the-Fly Model-Checking Algorithmus [ADI+17]. Dabei können Hardware-Umfänge und deren Fehler analog berücksichtigt werden [LL14]. Die präsentierten Anwendungsfälle bilden die Komplexität einer vernetzten Fahrzeugfunktion nicht ausreichend ab,

---

<sup>1</sup><https://altarica.labri.fr/wp/>

um Aussagen bezüglich der Anwendbarkeit abzuleiten.

Chen et al. [CJWZ17] stellen ein hierarchisches Modell und dessen Analyse mittels symbolischem Model-Checking vor. Die Modellierung beinhaltet dabei sowohl Hardware- als auch funktionale Umfänge sowie interne und externe Fehler durch spezifische Fehlermodi. Durch die hierarchische Modellierung kann sowohl vertikale als auch horizontale Fehlerpropagation dargestellt werden. Chen et al. [CJWZ17] nutzen darüber hinaus die Gegenbeispiele des Model-Checkings um Fehlerbäume zu generieren.

Neben den vorgestellten Ansätzen werden in der Fahrzeugentwicklung Simulationen eingesetzt. Anhand ausgewählter Arbeiten wird das Vorgehen und der Modellierungsumfang zur Untersuchung der Fehlerpropagation in E/E Umfängen aufgezeigt. Grundlagen der Simulation werden von Reif [Rei14] beschrieben.

Reiter et al. [RPV+01] nutzen Erweiterungen von Systemen durch Simulationsmodelle für eine Fehlerinjektion. Dabei werden Hard- und Software berücksichtigt. Die Spezifikation der diskreten Fehlermodi erfolgt dabei individuell je nach Komponente. Für Signale werden die Definition *fehlerhaftes Signal*, *kein Signal*, *zeitliche Abweichung* und *hängendes Signal* berücksichtigt. Das Zeitverhalten wird über die Stimulation beeinflusst. Die Verifikation erfolgt anhand des Vergleichs der Messungen am Ausgang. [RPV+01] Ehret [Ehr01] stellt ebenfalls ein auf Simulationen basierendes Vorgehen zur Analyse der Fehlerpropagation vor. Im Gegensatz zu [RPV+01] erfolgt, neben der Fehlerinjektion, auch eine Stimulation des Modells mittels Eingangsvektoren entsprechend definierter Testfälle. Die untersuchten Fehlermodi stellen Hardwarefehler dar. Das Verhalten wird anhand des Modellausgangs ausgewertet. [Ehr01]

Zur Analyse von Fehlerpropagation werden verschiedenen Ansätze untersucht. In modularen Fehlerbäumen wird die Fehlerlogik direkt modelliert. Modelle für die Analyse mittels Model-Checking und Simulationen, berücksichtigen sowohl Hardware- als auch Funktionsumfänge inklusive der jeweiligen Fehlermodi und des zeitlichen Verhaltens. Das deckt sich mit Veröffentli-

chungen zu modellbasierter Entwicklung von Fahrzeugsystemen. Kugele [Kug12] und Broy et al. [BFG+08] beispielsweise beschreiben analog eine Modellierung der funktionalen-, der Hardware- sowie plattformspezifische Umfänge ohne Bezug zu Fehleranalysen. Die Modelle werden hierarchisch aufgebaut. Fehlermodi werden diskret, entsprechend der physikalischen Mechanismen, beschrieben.

### 3.5 Quantitative Analyse von mehrkanaligen Systemen

Der Industriestandard ISO 26262 fordert den Nachweis eines akzeptablen Risikos durch zufällige Fehler [Int18], weswegen quantitative Analysen erforderlich sind. Dabei unterscheiden sich Fail-Operational-Systemen von herkömmlichen Fail-Silent Systemen durch die Mehrkanaligkeit und verschiedene Betriebsmodi mit individuellen Betriebsintervallen (vgl. Kapitel 2.3). In einem Fail-Operational-System ist der Ausfall abhängig vom Systemzustand und damit von Vorereignissen. In diesem Kapitel werden Arbeiten diskutiert, die sich mit der quantitativen Analyse von dynamischen Systemen unter Berücksichtigung von Zustandsreihenfolgen und verschiedenen Betriebsmodi und Betriebsphasen beschäftigen. Der Fokus liegt dabei auf Fehlerbaumanalysen als etablierten Ansatz, welcher auch für die Analyse komplexer Systeme geeignet ist [Eri16]. Die Grundlagen der Fehlerbaummodellierung werden in Kapitel 2.4 und von Edler [ESH15] beschrieben. Den aktuellen Stand der Wissenschaft bezüglich Fehlerbaumanalysen diskutieren Ruijters und Stoelinga [RS15]. Die Ansätze werden im Folgenden vorgestellt.

In der Arbeit von Schnellbach [Sch16] werden die Abhängigkeiten bei redundanten Strukturen vereinfacht durch Und-Logiken verbunden. Demnach wird die Reihenfolge der Ausfälle nicht berücksichtigt. Der Autor weist darauf hin, dass dynamische Fehlerbäume eine Optimierungsmöglichkeit bezüglich der Genauigkeit darstellen [Sch16]. Ericson [Eri16] zeigt, dass die Abweichungen für einfache Systeme, vergleichbar mit Abbildung 2.13, erst für Betriebszeiten, die über der Fahrzeuglebensdauer liegen signifikante



Abweichungen der Unzuverlässigkeit ergeben, erklärt aber auch, dass dies abhängig vom System ist.

Dynamische Fehlerbäume berücksichtigen Reihenfolgen der Ereignisse [RS15]. Hierzu werden entsprechende Elemente zur Abbildung, unter anderem das Prioritäts-Und Symbol (vgl. Abbildung 2.12), Dieser eingeführt. [RS15] Dugan et al. [DBB93] stellen eine Berechnungsmethode dynamischer Fehlerbäume vor. Dazu wird der Fehlerbaum in ein Markov-Modell transformiert. Der Ansatz wird auch in kommerziellen Software-Werkzeugen, wie Isograph, Reliability Workbench<sup>1</sup> genutzt. Boudali et al. [BCS07] präsentieren eine Erweiterung mittels interaktiven Eingangs/Ausgangs Markov-Modellen. Diese integrieren Eingangs und Ausgangsaktionen aus der Automatentheorie in Transitionen einer Markov-Kette. Zusätzlich werden Transitionsbedingungen berücksichtigt. Die Übersetzung erfolgt automatisiert anhand definierter Modellierungsregeln. Die quantitative Analyse erfolgt durch die Simulation des Markov-Modells. [BCS07]

Eine direkte Berechnungsmethode wird in der Arbeit von Schilling [Sch09] vorgestellt. Diese erfolgt, neben der Ausfallwahrscheinlichkeit des primären Ereignisses, mittels der Ausfalldichte des sekundären Ereignisses. Die Ausfalldichte ist die Ableitung der Ausfallwahrscheinlichkeit. Die Berechnungsmethode ermöglicht eine direkte Bestimmung ohne die Transformation in ein Markov-Modell, erfordert allerdings zusätzliche Informationen über die Ausfalldichte. [Sch09]

Neben der Darstellung der Abhängigkeiten ist die Berücksichtigung von Betriebsintervallen (vgl. Kapitel 2.3) für die Analyse von Fail-Operational-Systemen relevant. In der Luft- und Raumfahrt gibt es detaillierte Vorgaben zu Fehlerbaumanalysen. Vesely et al. [VSD+02] beschreiben im Fault-Tree Handbook der National Aeronautics and Space Administration (NASA) mögliche Ansätze. Ereignisse, welche über verschiedene Intervalle relevant sind, werden in Ereignisse für die jeweiligen Betriebsmodi aufgeteilt und durch ein logisches *exklusives Oder* verknüpft. Die Berechnung erfolgt für die je-

---

<sup>1</sup><https://www.isograph.com/software/reliability-workbench/>

weiligen Minimalschnitte mit der kumulierten Ausfallwahrscheinlichkeit der Vorereignisse. [VSD+02] Ohne Unterstützung durch ein Software-Werkzeug ist die Methode jedoch sehr aufwändig. [VSD+02]

Vaurio et al. [Vau01] stellen einen ähnlichen Ansatz für nicht reparierbare Systeme vor. Dazu werden analog zu [VSD+02] die Basisereignisse in einzelne Ereignisse für jede Phase aufgeteilt. Dann wird ein separater Fehlerbaum für jede Phase erstellt bei dem die relevanten Basisereignisse berücksichtigt werden. Die Bäume für jede Phase werden mittels einem *Oder* zur Gesamtwahrscheinlichkeit zusammengefügt. [Vau01] Die Berechnung der Ausfallwahrscheinlichkeit jedes Basisereignisses erfolgt über die Zeit der Betriebsphase. Die Reparierbarkeit wird als zusätzlicher Term zu Beginn der Phase berücksichtigt und bestimmt während der Phase das Integrationsintervall. [Vau01]

La Band und Andrews [LA04] berücksichtigen die Wahrscheinlichkeit, dass die vorherigen Betriebsphasen ohne Fehler abgeschlossen wurde als die Vorbedingung für den Eintritt in die nächste Phase. Abbildung 3.2 zeigt einen Fehlerbaum, der zwei Betriebsphasen, entsprechend [LA04], darstellt.

Ein Ausfall des Systems tritt ein, wenn das System in Phase 1 oder Phase 2 ausfällt. Ein Fehler in Phase 2 kann nur dann eintreten, wenn Phase 1 nicht bereits zum Ausfall geführt hat, was durch die komplementäre Wahrscheinlichkeit des Ausfalls in Phase 2 dargestellt wird. Die Wahrscheinlichkeit der einzelnen Ereignisse wird über deren relevantes Zeitintervall integriert, Reparierbarkeit wird analog durch das Integrationsintervall berücksichtigt. Die Berechnung der komplexen Baumstrukturen erfolgt nach La Band und Andrews [LA04] durch binäre Entscheidungsdiagramme.

Für die Berücksichtigung der dynamischen Umschaltung in Fehlerbäumen werden sowohl direkte als auch indirekte Berechnungsmethoden über Markov-Modelle genutzt. Allerdings liefert unter Umständen auch eine Vereinfachung durch *Oder-Gatter* abhängig vom System gute Näherungen. Für die Berücksichtigung von Betriebsphasen wird eine Aufteilung derer empfohlen. Dabei gibt es verschiedene Möglichkeiten der Strukturierung,

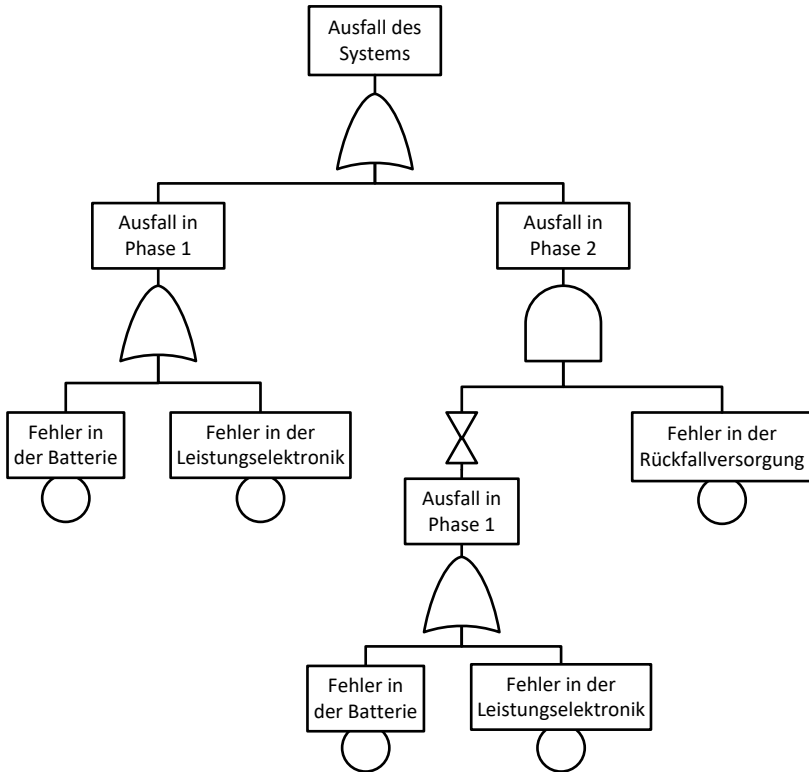


Abbildung 3.2: Berücksichtigung verschiedener Betriebsphasen im Fehlerbaum nach La Band und Andrews [LA04]

wie einzelne Baumstrukturen für jede Phase oder die Aufteilung in Einzereignisse. Die Berechnung der Einzel-Wahrscheinlichkeiten erfolgt durch Integration über das relevante Zeitintervall, die Betriebsphase oder das Reparaturintervall.

## 3.6 Zusammenfassung des Forschungsstands

Im Folgenden wird der Stand der Wissenschaft nochmal zusammengefasst. Dabei werden sowohl Konzepte als auch vorhandene Forschungslücken, welche im Rahmen dieser Arbeit adressiert werden, aufgegriffen.

Einzelne relevante Aspekte der funktionalen Sicherheit von Fail-Operational-Systemen wurden in der Literatur identifiziert und adressiert. Einen bedeutenden Betrag hat die Arbeit von Schnellbach [Sch16] geliefert, welche auch in der ISO 26262:2018 berücksichtigt wurde. Darüber hinaus wird insbesondere die Zuverlässigkeit solcher Systeme in der Literatur adressiert, Umschaltkonzepte oder weiterführende Sicherheitsanalysen werden nicht behandelt. Eine hinreichende Sicherheitsargumentation für Fail-Operational-Systeme gemäß ISO 26262 ist nicht verfügbar und stellt eine Forschungslücke dar. Vorgehen zur Erstellung von Sicherheitsargumentationen, auch unter Berücksichtigung der ISO 26262, werden in der Wissenschaft beschrieben. Dabei stellt die Goal Structuring Notation eine nachvollziehbare und strukturierte Möglichkeit für solche Argumentationen dar, welche auch im Kontext der ISO 26262, nicht jedoch für Fail-Operational Systeme publiziert wurde. Als Basis werden funktionale Modelle und Fehlermodelle genutzt und die ISO 26262 durch die Ergänzung relevanter Attribute der Sicherheitsanforderungen in die Argumentation integriert. In der folgenden Arbeit wird diese Forschungslücke durch die Vorstellung einer hinreichenden Sicherheitsargumentation gemäß ISO 26262 für Fail-Operational-Systeme und deren Herleitung auf Basis einer Goal Structuring Notation adressiert.

Bezüglich der Analyse gemeinsamer Ausfälle ist ein Industrie-übergreifendes generisches Vorgehen vorhanden, welches auch in der ISO 26262 empfohlen wird. Detailanalysen können auf unterschiedliche Art erfolgen, wobei insbesondere Checklisten-basierte Verfahren in der Literatur verwendet werden. Schnellbach präsentiert mit dem Abhängigkeitsmodell eine Erweiterung zur besseren Strukturierung der Analyse. Solche Checklisten sind domänenspezifisch. Deren Erstellung und Wartung, um deren Vollständigkeit

zu garantieren, wird in Publikationen nur rudimentär behandelt. Auch auf Randbedingungen komplexer Systeme, wie beispielsweise eine verteilte Entwicklung oder die Betrachtung kompletter Wirkketten, wird nicht eingegangen. Ebenso stellt die Effizienz nur einen Nebenaspekt der Arbeiten dar. Die Identifikation der Anforderungen an eine solche Analyse anhand einer Studie und deren Berücksichtigung in einem Vorgehen, basierend auf der Literatur, stellen einen Beitrag dieser Arbeit dar.

Im Bereich des Model-Checkings haben einige Arbeiten die Anwendung formaler Methoden zur Verifikation von Fahrzeugfunktion untersucht. Diese zeigen die Vorteile formaler Methoden durch den Nachweis der Korrektheit und die Steigerung der Anforderungsqualität, allerdings auch Einschränkungen durch lange Rechenzeiten. Die in der Literatur präsentierten Anwendungsfälle stellen jedoch abstrahierte Fahrzeugfunktionen dar, auf deren Basis eine Anwendbarkeit bisher nur für Konzeptentscheidungen oder Einzelaspekte bewertet werden konnte. Auch eine Berücksichtigung der ISO 26262 erfolgt nicht hinreichend. Die Untersuchung praxisrelevanter, komplexer Funktionen wurde im veröffentlichten Stand der Wissenschaft nicht behandelt. Die Integration möglicher Optimierungsansätze wurde im Rahmen von Fallstudien daher nicht untersucht, wenngleich die Forschung Ansätze bietet. Die Anwendbarkeit von Model-Checking Ansätzen für eine Sicherheitsargumentation und zugleich eine Methodik für die Sicherheitsanalyse einer Umschaltlogik werden mit der Definition eines Vorgehens gemäß den Vorgaben der ISO 26262 und der Analyse eines industrierelevanten Systems, einer Umschaltlogik für Fail-Operational-Systeme, im Rahmen dieser Arbeit behandelt.

Für die Analyse von Fehlerpropagationen werden verschiedene Modellierungsansätze wie modulare Fehlerbäume oder graphen-basierte Ansätze vorgestellt. Vergleichbar ist dabei der Modellierungsumfang, der insbesondere zeitliche Aspekte und verschiedene Fehlermodi umfasst. Die Evaluation der Modelle erfolgt vorwiegend simulativ, Arbeiten zu formalen Methoden dagegen unterliegen wiederum den bereits genannten Limitierungen. Die Fehlerpropagation in einem Fail-Operational System und der entsprechende

Sicherheitsnachweis gemäß ISO 26262 wird anhand eines systemspezifischen Model-Checking Ansatzes adressiert.

Für die quantitative Analyse von Fail-Operational-Systemen ist die Berücksichtigung dynamischer Aspekte notwendig. Dies beinhaltet die Modellierung von Betriebsphasen und Ausfällen abhängig von Betriebsmodi und damit von Vorbedingungen. Der Stand der Wissenschaft umfasst dabei verschiedene Konzepte zur Modellierung entsprechender Fehlerbäume, welche sich im Grad der Abstraktion und dem Modellierungsaufwand unterscheiden. Vergleichbare Systeme zu Fail-Operational Fahrzeugführungen, wie zum Beispiel das in Kapitel 2.3 gezeigte System, wurden in Publikationen nicht behandelt. Eine Untersuchung der akzeptablen Näherungen im Rahmen einer Fehlerbaumanalyse erfordert daher weitere Untersuchungen, welche in dieser Arbeit behandelt werden. Darauf basierend wird ein möglicher Modellierungsvorschlag und die Generalisierbarkeit untersucht.

Die identifizierten Forschungslücken werden in den folgenden Kapiteln adressiert. Die vorliegende Arbeit leistet damit einen Beitrag zur funktionalen Sicherheit von hochautomatisierten Fahrzeugsystemen. Dieser besteht sowohl in der Vorstellung einer hinreichenden Sicherheitsargumentation als auch der einzelnen notwendigen Analysen.

# SICHERHEITSARGUMENTATION GEMÄSS ISO 26262

Eine Sicherheitsargumentation zeigt gemäß ISO 26262 nachvollziehbar das Erreichen funktionaler Sicherheit. Die Argumentation dient als Basis für einen Safety-Case und beinhaltet zum einen Sicherheitsziele und Sicherheitsanforderungen und zum anderen die Begründung und den Nachweis deren Erfüllung (vgl. Kapitel 2.2). Damit gehen Nachweis- und Analyseziele, sowie eine Strukturierung der Analysen, ebenso wie Schnittstellen aus der Argumentation hervor. Die Sicherheitsanalysen unterstützen die Argumentation durch die Verifikation des Systemdesigns und damit einhergehend der Identifikation möglicher Fehler. [Int18] In dieser Arbeit wird eine solche Sicherheitsargumentation zur Identifikation notwendiger und hinreichender Sicherheitsanalysen genutzt. Die Argumentation gibt daher den Umfang und das Ziel der einzelnen Analysen vor, welche in den darauffolgenden Kapiteln behandelt werden.

Um die notwendigen und hinreichenden Sicherheitsanalysen für die Argumentation der funktionalen Sicherheit zu identifizieren, wird eine Sicherheitsargumentation für ein generisches Fail-Operational-System (vgl. Abbildung 2.9) abgeleitet. Das Vorgehen berücksichtigt sicherheitsrelevante Systemaspekte aus der Beschreibung von Sicherheitsanforderungen der ISO 26262 und mögliche Fehler des Systems. Eine Übersicht über das Vorgehen wird in Abbildung 4.1 anhand der Prozessschritte dargestellt.

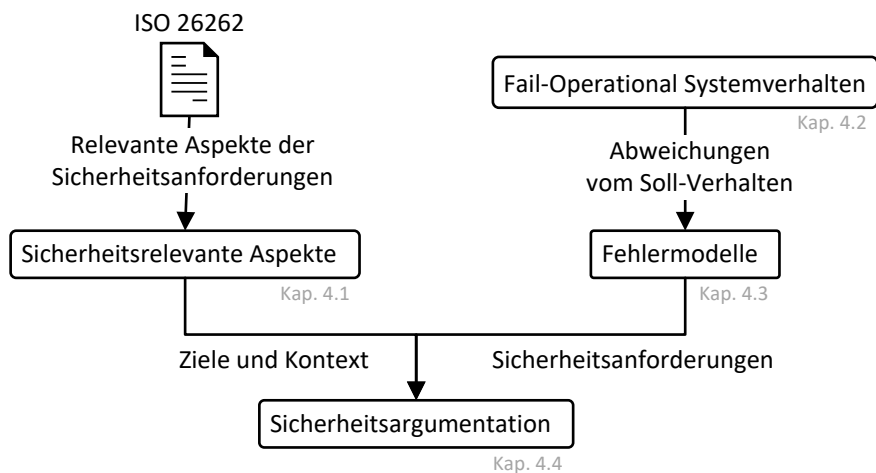


Abbildung 4.1: Vorgehen für das Ableiten einer Sicherheitsargumentation

Im Standard ISO 26262 werden die einzelnen Arbeitsprodukte des Sicherheitslebenszyklus definiert. Aus der Gefahren- und Risikoanalyse ergeben sich Sicherheitsziele. Auf Basis der Sicherheitsziele werden Sicherheitsanforderungen abgeleitet. Die Allokation der Anforderungen stellt das funktionale, beziehungsweise technische Sicherheitskonzept dar und liefert damit eine Strategie für das Vermeiden von Sicherheitszielverletzungen [Hil12, Int18]. Das Ziel der Sicherheitsanalysen ist, neben der Identifikation von Fehlern, die Verifikation der Sicherheitskonzepte. Die ISO 26262 beschreibt die sicherheitsrelevanten Aspekte der Anforderungen, für welche durch Sicherheitsanalysen ein Nachweis erbracht werden muss (vgl. Kapitel 3.1.2).



Zudem müssen mögliche Fehler des Fail-Operational-Systems für eine hinreichende Argumentation der Sicherheit berücksichtigt werden. Hierfür werden, entsprechend dem Vorgehen in der Literatur (vgl. Kapitel 3.1.2), ausgehend vom Systemverhalten Fehlermodelle unter Verwendung von Fehlerbäumen abgeleitet. Für identifizierte Fehler wird der Nachweis eines ausreichend geringen Risikos erbracht, wofür unter Umständen Maßnahmen definiert werden.

Die relevanten Aspekte aus dem Umfang der Sicherheitsanforderungen gemäß ISO 26262 und die Sicherheitsanforderungen aus der Fehleranalyse werden in der Sicherheitsargumentation konsolidiert. Das Verhindern von Fehlern, ebenso wie Prämissen der Argumentation, beispielsweise Randbedingungen, stellen dabei, unter Berücksichtigung der sicherheitsrelevanten Aspekte, Ziele im Sinne der Sicherheitsargumentation dar. Der Nachweis der funktionalen Sicherheit wird durch Analysen erbracht, weswegen aus der Argumentation deren Notwendigkeit hervor geht. Zur Darstellung der Sicherheitsargumentation wird eine Goal Structuring Notation (vgl. Kapitel 2.2) genutzt [Kel98], welche eine nachvollziehbare und übersichtliche Form bietet.

## 4.1 Sicherheitsrelevante Anforderungen nach ISO 26262

Der Industriestandard ISO 26262 beschreibt die Ziele der Sicherheitsanalysen ebenso wie den Umfang von Sicherheitsanforderungen. Aus der Beschreibung der Sicherheitsanforderungen ergeben sich relevante Attribute, die im Rahmen der Analysen berücksichtigt werden müssen.

Ziel der Sicherheitsanalysen, wie eingangs und in Kapitel 2.4 beschrieben, ist die Verifikation des Systemdesigns und der Sicherheitskonzepte (vgl. Abbildung 4.1). Dabei werden die Sicherheitsanforderungen bezüglich Vollständigkeit sowie deren Konsistenz validiert, das Systemdesign bezüglich deren Erfüllung verifiziert wobei auch mögliche Fehler berücksichtigt

werden. Dadurch wird ein ausreichend geringes Risiko von Sicherheitszielverletzungen sichergestellt.

Ausgangspunkt der Sicherheitsbetrachtung stellen die Sicherheitsziele aus der Gefahren- und Risikoanalyse dar. Durch eine angepasste Fahrstrategie und die Fahrerwarnung ergeben sich unter Umständen geringere Gefährdungen im Rückfallbetrieb. Daher werden in der Gefährdungs- und Risikoanalyse zwischen dem Normalbetrieb und einem Rückfallbetrieb unterschieden und separate Sicherheitsziele abgeleitet (vgl. Kapitel 2.3). [Int18, Sch16] Die Definition der Sicherheitsanforderungen, abgeleitet von Sicherheitszielen, und die Allokation auf funktionale Elemente, stellen das funktionale Sicherheitskonzept dar. Die Detaillierung auf technischer Ebene resultiert im technischen Sicherheitskonzept. Die Sicherheitsanforderungen und deren Attribute ergeben daher in Kombination mit den relevanten Fehlermodi den Umfang der Verifikation und demnach der Sicherheitsanalysen. Relevante Aspekte für Fail-Operational-Systeme werden im Folgenden, auf Basis des in Kapitel 2.3 eingeführten Sicherheitskonzeptes identifiziert. Dabei wird analog der ISO 26262 zwischen funktionalen und technischen Anforderungen unterschieden.

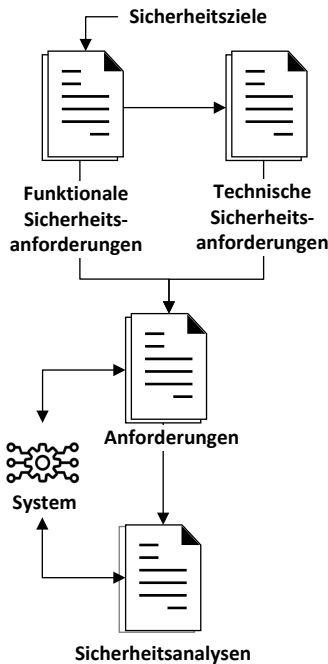
Im funktionalen Sicherheitskonzept erfolgt die Spezifikation und Allokation von Sicherheitsanforderungen entsprechend der Sicherheitsziele auf Elemente der funktionalen Architektur [Int18]. Die relevanten Aspekte funktionaler Anforderungen sind, zusammen mit den technischen Anforderungen und Zielen der Sicherheitsanalysen, in Abbildung 4.2 dargestellt.

Ein zu berücksichtigender Aspekt stellt das Degradationskonzept dar. Dieses umfasst unter anderem den Rückfall- beziehungsweise Notfallbetrieb und die Fahrerwarnung. Im Degradationskonzept werden außerdem Fehlerreaktionen auf funktionaler Ebene spezifiziert. Für das Fail-Operational-System umfasst dies die Betriebsmodi inklusive der aktiven Konfiguration abhängig vom jeweiligen Fehlerfall. Im Zusammenhang mit der Degradation steht auch der Übergang in den sicheren Zustand. Der sichere Zustand ist eine Situation in der die Gefährdung nicht größer als die Integrität des Restsystems ist, beziehungsweise keine Gefährdung vorliegt. In Kapitel 2.3 wurde

ein solches Degradationskonzept auf Fahrzeugebene mittels Umschaltung in einen Rückfallbetrieb und Verzögerung beschrieben. Der sichere Zustand der Fahrzeugführung unterscheidet sich dabei von dem der Komponenten. Komponenten und Subsysteme können im Fehlerfall abschalten, dürfen dabei andere Subsysteme allerdings nicht beeinträchtigen. Dieses Verhalten wird als Fail-Silent beschrieben. Darüber hinaus werden die jeweiligen Reaktionszeiten und Übergangszeiten in Anforderungen spezifiziert. Diese beinhalten die Fehlertoleranzzeiten und die Fehlerbehandlungszeiten auf Fahrzeug-, System- und Subsystemebene. Dabei müssen Erkennungszeiten und Reaktionszeiten, wie die Umschaltzeit auf Systemebene und die Rekonfigurationszeit berücksichtigt werden. Fehlertoleranz und Fehlervermeidung sind elementare Attribute eines Fail-Operational-Systems und sind auf Systemebene in Kombination mit Redundanzen und dem Umschaltkonzept zu betrachten. Im Gegensatz zu Fail-Silent Systemen stellt eine Unverfügbarkeit des Systems bei Fail-Operational-Systemen eine Verletzung von Sicherheitszielen dar und muss daher im Besonderen berücksichtigt werden.

Den nächsten Detaillierungsgrad des Sicherheitskonzeptes stellt das technische Sicherheitskonzept dar. Dabei werden ausgehend von den funktionalen Sicherheitsanforderungen technische Sicherheitsanforderungen abgeleitet und auf die technische Architektur und damit auch auf Hardwareelemente allokiert. Im Folgenden werden solche Aspekte näher erläutert, welche erst auf technischer Ebene relevant sind.

Bezüglich der Fehlervermeidung und Fehlertoleranzmaßnahmen müssen sowohl interne als auch externe Fehler, an Schnittstellen, berücksichtigt werden. Dabei beinhalten die technischen Anforderungen die Spezifikationen der Erkennungsmechanismen, Meldungen und die Behandlung der Fehler, beispielsweise die Berechnung von Signalen mittels Ersatzwerten. Im Kontext des Fail-Operational-Systems müssen Fehler einer Komponente der Umschaltlogik gemeldet werden und eine Propagation verhindert werden. Dies beinhaltet auch potentiell latente Fehler in Komponenten im Rückfallkanal während des Standby-Modus. Die ISO 26262 definiert



#### Sicherheitsrelevante Aspekte

- Degradation  
Rück- und Notfallbetrieb, Fahrerwarnung
- Sicherer Zustand und Übergang
- Zeitliche Aspekte  
Fehlertoleranz, Fehlerbehandlungszeit auf Fahrzeug-, System-, Subsystemebene
- Fehlervermeidung/Toleranz  
Entschärfung, Erkennung, Mitigation, Anzeige, Fehlerkontrolle
- Redundanzen
- Maßnahmen auf Hardware- und Software-Ebene entsprechend des ASIL

#### Ziele der Sicherheitsanalysen

- Sicherstellen von ausreichend geringem Risiko durch Verletzung von Sicherheitszielen
- Verifikation des Designs und der Sicherheitsmechanismen mit entsprechendem ASIL
- Vervollständigung/ Unterstützung der Spezifikation
- Identifikation von Fehlerursachen und Effekten

Abbildung 4.2: Sicherheitsrelevante Attribute der Sicherheitsanforderungen nach ISO 26262 [Int18]

außerdem die Ordnung der Fehlerkombination, welche betrachtet werden muss. Es wird, wie in Kapitel 2.1.2 eingeführt, zwischen sicheren Fehlern, Einfach- und Mehrfachfehlern, inklusive latenter Fehler, unterschieden. Die ausgehend vom Sicherheitskonzept relevanten Fehlerkombinationen definieren den Umfang der Analysen. [Int18] In den Anforderungen wird außerdem das Integritätslevel (ASIL) berücksichtigt. Neben Entwicklungs- und Verifikationsmaßnahmen für Software und Hardware ergeben sich für Hardwareumfänge zusätzliche Anforderungen an die Erfüllung von Metriken und Grenzwerte für die Häufigkeit zufälliger Fehler. Die Wahrscheinlichkeit einer Sicherheitszielverletzung wird dabei durch Ausfallraten, eine Diagnoseabdeckung und die relevanten Zeitintervalle bestimmt. Insbesondere nicht

diagnostizierte Fehler sind für Fail-Operational-Systeme kritisch, da diese zu einem direkten Ausfall auf dem Nominalkanal beziehungsweise zum Ausfall direkt nach der Umschaltung auf dem Rückfallkanal führen.

## 4.2 Modellierung des Fail-Operational-Systemverhaltens

Für eine Sicherheitsargumentation und damit die Ableitung der notwendigen Analysen müssen, wie in Abbildung 4.1 gezeigt, neben den Vorgaben der ISO 26262, das Systemdesign und die relevanten Fehlermodi berücksichtigt werden. Die Ableitung der Fehlermodi erfolgt dabei, wie in der Literatur vorgeschlagen (vgl. Kapitel 3.1.2), auf Basis des Systemverhaltens. Im Folgenden wird das Fail-Operational-Verhalten des Fahrzeugsystems modelliert, um in Kapitel 4.3 Fehlermodi aus Abweichungen zum gewünschten Verhalten abzuleiten.

Das Fail-Operational-Verhalten wird auf Systemebene durch die Betriebsmodi beschrieben. Abbildung 4.3 zeigt das Verhalten des Fahrzeugsystems anhand eines Zustandsdiagrammes unter Verwendung der UML-Notation. Das Fail-Operational-Verhalten beschränkt sich auf den hochautomatisierten Fahrmodus, daher sind nur die relevanten Zustände und Übergänge dargestellt. Das Degradationskonzept wurde in Kapitel 2.1.2 beschrieben. Der Betrieb im hochautomatisierten Fahrmodus beginnt durch die Aktivierung seitens des Fahrers. Dabei werden systemseitig relevante Vorbedingungen, wie die Bereitschaft der Systeme oder die Freigabe der Strecke, geprüft. Der Betrieb erfolgt im Nominalmodus, bei dem der Nominalkanal die Fahrzeugführung übernimmt und der Rückfallkanal übernahmebereit ist. Den Modus kann der Fahrer jederzeit beenden und die Fahraufgabe übernehmen. Im Fehlerfall muss zwischen diagnostizierten und nicht diagnostizierten Fehlern unterschieden werden. Letztere führen zu einem Systemausfall, da keine Reaktion des Systems erfolgen kann. Wird ein Fehler auf dem Nominal- oder Rückfallkanal erkannt, wechselt das System in einen Rückfallbetrieb. Eine Diagnose ist daher notwendig, um Fehlertoleranz und

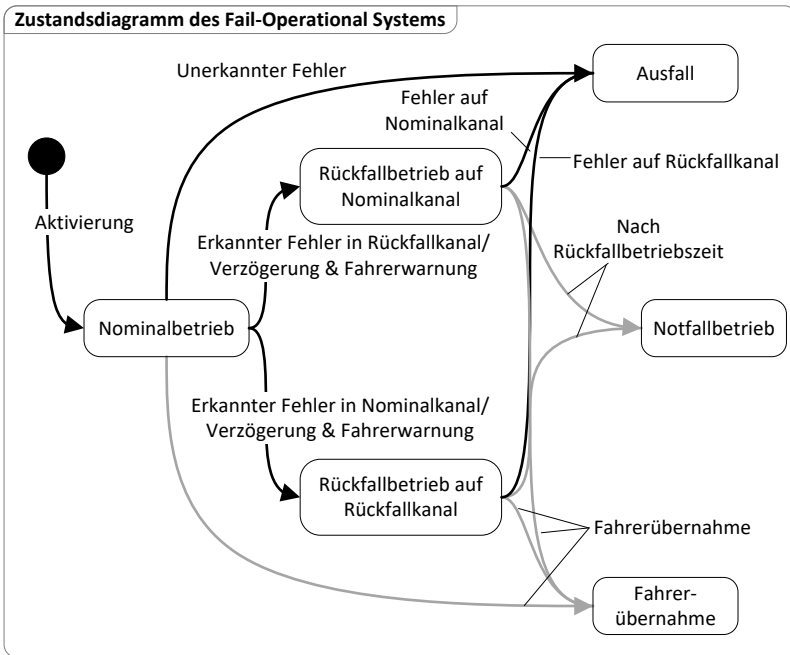


Abbildung 4.3: Zustandsdarstellung des Fail-Operational-Systemsverhaltens

damit das Fail-Operational-Verhalten sicherzustellen. Abhängig von der betroffenen Komponente wird dabei eine Systemkonfiguration aktiviert, die die Fahrzeugführung ermöglicht. Das bedeutet, nach Fehlern auf dem Nominalkanal, wird der Rückfallkanal aktiviert. Fehler auf dem Rückfallkanal führen zu dessen Deaktivierung und dem Rückfallbetrieb im Nominalkanal. Zwischen Betriebsmodi auf dem Rückfallkanal wird erst im nächsten Schritt differenziert (vgl. Abbildung 4.5). Mit dem Übergang in den Rückfallbetrieb wird außerdem eine Aufforderung zur Fahrerübernahme gegeben und ein Übergang in einen Fahrzustand, der der Restintegrität des Systems entspricht, durchgeführt [Int18]. Wenn der Fahrer die Fahraufgabe übernimmt wird der Fail-Operational-Betrieb beendet. Passiert dies nicht, erfolgt nach einer definierten Zeit ein Notfallbetrieb mit dem Ziel, das Fahrzeug in einen final sicheren Zustand, beispielsweise den Stillstand auf dem Standstreifen,

zu bewegen. Kritisch sind Zweitfehler während des Rückfallbetriebes welche in dem aktiven Kanal auftreten, beispielsweise ein Fehler im Rückfallkanal, wenn der Nominalkanal bereits ausgefallen ist. Ein solcher Fehler würde zum Verlust der Fahrzeugführung und zur Verletzung von Sicherheitszielen führen. Die Diagnose der Fehler ist dann irrelevant.

Die Beschreibung im Zustandsdiagramm bildet dabei die Systemebene ab. Auf Komponentenebene sind für den Wechsel der Betriebsmodi Teilschritte notwendig. Abbildung 4.4 zeigt diese anhand einer Umschaltung auf den Rückfallkanal in einem Aktivitätsdiagramm. Mehrfache Umschaltungen oder ein Komplettausfall erfolgen analog.

Wie in Abbildung 4.3 gezeigt, wird der Nominalbetrieb durch die Aktivierung gestartet. Dabei ist der Rückfallkanal in einem Standby-Modus. Im Falle eines kritischen Fehlers auf dem Nominalkanal ist, wie beschrieben, die Diagnose des Fehlers notwendig. Wenn der Fehler durch die Diagnosemechanismen erkannt wird, erfolgt eine Reaktion. Diese umfasst entweder eine interne Degradation, beispielsweise den Wechsel auf sichere Ersatzwerte eines Nutzsymbols, oder aber ein Fail-Silent Verhalten der Funktion oder Komponente. Außerdem sind Maßnahmen zum Verhindern der Fehlerpropagation notwendig. Diese sind sowohl auf Komponentenebene, zum Beispiel durch die Degradation von Qualifier-Signalen, Signalen, die die Informationsgüte beschreiben, als auch auf Systemebene, wie der Meldung an die Umschaltlogik, umgesetzt. Meldet eine Komponente einen Fehler, was auch durch das Ausbleiben einer Statusmeldung erfolgen kann, erfolgt zuerst die Reaktion im entsprechenden, primären Zustandsautomat. Im nächsten Schritt reagieren die weiteren, sekundären Zustandsautomaten durch das Deaktivieren des Nominalkanals sowie das Aktivieren des Rückfallkanals. Die Komponenten reagieren auf den Status der Zustandsautomaten und der Rückfallkanal übernimmt im Rückfallbetrieb die Fahrzeugführung. Ein Zweitfehler würde einen analogen Ablauf auf dem Rückfallkanal bedingen. Zusätzlich würde entsprechend Abbildung 4.3 ein Notfallbetrieb ausgelöst werden, wenn kein funktionsfähiger Kanal mehr verfügbar ist.

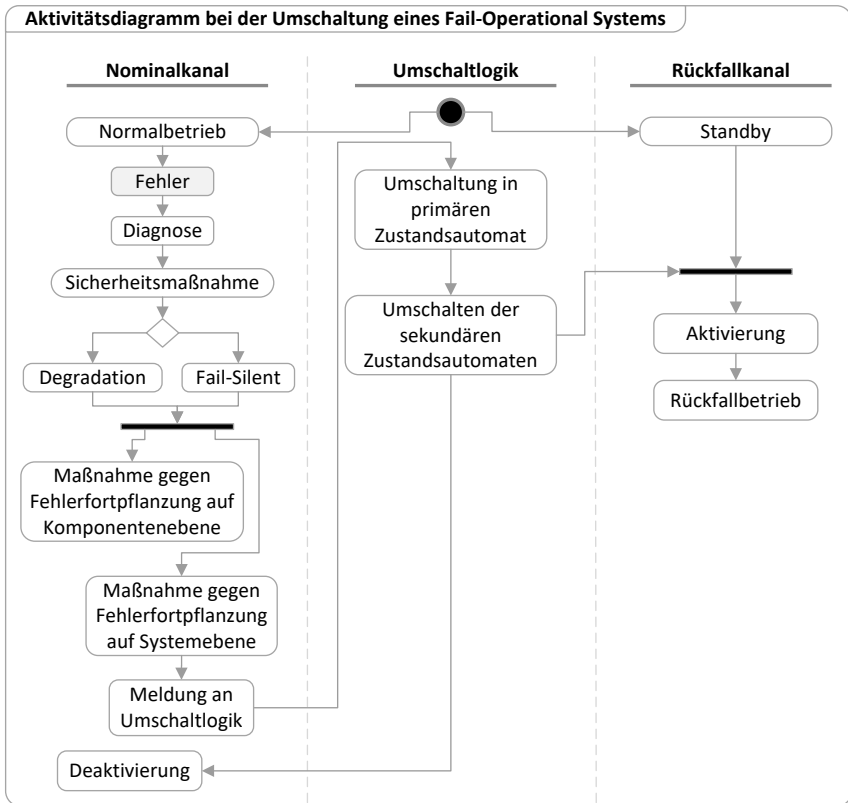


Abbildung 4.4: Aktivitätsdiagramm der Reaktion des Systems auf Komponentenebene im Fall eines Doppelfehlers

### 4.3 Ableitung von Fehlermodellen

Durch Abweichungen vom beschriebenen, spezifizierten Verhalten erfolgt im Folgenden die Identifikation von Fehlern. Entsprechend der ISO 26262 werden nur Fehler in elektrischen und elektronischen Systemen berücksichtigt. Systemgrenzen, beispielsweise von Sensoren, stellen keine Fehler im Sinne der funktionalen Sicherheit nach ISO 26262 dar.



Der abgeleitete Fehlerbaum für die Verletzung des Sicherheitsziels, *Kollision durch Verlassen der Trajektorie*, einen Fehler der Fahrzeugführung, ist in Abbildung 4.5 dargestellt.

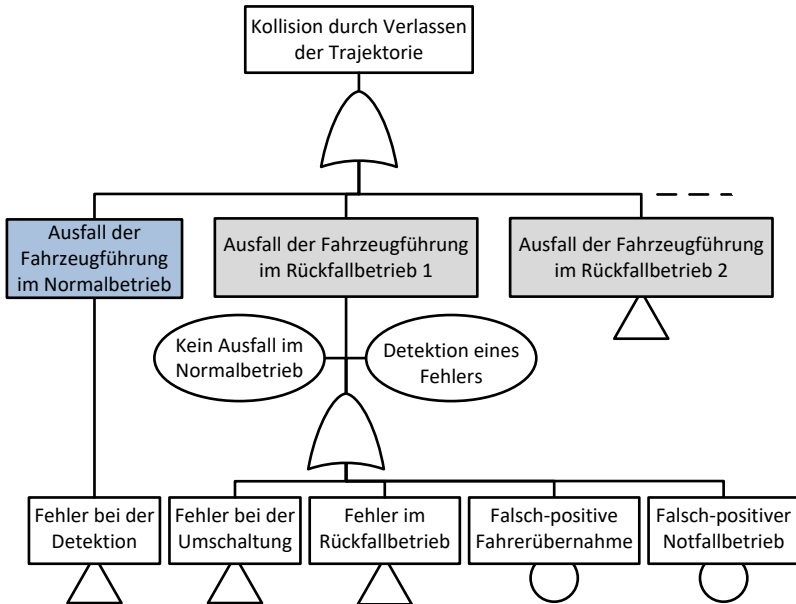


Abbildung 4.5: Schematischer Fehlerbaum für das Hauptereignis *Kollision durch ein Verlassen der Trajektorie*

Zuerst erfolgt eine Aufteilung entsprechend der Betriebsmodi. Ausgehend vom Zustandsdiagramm in Abbildung 4.3 werden im Fehlerbaum die Betriebsmodi mit einem Übergang zum Zustand *Ausfall* im Zustandsdiagramm berücksichtigt. Der Rückfallbetrieb wird dabei direkt in die relevanten Modi unterteilt. Die Detaillierung erfolgt anhand des Aktivitätsdiagrammes in Abbildung 4.4 in dem die Übergänge mit den Teilschritten dargestellt sind. Relevante aber in diesem Fehlerbaum nicht weiter detaillierte Ereignisse, sind mit einem Dreieck dargestellt. Ereignisse mit einem Kreis sind Basisereignisse, die im Rahmen der Analyse nicht weiter detailliert werden, weil der Detaillierungsgrad ausreichend ist oder weil keine Relevanz besteht. Kompo-

nentenausfälle stellen beispielsweise solche Basisereignisse im Rahmen der Systemanalyse dar. Mit der Ellipse wird in einem qualitativen Fehlerbaum eine Bedingung dargestellt. Fehler in den einzelnen Teilschritten führen zum Ausfall, allerdings kann ein späterer Teilschritt nur dann kritisch sein, wenn die Teilschritte zuvor nicht bereits zum Ausfall geführt haben. Abbildung 4.4 zeigt die Diagnose als ersten Schritt. Nicht diagnostizierte Fehler führen direkt zu einem Verlust der Fahrzeugführung im Nominalbetrieb. Für den Rückfallbetrieb stellt die Diagnose daher eine Voraussetzung dar, ebenso darf der Ausfall nicht bereits erfolgt sein. Im nächsten Schritt erfolgt die Kommunikation an die Umschaltlogik und die Umschaltung. Eine inkorrekte Umschaltung führt zum Verlust der Fahrzeugführung, dieser wird in Abbildung 4.6 weiter detailliert. Entsprechend dem Zustandsdiagramm erfolgt nach dem Übergang der Betrieb im Rückfallmodus. Ein Fehler in diesem führt ebenso wie eine falsch-positive Fahrerübernahme und ein falsch-positiv ausgelöster Notfallbetrieb zu einer Verletzung des Sicherheitsziels. Letzteres stellt aufgrund der geringeren Integrität eine Sicherheitszielverletzung dar. Die Fehler sind im Fehlerbaum in Abbildung 4.5 unter dem Ausfall im Rückfallbetrieb aufgeführt.

Der Verlust der Fahrzeugführung durch einen Fehler in der Umschaltlogik wird in Abbildung 4.6 entsprechend dem Aktivitätsdiagramm in Abbildung 4.4 detailliert. Ein Ausfall während der Umschaltung erfordert deren Auslösen, eine Detektion und Kommunikation eines Fehlers, modelliert durch ein *Prioritäts-Und*. Dabei spielt es keine Rolle, ob der Fehler korrekt oder falsch-positiv detektiert wird. Entsprechend des Ablaufs der Umschaltung (vgl. Abbildung 4.4) kann ein Fehlereignis zuerst im primären Zustandsautomat auftreten. Fehler in der Logik oder an Schnittstellen, also der Interaktion zwischen Funktion und Zustandsautomat, führen zu entsprechenden Ausfällen. Eine weitere Detaillierung der Logik ist nicht sinnvoll, da die Reaktion der Logik im Verbund der Zustandsautomaten betrachtet werden muss. Diese stellen daher Basisereignisse dar. Im nächsten Schritt erfolgt die Deaktivierung des Nominalbetriebs, zusätzlich zur Logik und der Interaktion stellt ein funktionales Fehlerverhalten, welche eine

Deaktivierung verhindert, eine Fehlerursache dar. Der Teilschritt ist dabei nur relevant, sofern der primäre Zustandsautomat korrekt geschaltet hat, was aufgrund des Detaillierungsgrades mit einem negierten Ereignis dargestellt wird. Nachfolgend wird der Rückfallbetrieb aktiviert. Auch dieser erfordert, dass ein Ausfall nicht bereits eingetreten ist. Hier kann ein Ausfall zusätzlich erfolgen, wenn die Fahrzeugführung die Fahrsituation nicht kontrollieren kann, zum Beispiel weil Systemgrenzen erreicht sind.

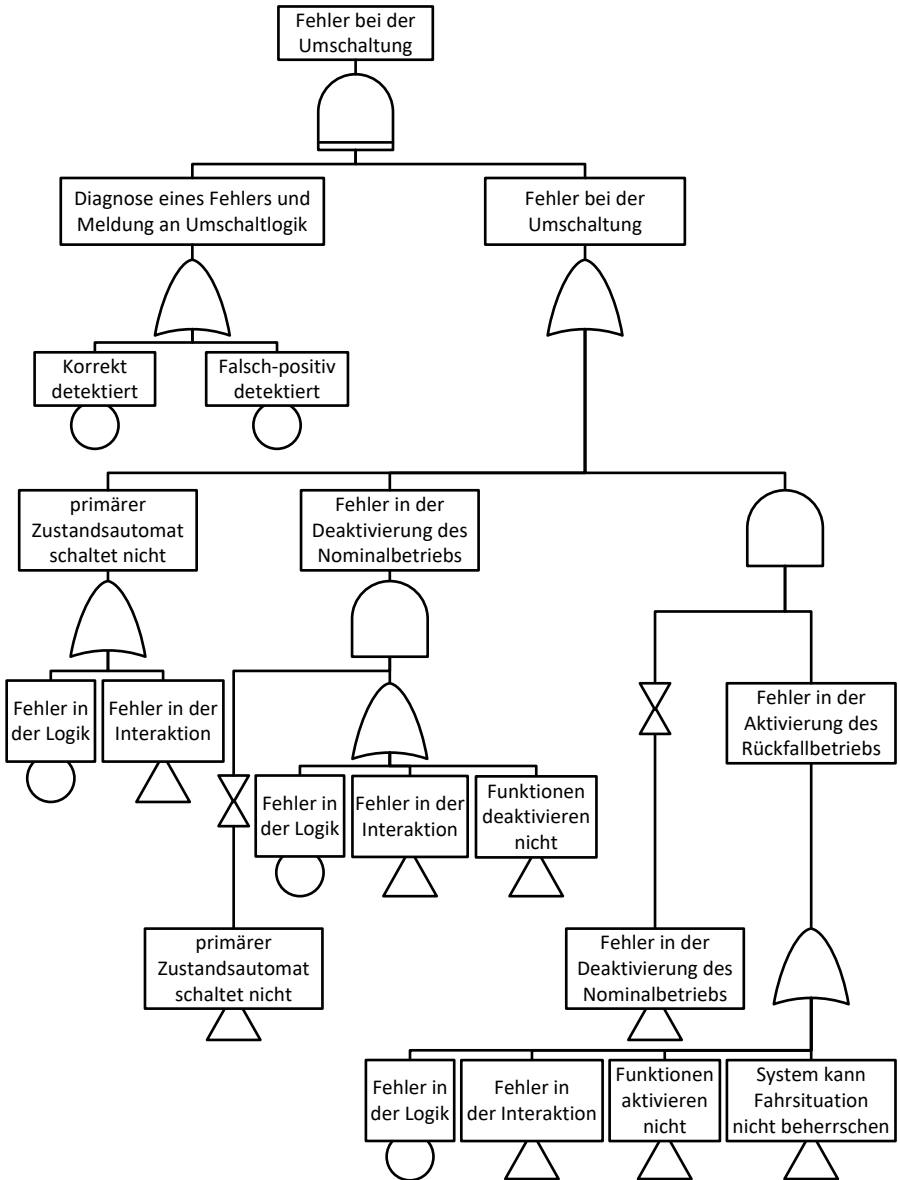


Abbildung 4.6: Fehlerbaumlogik für einen Fehler bei der Umschaltung

## 4.4 Sicherheitsargumentation mittels Goal Structuring Notation

Eine Sicherheitsargumentation stellt den Zusammenhang zwischen den Sicherheitszielen beziehungsweise Anforderungen und den Sicherheitsnachweisen dar, wodurch sich ein Safety Case ergibt. [Int18]

Im Folgenden wird eine Sicherheitsargumentation für ein Fail-Operational-System abgeleitet und daraus notwendige und hinreichende Analysen identifiziert. Da Sicherheitsanalysen sowohl die Verifikation als auch die Fehleridentifikation abdecken müssen, werden, entsprechend Abbildung 4.1, sicherheitsrelevante Aspekte der Anforderungen nach ISO 26262 (vgl. Kapitel 4.1) und die Fehlermodelle des Systems (vgl. Kapitel 4.3) konsolidiert. Die identifizierten Fehler stellen im Sinne der Sicherheitsargumentation die Nachweisziele dar. Der Fokus der Argumentation liegt dabei auf der Systemebene. Das umfasst insbesondere Analysen, welche auf Systemebene oder im Verbund durchgeführt werden müssen. Diese erfordern mitunter Nachweise auf Subsystem- und Komponentenebene, weswegen die Schnittstellen in der Argumentation berücksichtigt werden. Schnittstellen umfassen sowohl intern- als auch extern-entwickelte Umfänge. Im Rahmen der Entwicklung ist es insbesondere notwendig, externe Beiträge frühzeitig zu spezifizieren, um Abstimmungsaufwand und nachträgliche Kosten durch Änderungen des Vergabeumfangs zu minimieren.

Für eine nachvollziehbare Darstellung der Sicherheitsargumentation wird die Goal Structuring Notation von Kelly [Kel98] genutzt. Die Notation wurde in Kapitel 2.2 eingeführt und entsprechend dem Zweck der Argumentation gegenüber der Arbeit von Kelly [Kel98] erweitert. Abbildung 4.7 zeigt die wichtigsten Symbole der Notation und die definierten Erweiterungen für die Sicherheitsargumentation im Rahmen dieser Arbeit.

Ziele (engl. Goals) stellen die Sicherheitsziele und Anforderungen dar, Prämissen werden durch Annahmen berücksichtigt. Bei Lösungen, Nachweisen, wird entsprechend dem Fokus der Argumentation zwischen Solchen auf System- und Subsystem-Ebene unterschieden. Darüber hinaus wird

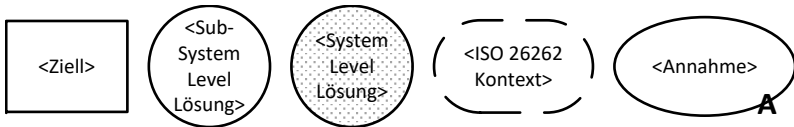


Abbildung 4.7: Wichtigste Symbole der Sicherheitsargumentation auf Basis der Goal Structuring Notation

die hinreichende Erfüllung der ISO 26262 durch ein gesondertes Kontext Symbol hervorgehoben. Weitere verwendete Elemente stellen Beziehungen, Strategien und Begründungen dar (vgl. Kap. 4.1)

Im Zentrum der Argumentation stehen die Nachweisziele. Diese entsprechen den Sicherheitszielen, den ableitenden Sicherheitsanforderungen beziehungsweise der Vermeidung identifizierter Fehlermöglichkeiten entsprechend Kapitel 4.3. Abbildung 4.8 zeigt die Nachweisziele und die Detaillierungsschritte anhand von Strategien. Eine Detaillierung erfolgt bis zu einem Ziel, dessen Erfüllung durch eine Analyse vollständig nachgewiesen werden kann.

Das beispielhaft betrachtete Sicherheitsziel stellt die Vermeidung einer Kollision durch das Verlassen der Trajektorie mit ASIL D dar, welches durch den Ausfall der Fahrzeugführung verletzt wird. Das entsprechende Nachweisziel ist die korrekte und verfügbare Fahrzeugführung. Das Nachweisziel beschreibt das Fail-Operational-Verhalten. Die relevanten Attribute des Sicherheitsziels auf System-Ebene entsprechend Kapitel 4.1 sind der sichere Zustand, das Degradationskonzept inklusive zeitlicher Toleranzen und der ASIL mit entsprechenden Vorgaben. Eine Betrachtung in einzelnen Betriebsmodi ist nicht hinreichend, weswegen Nachweisziele für die Attribute auf Systemebene abgeleitet wurden. Diese umfassen den Nachweis der Verfügbarkeit respektive eines ausreichend geringen Risikos bezüglich zufälliger Hardwareausfälle entsprechen des ASILs. Auch der Übergang in den sicheren Zustand, der durch das Degradationskonzept sichergestellt wird, muss auf Systemebene betrachtet werden. Der sichere Zustand variiert je nach Betriebsmodus. Bei Einzelfehlern im Normalbetrieb stellt die Umschaltung

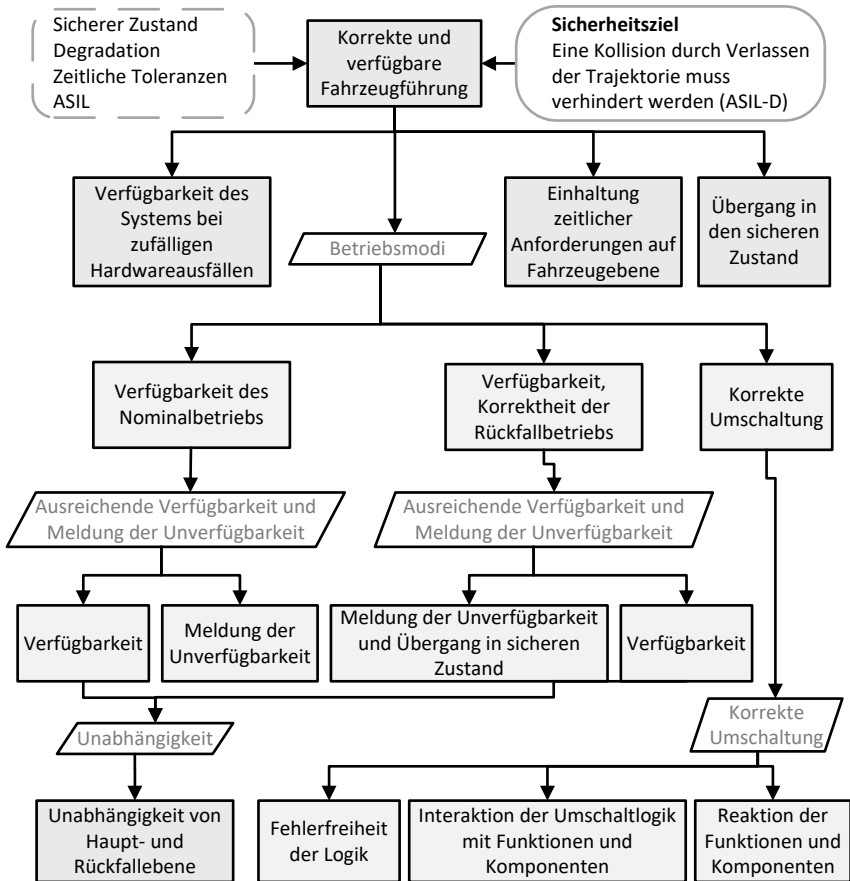


Abbildung 4.8: Ziele im Rahmen der Sicherheitsargumentation

und die Fortführung der Fahrzeugführung im Rückfallbetrieb den sicheren Zustand dar (vgl. Kapitel 2.3). Auch die Reaktionszeiten müssen auf Ebene der Fahrzeugführung als Summe der einzelnen Reaktionszeiten betrachtet werden.

Darüber hinaus ergeben sich durch die Fehlermodelle (Abbildung 4.5 und 4.6) weitere Detaillierungen, die berücksichtigt werden müssen. Entsprechend wird zwischen dem Nominalbetrieb, dem Rückfallbetrieb und der

Umschaltung unterschieden. Die Verfügbarkeit, die Bereitstellung einer Funktion auf Anfrage, stellt dabei das Nachweisziel für den Nominal- und den Rückfallbetrieb dar. Zusätzlich ist die Meldung der Unverfügbarkeit im Fehlerfall ein Nachweisziel. Auch die Korrektheit stellt eine Anforderung dar, im Falle des Nominalbetriebs ist diese im fehlerfreien Fall jedoch nicht Umfang der Sicherheitsbetrachtung [Int18]. Im Rückfallbetrieb geht mit der Meldung der Unverfügbarkeit auch der Übergang in den sicheren Zustand einher. Zusätzlich ist die Unabhängigkeit von Nominal- und Rückfallbetrieb, welche sich als Prämisse aus der Fehlerbaumbetrachtung ergibt, nachzuweisen. Diese ist notwendig, da ein Fehler, der zu einem Rückfallbetrieb führt, diesen nicht ebenfalls deaktivieren darf.

Neben dem Betrieb stellt auch die korrekte Umschaltung ein notwendiges Nachweisziel dar. Dabei muss zum einen die Fehlerfreiheit der Logik und zum anderen die Interaktion zwischen Logik und der Funktionen und deren Reaktion adressiert werden. Diese müssen dem Degradationskonzept entsprechen und damit dem Übergang in einen sicheren Zustand genügen.

Durch Nachweise, Lösungen im Sinne der Goal Structuring Notation, und Begründungen sowie weiteren Elementen (vgl. Abbildung 4.7) werden die Sicherheitsziele zu einer Sicherheitsargumentation ergänzt. Die ISO 26262 und der Kontext des Fail-Operational-Systems werden über Kontext-Symbole berücksichtigt. Die Abbildungen 4.9 und 4.10 zeigen die Argumentation mittels der Goal Structuring Notation, der Übersichtlichkeit wegen aufgeteilt in zwei Teile. Im Folgenden wird die Argumentation für das Erreichen der funktionalen Sicherheit für die einzelnen Nachweisziele in Abbildung 4.8 vorgestellt.



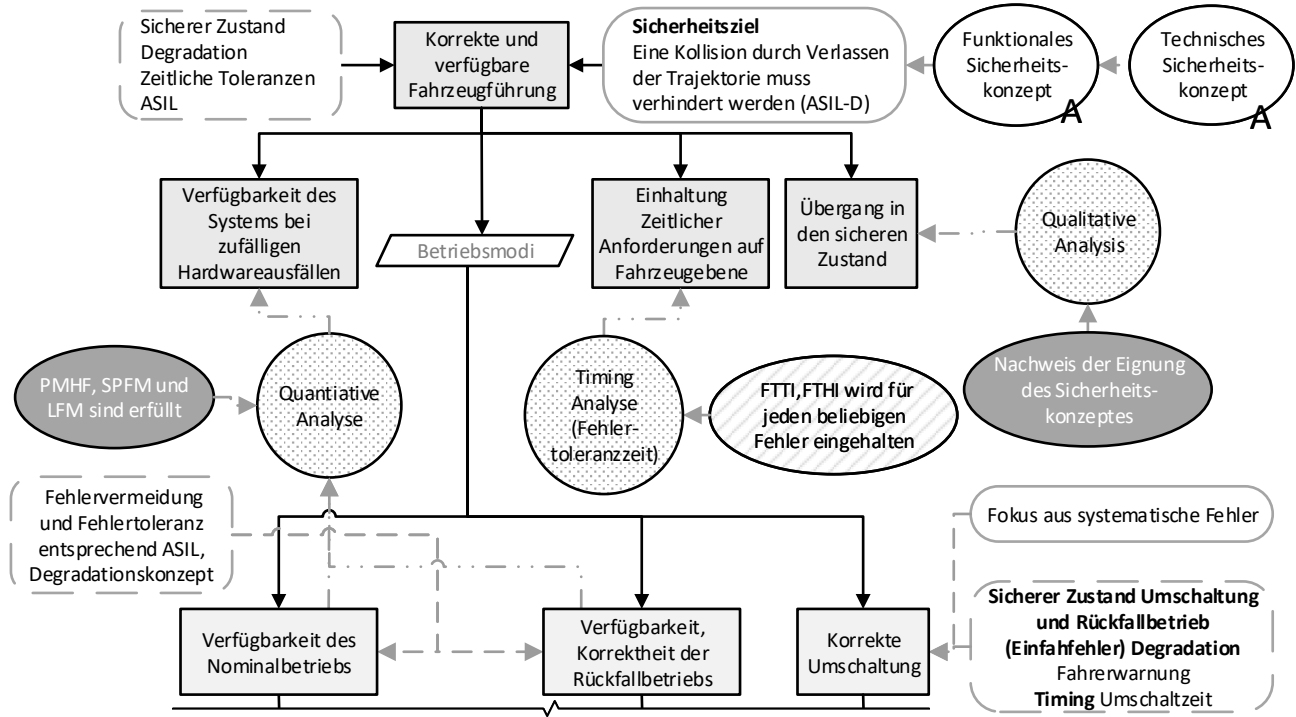


Abbildung 4.9: Sicherheitsargumentation für ein Fail-Operational-System, dargestellt anhand einer Goal Structuring Notation (Teil 1)

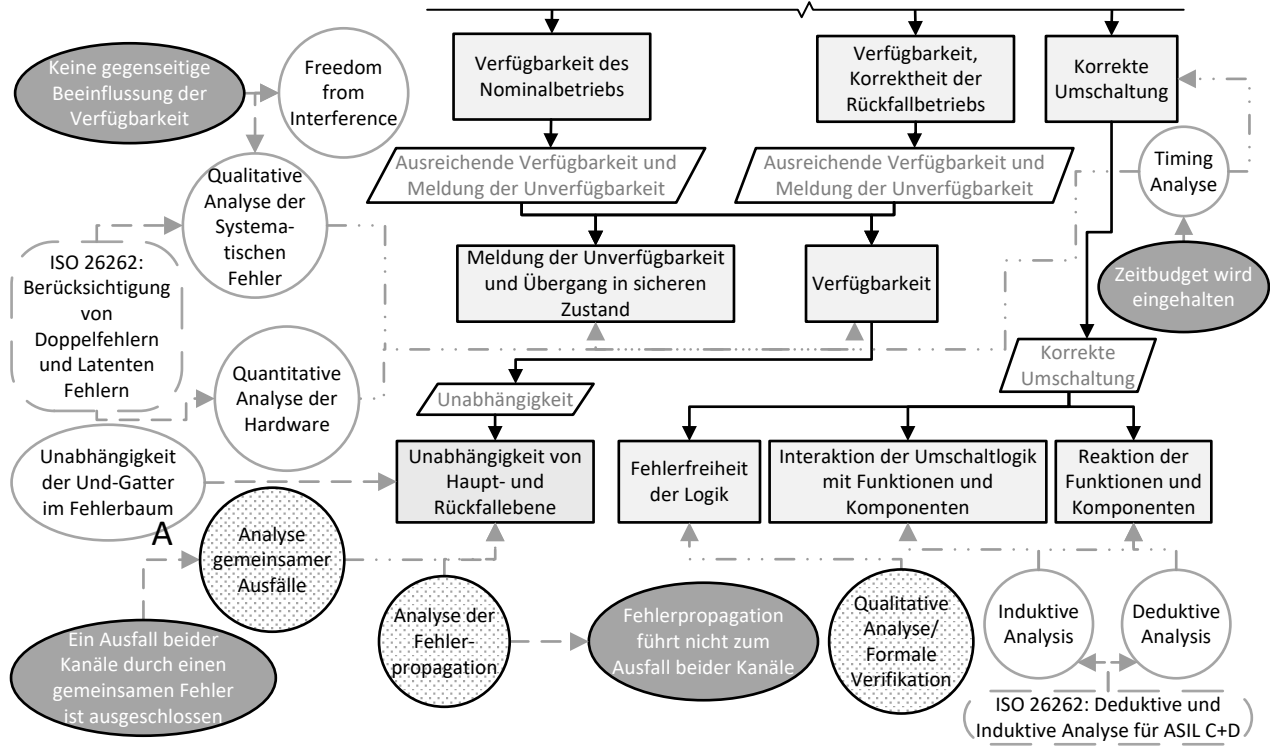


Abbildung 4.10: Sicherheitsargumentation für ein Fail-Operational-System, dargestellt anhand einer Goal Structuring Notation (Teil 2)

Abbildung 4.9 zeigt dabei die Ziele, welche sich auf der Systemebene direkt aus dem Sicherheitsziel ergeben. Aus den Sicherheitszielen wird durch Allokationen auf die Architektur das funktionale und technische Sicherheitskonzept abgeleitet. Auf Darstellung der entsprechenden Relationen wird aus Übersichtsgründen verzichtet. Um die Verfügbarkeit des Systems und ein ausreichend geringes Risiko bezüglich Sicherheitszielverletzungen durch zufällige Ausfälle von Hardwareelementen nachzuweisen ist eine quantitative Analyse notwendig (vgl. Abbildung 4.9 links). Ziel dieser ist der Nachweis von Metriken entsprechend der ISO 26262, wobei für Fail-Operational-Systeme insbesondere die Probabilistic Metric of Random Hardware Faults (PMHF) als Maß für die mittlere Ausfallwahrscheinlichkeit relevant ist. Hierfür sind auf Systemebene, neben der Systemarchitektur und dem Degradationskonzept, Fehlerraten und Diagnoseabdeckungen der einzelnen Komponenten notwendig. Der Nachweis des Übergangs in den sicheren Zustand erfolgt mittels einer qualitativen Analyse auf Systemebene, mit der die Eignung des Degradationskonzeptes analysiert wird (vgl. Abbildung 4.9 rechts). Eine Fehlerreaktion respektive Degradation muss innerhalb spezifizierter zeitlicher Toleranzen erfolgen, weswegen eine Timing Analyse notwendig ist (vgl. Abbildung 4.9 mittig). Sowohl die Timing Analyse als auch die Analyse des Sicherheitskonzeptes dienen auf Systemebene der Konsolidierung der Einzelbeträge der Architekturelemente, wodurch die Verifikation des Sicherheitskonzeptes erfolgt.

Abbildung 4.10 zeigt die Argumentation auf System- aber auch Subsystem-Ebene unter Berücksichtigung der einzelnen Betriebsmodi. Für die Betriebsmodi spezifische Aspekte des Degradationskonzeptes sind, wie in Abbildung 4.8 beschrieben, die Verfügbarkeit und die Reaktion auf eine Unverfügbarkeit in Form der Meldung und des Übergangs in den jeweiligen sicheren Zustand. Auf Komponentenebenen ist daher sowohl eine quantitative als auch qualitative Analyse notwendig (vgl. Abbildung 4.10 links). Deren Ergebnisse stellen notwendige Eingangsgrößen, wie Ausfallraten, für die entsprechenden Analysen auf Systemebene dar. Entsprechend der ISO 26262 müssen die Analysen sowohl latente Fehler als auch Doppelfehler berücksichtigen. Im Rahmen der qualitativen Analyse muss außerdem die Freiheit von Beeinträchtigung (engl.

freedom-from-interference) berücksichtigt werden. Das bedeutet, Fehler nicht systemrelevanter Komponenten mit geringerer Integrität dürfen die Verfügbarkeit des Fail-Operational-Systems nicht negativ beeinflussen. Die Unabhängigkeit von Nominal- und Rückfallbetrieb muss bezüglich gemeinsamer Ausfälle (engl. common-cause failures) und der Fehlerpropagation (engl. cascading failures) nachgewiesen werden (vgl. Abbildung 4.10 links unten). Die Umschaltlogik wird qualitativ analysiert, da quantitative Aspekte bereits auf der Systemebene berücksichtigt werden (vgl. Abbildung 4.10 rechts). Die Umschaltung kann dabei, mit Ausnahme der Verifikation der Interaktion, nur im Verbund vollständig analysiert werden, da das Umschaltkonzept aus verteilten und gekoppelten Zustandsautomaten besteht. Die Interaktion der Komponenten mit der Umschaltlogik sowie deren Reaktion kann auf Komponentenebene erfolgen. Die ISO 26262 schlägt sowohl deduktive als auch induktive Analysen und formale Methoden für Integritäten gemäß ASIL C und ASIL D vor. Der Nachweis der korrekten Umschaltung ist dabei auch für die Analyse des Sicherheitskonzeptes auf Systemebene relevant. Ebenso werden einzelne Zeitbudgets auf Komponentenebene, Diagnose- und Reaktionszeiten, analysiert und auf Systemebene konsolidiert.

Zusammenfassend sind daher auf Systemebene folgende Analysen notwendig und durch Abdeckung der Nachweisziele hinreichend für einen Sicherheitsnachweis nach ISO 26262:

- Quantitative Analyse zum Nachweis eines ausreichend geringen Risikos durch zufällige Fehler,
- Analyse des Degradationskonzeptes und des entsprechenden Übergangs in den sicheren Zustand,
- Analyse des Timings zum Nachweis der Einhaltung der Fehlertoleranzzeit,
- Analyse gemeinsamer Ausfälle und Analyse der Fehlerpropagation zum Nachweis der Unabhängigkeit und
- Nachweis der Korrektheit der Umschaltlogik.

Nachweise für die einzelnen Punkte werden in den folgenden Kapiteln adressiert. Zielwerte für Toleranzzeiten und Ausfallwahrscheinlichkeiten ergeben dabei zum einen aus der Norm ISO 26262 und zum anderen aus den Sicherheitszielen beziehungsweise dem Sicherheitskonzept.

Das Vorgehen und die Argumentation wurden in diesem Kapitel anhand konkreten Systems gezeigt, dennoch ist es analog auf andere Fail-Operational-Systeme anwendbar. Die Industrienorm ISO 26262 stellt eine Vorgabe für alle Fahrzeugsysteme dar, weswegen die Anforderungen und sicherheitsrelevanten Attribute auch für andere Konzepte und Architekturen gültig sind. Auch im Kontext anderer Normen ist das Vorgehen anwendbar. Das Vorgehen zur Identifikation der Nachweisziele, basierend auf funktionalen Beschreibungen und Fehlermodellen, stellt ein etabliertes Vorgehen (vgl. Kapitel 3.1.2) und wurde bereits für verschiedene Systeme publiziert. Die Auswahl der Beschreibungsmodelle ist dabei jedoch system- und insbesondere konzeptspezifisch, wobei Zustandsbeschreibungen und Fehlerbaummodelle für die in der Literatur präsentierten Systeme eine geeignete Beschreibungsform darstellen. Die oben identifizierten Analysen ergeben sich dabei für jedes Fail-Operational-Fahrzeugsystem mit redundanten Strukturen. Der Umfang der einzelnen Analysen, im Falle der Analysen gemeinsamer Ausfälle beispielsweise welche Komponenten voneinander unabhängig sein müssen, stellt jedoch ein systemspezifisches Ergebnis der Argumentation dar. Systeme, die auf die Umschaltung kompletter Kanäle verzichten, und nur zwischen redundanten Komponenten schalten, erfordern keinen Unabhängigkeitsnachweis der kompletten Kanäle, sondern eben nur dieser Komponenten. Dies gilt in Konsequenz auch für die Auswahl geeigneter Analysenmethoden. Die Unterscheidung von quantitativer und qualitativer Analysen ist jedoch durch die sicherheitsrelevanten Attribute bestimmt und daher durch die ISO 26262 definiert.



# KAPITEL 5

## ANALYSE GEMEINSAMER AUSFÄLLE BEI FAIL-OPERATIONAL-SYSTEMEN

Die Unabhängigkeit von Nominal- und Rückfallbetrieb bezüglich gemeinsamer Ausfälle ist gemäß Kapitel 4.4 eine notwendige Voraussetzung für die Funktionale Sicherheit des Fail-Operational-Fahrzeugsystems. Ein Fehler im Nominalbetrieb, der auch zum Ausfall des entsprechenden Rückfallbetriebs führt, verletzt durch den Systemausfall direkt ein Sicherheitsziel. Daher müssen entsprechende Fehlerursachen identifiziert und ein Nachweis der Unabhängigkeit und funktionalen Sicherheit geführt werden.

Im Industriestandard ISO 26262, wie auch in der Literatur, werden Vorgehensweisen zur Analyse gemeinsamer Ausfälle präsentiert (vgl. Kap. 3.2). Die generischen Vorgehen berücksichtigen Anforderungen einer Anwendung in Fail-Operational-Systemen nicht, da deren Fokus die Analyse einzelner Dekompositionen mit meist begrenztem Umfang ist. In Fail-Operational-

Fahrzeugführungssystemen ist dagegen mitunter die Unabhängigkeit von kompletten Wirkketten notwendig. Darüber hinaus erfordert eine verteilte Entwicklung im Unternehmensumfeld die Berücksichtigung von Randbedingungen, wie einer externen Entwicklung, Unternehmensstandards oder einer Projektplanung, welche in der Literatur bisher nicht adressiert wurde.

Um notwendige Anforderungen an eine Analyse gemeinsamer Ausfälle bei Fail-Operational-Systemen zu identifizieren, wurde eine Studie mit semi-strukturierten Interviews durchgeführt (vgl. Kapitel 5.1). Die Auswertung und Identifikation der Anforderungen erfolgt mittels der Methode des konstanten Vergleichs. Basierend auf den Ergebnissen der Studie und der Literatur wird das bestehende Vorgehen in Kapitel 5.2 erweitert und in Kapitel 5.3 anhand eines Anwendungsbeispiels validiert. Darüber hinaus wird die Anwendung von Quantifizierungen gemeinsamer Ausfälle in der Automobilindustrie in Kapitel 5.4 diskutiert.

## 5.1 Studie zur Identifikation von Anforderungen an die Analyse gemeinsamer Ausfälle

In diesem Kapitel wird eine explorative Studie zur Identifikation der Anforderungen an eine Analyse gemeinsamer Ausfälle für ein Fail-Operational-System vorgestellt. Hierzu werden zuerst die Forschungsfragen abgeleitet. Ausgehend davon erfolgt die Beschreibung des Aufbaus, der Durchführung und der Auswertung der Studie. Im Anschluss werden die Ergebnisse präsentiert und erläutert sowie deren Validität diskutiert.

### 5.1.1 Ziel der Studie

Das Ziel der Studie ist die Identifikation der Anforderungen, welche für ein anwendbares Vorgehen für die Analyse gemeinsamer Ausfälle von Fail-Operational-Fahrzeugsystemen gemäß ISO 26262 im Rahmen eines Fahrzeugentwicklungsprozesses relevant sind.



Abgeleitet aus dem Forschungsziel ergeben sich folgende Forschungsfragen:

1. Welche Stakeholder stellen Anforderungen an eine Common-Cause Analyse?
2. Welche Anforderungen stellen die einzelnen Stakeholder an ein Vorgehen?
3. Welche Anforderungen stellen die einzelnen Stakeholder an die Durchführung und die Dokumentation?

Im Rahmen der Studie muss sichergestellt werden, dass alle Stakeholder berücksichtigt werden. Daher dient die erste Forschungsfrage deren Identifikation. Im nächsten Schritt, berücksichtigt in der zweiten Forschungsfrage, sollen im Rahmen der Studie die relevanten Anforderungen an das Vorgehen identifiziert werden. Da nicht alle Anforderungen an eine Analyse durch das Vorgehen abgedeckt werden können, sondern auch bei der Durchführung und Dokumentation berücksichtigt werden müssen, wird auch die Identifikation solcher Anforderungen in den Forschungsfragen adressiert.

### 5.1.2 Aufbau, Durchführung und Auswertung der Studie

Die Durchführung der Studie erfolgte im Umfeld des Entwicklungsbereichs eines Automobilherstellers, der BMW AG. Orientiert an den Arbeiten von Niedermaier et al. [NKF19], Seaman [Sea99] und Mayring [May14] wurden semi-strukturierte Interviews durchgeführt. Offene Fragen ermöglichen dabei eine vollständige Identifikation von Anforderungen über alle relevanten Stakeholder hinweg. Zudem kann bei semi-strukturierten Interviews durch die Interaktion zwischen Interviewer und den Befragten sichergestellt werden, dass der Kontext verstanden wird und die Fragen vollständig beantwortet werden [Sea99].

Im Rahmen der Studie wurden neun Interviews mit durchschnittlich vierzig Minuten Dauer im Zeitraum von Juni bis August 2019 in einem direkten

Dialog durchgeführt. Um heterogene Sichtweisen zu berücksichtigen wurden Interviewpartner von einem OEM, einem Zulieferer und Dienstleistern befragt. Um die Einflüsse projektspezifischer Randbedingungen zu minimieren, wurden außerdem Personen aus verschiedenen Projekten, Domänen und Verantwortlichkeiten ausgewählt. Die initiale Identifikation erfolgt auf Basis von Prozessmodellen und Rollenbeschreibungen des OEMs und der ISO 26262 [Int18]. Diese sind ein Sicherheitsverantwortliche, ein Sicherheitsanalyst und der Projektleiter. Weitere Rollen wurde im Rahmen der Befragungen identifiziert und die Menge der Studienteilnehmer entsprechend erweitert. Bei deren Auswahl wurde außerdem darauf geachtet, dass die Befragten bereits grundlegende Erfahrungen mit den Analysen gemeinsamer Ausfälle aufweisen und an der Entwicklung von Fail-Operational-Systemen für Fahrzeuge beteiligt sind, um gleiche Randbedingungen zu schaffen.

Mit Hilfe eines Interview-Guides, wie beispielsweise von Mayring [May14] vorgeschlagen, konnte sichergestellt werden, dass im Rahmen des Interviews alle Forschungsfragen adressiert wurden und die Reihenfolge der Fragen einheitlich war, um vergleichbare Randbedingungen zu erzeugen. Dieser besteht aus offenen Fragen und wurde vom Interviewer genutzt, um im Rahmen des Interviews Rückmeldungen zu dokumentieren. Die Fragen waren den Teilnehmern vorab bekannt. Zu Beginn wurde nach den Zielen der Analyse gefragt. Die jeweilige Sicht der Rolle wurde nicht explizit angesprochen, da diese der Sichtweise implizit zugrunde liegt. Danach erfolgte eine kurze Vorstellung der wichtigsten Schritte einer solchen Analyse nach der ISO 26262 seitens des Interviewers entsprechend Kapitel 3.2. Im Anschluss wurden die Forschungsfragen als offene Fragen adressiert und Rückfragen bei Bedarf beantwortet. Am Ende des Interviews wurden die Notizen durchgesprochen, woraufhin die Befragten diese gegebenenfalls korrigieren und entsprechend der Relevanz quantitativ bewerten konnten.

Die Auswertung der Studie erfolgte nach dem von Seaman [Sea99] beschriebenen Vorgehen des konstanten Vergleichs. Dabei werden Schlagwörter und Begriffe aus Antworten identifiziert. Diese wiederum wurden konsolidiert

und gruppiert, um Muster und Themencluster zu identifizieren. Im Rahmen der Auswertung wurden zwei Ebenen des Clusterings genutzt. Auf oberster Ebene orientieren sich diese an den Forschungsfragen, darunter wurden ähnliche Nennungen zusammengefasst. Die Auswertung und Anpassung des bestehenden Clusterings erfolgt, wie von Seaman vorgeschlagen, nach jedem Interview. Die konsolidierten Anforderungen wurden nach dem letzten Interview allen Befragten gezeigt, um zu verifizieren, dass ihre Nennungen durch die konsolidierten Anforderungen berücksichtigt werden.

### 5.1.3 Ergebnisse der Studie

Im Folgenden werden die Ergebnisse der Studie vorgestellt. Dabei erfolgt zuerst die Präsentation der einzelnen Stakeholder, entsprechend der Identifikation anhand der Studie. Im Anschluss werden die genannten Ziele und identifizierten Anforderungen an ein Vorgehen und die Dokumentation erläutert.

Abbildung 5.1 zeigt die Nennungen der Stakeholder.

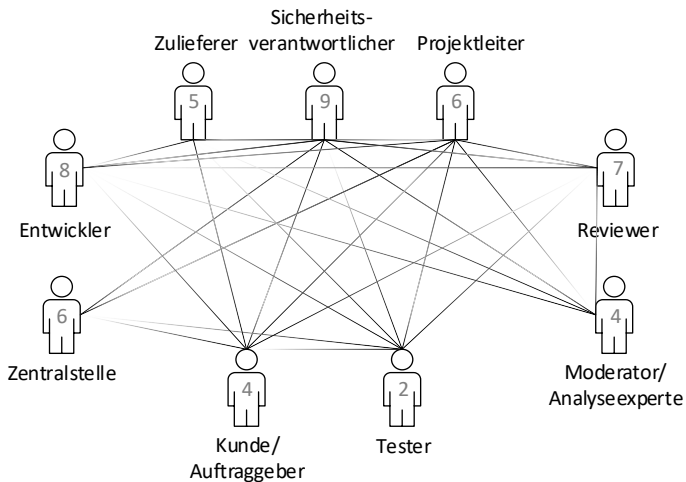


Abbildung 5.1: Identifizierte Stakeholder der Common-Cause Analyse

Knoten stellen die identifizierten Stakeholder dar. Kanten zeigen die jeweiligen Nennungen. Die Nennung beispielsweise des Entwicklers durch den Kunden ist durch eine Kante vom Kunden zum Entwickler, hell zu dunkel, angetragen. Die Anzahl der jeweiligen Nennungen ist zusätzlich an den Knoten angetragen. Alle Stakeholder wurden mehrmals genannt. Der Sicherheitsverantwortliche wurde von allen Befragten als relevant identifiziert. Die identifizierten Stakeholder werden im Folgenden anhand der Verantwortung für die Sicherheitsaktivitäten kurz vorgestellt. Die Beschreibung basiert auf den Interviews, der Norm ISO 26262 und den Rollenbeschreibungen des OEMs.

Die Aufgaben des Sicherheitsverantwortlichen umfassen die Planung und Kontrolle von Aktivitäten zum Erreichen funktionaler Sicherheit. Außerdem ist er der Ansprechpartner für die Zentralstelle bei Reviews der Arbeitsprodukte.

Der Projektleiter trägt die Verantwortung für die Durchführung der Sicherheitsaktivitäten und deren Konformität mit dem Industriestandard ISO 26262.

Reviewer prüfen und bestätigen diese Konformität und gegebenenfalls die Erfüllung von Unternehmensstandards im Rahmen von Assessments. Dabei wird zwischen inhaltlichen und formalen Reviews unterschieden [Int18].

Die Industrialisierung von Komponenten und Funktionen, gemäß den vereinbarten Anforderungen, wird von Entwicklern verantwortet. Diese sind daher auch die Experten für technische Zusammenhänge und die Implementierung. Moderatoren und Experten für Sicherheitsanalysen definieren die Methodik und leiten die Analyse. Der Testverantwortliche stellt die zeitliche und inhaltliche Planung, Abarbeitung und Auswertung der Testfälle sicher.

Als Kunde oder Auftraggeber wurde ein Stakeholder verstanden, der Abnehmer des jeweiligen Umfangs, an den Analysen aber nicht unmittelbar beteiligt ist. Dieser kann unternehmensintern oder extern sein.

Unternehmen setzen Zentralstellen ein, um identische Standards für die Sicherheitsbetrachtung und Interpretation der ISO 26262 unternehmensweit sicherstellen. Die Standardisierung erfolgt zum einen durch Vorgaben und zum anderen durch die Inspektion der Arbeitsprodukte. Die Zentralstelle

kann daher auch eine Reviewfunktion beinhalten.

Der letzte, genannte Stakeholder ist der Zulieferer. Dieser stellt Umfänge entsprechend einer Beauftragung bereit. Da diese im gesamtheitlichen Sicherheitskonzept berücksichtigt werden müssen, ist es erforderlich, dass die Schnittstellen der beauftragten Umfänge und Sicherheitsprodukte definiert werden.

Das Zielbild der Analyse gemeinsamer Ausfälle im Fail-Operational-Kontext ist über alle Befragten hinweg homogen formuliert worden. Abbildung 5.2 zeigt die genannten Ziele.

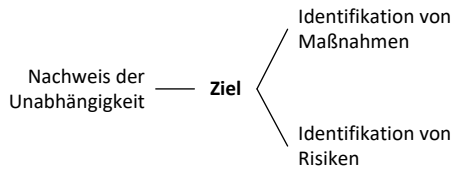


Abbildung 5.2: Identifizierte Ziele der Analyse gemeinsamer Ausfälle

Nach Meinung der Befragten dient die Analyse zum einen dem Nachweis der Unabhängigkeit und zum anderen der Identifikation von Risiken beziehungsweise Schwachstellen für einen gemeinsamen Ausfall und der Ableitung entsprechender Maßnahmen. Die Ziele decken sich dabei mit der ISO 26262 [Int18].

Die Anforderungen bezüglich des Vorgehens und deren Abhängigkeiten werden in Abbildung 5.3 gezeigt. Dabei wurden Anforderungen teilweise detaillierter formuliert, daher zeigt die Abbildung zwei hierarchische Ebenen. Beziehungen, Detaillierungen und Abhängigkeiten werden durch Kanten dargestellt.

Ein geeignetes Vorgehen muss gemäß der Studie ein definiertes Ziel verfolgen. Die Unabhängigkeitsbeziehungen und damit der Umfang und die Schnittstellen müssen ebenso eindeutig definiert werden. Letzteres ist insbesondere für Lieferanten von Bedeutung. Auch stellt die Erfüllung von

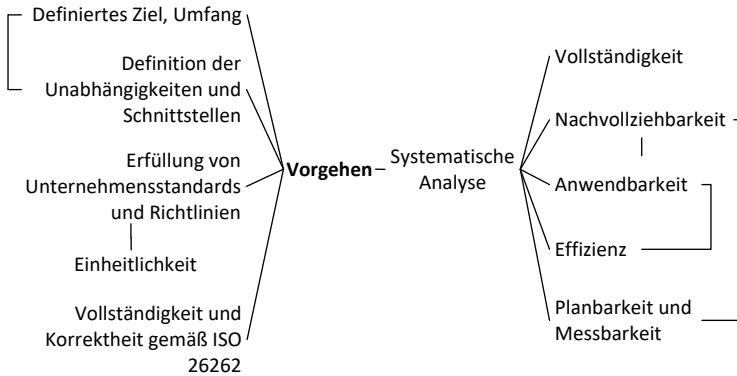


Abbildung 5.3: Identifizierte Anforderungen an das Vorgehen bei der Analyse gemeinsamer Ausfälle

Unternehmensstandards und Richtlinien eine Anforderung an das Vorgehen dar. Dies ist notwendig, um Vorgaben der Zentralstelle zu erfüllen und dient dazu, eine gleichbleibende Qualität und einen gleichbleibenden Detaillierungsgrad über verschiedene Projekte und Produkte hinweg zu gewährleisten. Das Vorgehen muss zusätzlich die Vorgaben der ISO 26262 vollständig erfüllen. Das bedeutet insbesondere, die Analyse muss systematische und zufällige Fehler berücksichtigen und der Detaillierungsgrad ausreichend für eine Argumentation der funktionalen Sicherheit sein. Die Analyse muss die in Kapitel 3.2 genannten Aspekte behandeln. Auch muss die Bewertung der Risiken vollständig erfolgen. Kriterien, die bei der Bewertung der Unabhängigkeit und der abgeleiteten Maßnahmen berücksichtigt werden können sind die Integrität, der Grad der Unabhängigkeit, die Komplexität und die Technologie sowie die Diversität der Umweltbedingungen. Darüber hinaus wurden Anforderungen identifiziert, welche unter dem Begriff systematischen Analyse eingeordnet wurden. Die Vollständigkeit der Analyse bezüglich des Items oder Betrachtungsumfangs stellt eine solche Anforderung dar. Hierfür müssen auch Änderungen in späteren Entwicklungsphasen in der Analyse berücksichtigt werden. Das Vorgehen soll nachvollziehbar sein, das bedeutet, dass insbesondere Bewertungen

und Maßnahmen verständlich sein müssen. Das wiederum begünstigt auch die Anwendbarkeit. Bezüglich der Anwendbarkeit wurden darüber hinaus klare Prüf- und Bewertungskriterien gefordert. Außerdem soll die Analyse möglichst effizient durchführbar sein, was bedeutet, dass Doppelarbeit durch mehrfache Analysen oder aufwändige Überarbeitungen minimiert und weitere Bewertungsmaßnahmen, wie Testen, Simulation aber auch Sicherheitsanalysen eingesetzt werden, sofern diese einen Mehrwert bieten. Für ein Projekt ist außerdem die Planbarkeit und damit einhergehend die Messbarkeit wichtig. Kritische und umfangreiche Änderungen am System sollen möglichst frühzeitig identifiziert werden.

Neben Anforderungen an das Vorgehen wurden im Rahmen der Studie Anforderungen an die Dokumentation der Analyse identifiziert. Dies ist notwendig, da durch ein Vorgehen ohne Berücksichtigung der Umsetzung nicht alle Anforderungen an eine Analyse abgedeckt werden können. Zusätzlich sind für die Umsetzung der Anforderungen bezüglich des Vorgehens Aspekte der Dokumentation zu berücksichtigen. Abbildung 5.4 zeigt die identifizierten Anforderungen.

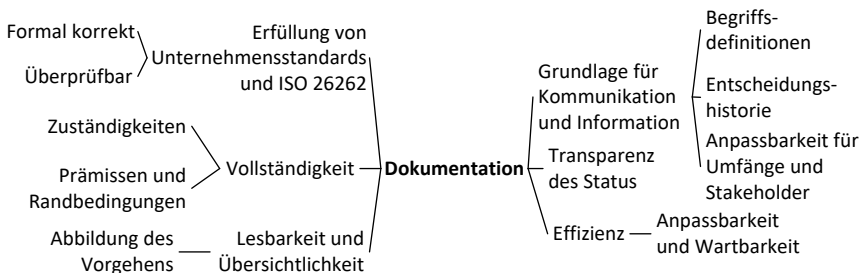


Abbildung 5.4: Identifizierte Anforderungen an die Dokumentation bei der Analyse gemeinsamer Ausfälle

Seitens der ISO 26262 und auch durch Unternehmensstandards gibt es Vorgaben an die Dokumentation, die zu beachten sind. Diese beziehen sich auf formale Aspekte, wie die Nennung von Verantwortlichen, der Zielsetzung des Dokumentes und die Überprüfbarkeit im Sinne der Vollständigkeit und

Nachvollziehbarkeit, zum Beispiel durch eine klare Strukturierung und die Angabe von Referenzen. Durch Vorlagen sichern Unternehmensstandards eine Konformität mit der ISO 26262 und erleichtern darüber hinaus Reviews und die Vergleichbarkeit von Analysen durch eine einheitliche Struktur der Dokumente. Als vollständig wurde ein Dokument im Rahmen der Studie bezeichnet, wenn es in sich geschlossen ist. Das bedeutet, dass Prämissen, Randbedingungen und Referenzen dokumentiert sind. Für den Projektleiter sind darüber hinaus Zuständigkeiten eine wichtige Information. Die Dokumentation soll lesbar und übersichtlich sein. Unter diesem Aspekt wurde mehrmals genannt, dass das Vorgehen auch im Dokument abgebildet sein sollte. Um als Kommunikationsmittel nutzbar zu sein, müssen relevante Begriffe definiert werden. Zudem ist eine Historie der einzelnen Entscheidungen und Bewertungen erforderlich, die Umfänge sollen projekt- und produktspezifisch anpassbar und die Sichten wiederum exportierbar sein, um auch die Kommunikation mit Lieferanten zu ermöglichen. Für eine Planung ist der Status der Analyse relevant. Dabei sollen kritische Punkte und Risiken, transparent werden. Alle Stakeholder, die direkt an der Analyse arbeiten, forderten eine Unterstützung einer effizienten Arbeitsweise. Darunter werden eine einfache Anpassbarkeit und Wartbarkeit bei Änderungen oder Weiterverwendung verstanden.

#### 5.1.4 Zusammenfassung und Diskussion der Ergebnisse

Die Gewichtung der Anforderungen variierte abhängig von der jeweiligen Rolle deutlich, sodass im Allgemeinen keine Aussage über die Relevanz der Anforderungen möglich war. Einzig solche, welche die Erfüllung von Standards, der ISO 26262 oder Unternehmensrichtlinien erfordern, hatten für alle Befragten einen hohen Stellenwert und wurden als eine unausweichliche Bedingung betrachtet. Darüber hinaus haben alle Befragten Anforderungen an das strukturierte Vorgehen gestellt, wenngleich die genannten Aspekte variierten. Insbesondere die Vollständigkeit wurde dabei von einem Großteil aufgeführt. Bezüglich der Dokumentation waren die Nachvollziehbarkeit und die Wartbarkeit häufig genannte Anforderungen. Die Mehrfachnennung



aller Anforderungen zeigt deren Legitimität und die umfassende Betrachtung der Thematik in den Interviews.

Die Validität der Studie wird analog zu den Kriterien in [NKFW19] und [WRH+12] analysiert und anhand derer die Vertrauenswürdigkeit und Belastbarkeit der Ergebnisse diskutiert. Niedermaier et al. [NKFW19] und Wohlin [WRH+12] unterscheiden zwischen der Konstrukt-Validität, der internen Validität, der externen Validität und der Zuverlässigkeit.

Die Konstrukt-Validität adressiert Abweichungen zwischen dem Forschungsziel und dem formulierten Fragen, welche den Forschungsfragen entsprechen [WRH+12]. Um die Übereinstimmung sicher zu stellen wurden der Forschungskontext und die Forschungsziele zu Beginn jedes Interviews kommuniziert. Darüber hinaus ermöglichte der direkte Dialog beidseitige Rückfragen bei Unklarheiten, sodass unterschiedliche Interpretationsmöglichkeiten minimiert wurden.

Die interne Validität behandelt die Kausalität der Ergebnisse und den untersuchten Faktoren, das heißt inwiefern die Studie frei von systematischen Fehlern ist [RH09]. Da die Befragung in laufenden Projekten durchgeführt wurde, könnten Befragte darin eine gewisse Kontrolle sehen und die Antworten dementsprechend geben. Dem wurde zum einen durch Anonymität nach außen, und zum anderen durch die Erklärung des Forschungshintergrundes und der Trennung vom Projektgeschäft entgegengewirkt.

Die externe Validität behandelt die Generalisierbarkeit der Studie [WRH+12]. Da die Durchführung der Studie im Umfeld eines OEMs erfolgte, wurde darauf geachtet Stakeholder aus verschiedenen Projekten, Umfängen und Unternehmen zu befragen, um generalisierbare Ergebnisse zu erzielen. Dabei verfügten alle über Erfahrungen mit Analysen gemeinsamer Ausfälle. Die Fragen wurden außerdem vorab kommuniziert, um unabhängiger von spontanen Nennungen und den Randbedingungen eines Interviews zu sein.

Die Zuverlässigkeit betrifft den individuellen Einfluss des Forschers [WRH+12]. Während des Interviews wurden Rückfragen gestellt, um

das Verständnis der Aussagen und Intentionen zu sichern. Die Antworten und zusätzliche Erklärungen wurden notiert und am Ende des Interviews nochmals mit dem Befragten abgestimmt. Die Interviews und die Auswertung wurden von derselben Person durchgeführt. Ein systematisches Vorgehen zur Interpretation und Ableitung von Clustern, die Methode des konstanten Vergleiches nach [Sea99], ermöglichte außerdem die Minimierung individueller Einflüsse bei der Auswertung, insbesondere weil die aufbereiteten Ergebnisse durch die Befragten nochmals auf Vollständigkeit geprüft wurden.

## 5.2 Erweiterung der Vorgehensweise und Ableitung eines Datenmodells

Die identifizierten Anforderungen werden durch die Erweiterung des, in der Literatur und der ISO 26262 vorgestellten, generischen Vorgehens (vgl. Kapitel 3.2), berücksichtigt. Zusätzlich wird zur vollständigen Abdeckung der Anforderungen ein Datenmodell abgeleitet, welches die Vorgehensweise unterstützt und die Anforderungen an die Dokumentation berücksichtigt. Das Resultat stellt eine Methodik für die Analyse gemeinsamer Ausfälle dar, welche den Anforderungen der Entwicklung von Fail-Operational-Fahrzeugsystemen im Industriefeld gerecht wird.

Das erweiterte Vorgehen zeigt Abbildung 5.5 in Form eines Ablaufdiagramms mit den relevanten Schritten. Auf die detaillierte Auflistung aller Arbeitsprodukte wurde aus Gründen der Übersicht verzichtet.

Die **Identifikation der notwendigen Unabhängigkeiten** erfolgt auf Basis einer deduktiven Systemanalyse wie der Fehlerbaumanalyse (FTA). Die Umschaltlogik definiert dabei Systemkonfigurationen und damit die Menge der tolerierten Fehler, welche nicht direkt zu einem Ausfall führen. In Kombination mit der Systemarchitektur ergeben sich damit die notwendigerweise unabhängigen Umfänge. Dabei sind die Sicherheitsziele und

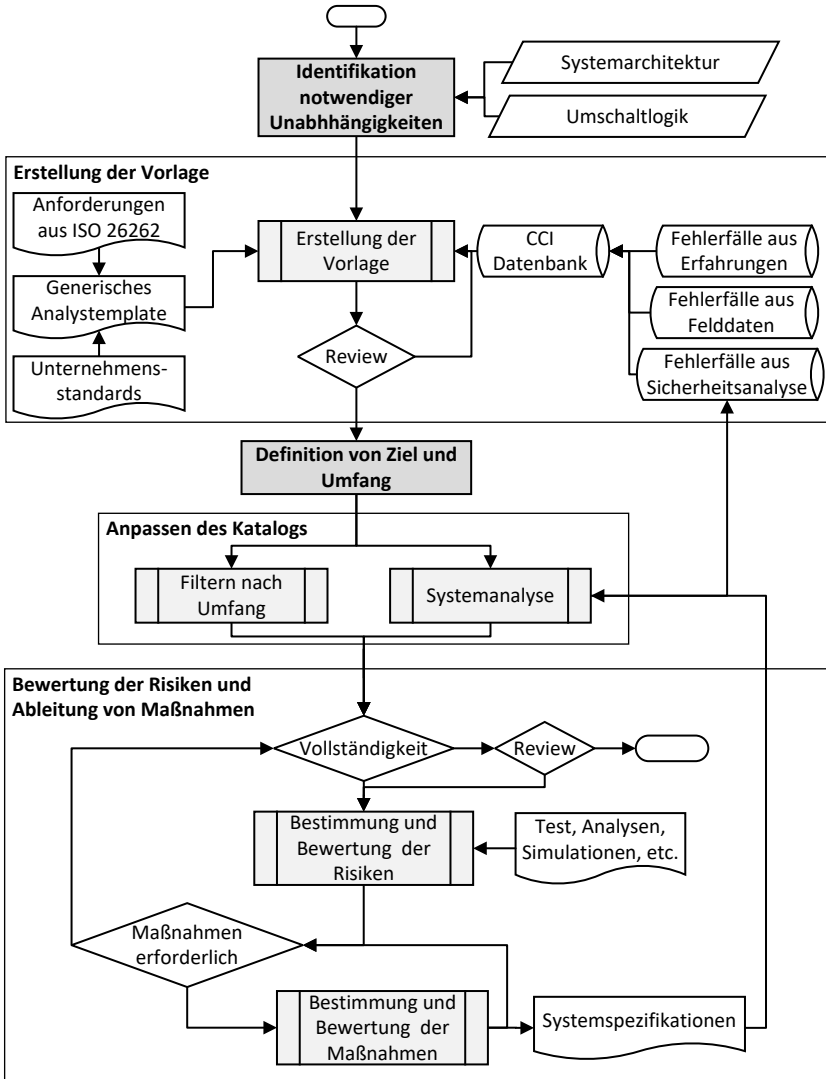


Abbildung 5.5: Erweitertes Vorgehen für die Analyse gemeinsamer Ausfälle für Fail-Operational-Systeme

Fehlermodi, für die die Unabhängigkeit notwendig ist, relevant. Im Falle des hochautomatisierten Fahrens ist eine Unverfügbarkeit kritisch, weswegen ein gleichzeitiges Abschalten der redundanten Kanäle verhindert werden muss.

Den nächsten Schritt, analog zur Literatur (vgl. Kapitel 3.2), stellt die Identifikation der Abhängigkeiten und Kopplungen dar. Dieser wird im erweiterten Vorgehen in zwei Einzelschritte, einen generischen und einen projektspezifischen Schritt, aufgeteilt. Der generische Teil dient der Erstellung und Aktualisierung einer Vorlage. Dieser Schritt kann unternehmensweit initial durchgeführt und laufend aktualisiert werden und stellt daher keinen projektspezifischen Aufwand dar.

Zur **Erstellung der Vorlage** wird, vergleichbar zur Luft und Raumfahrt (vgl. Common Mode Analyse [Bal15, SAE96]), ein Katalog an Initiatoren gemeinsamer Ausfälle erstellt. Dieser wird durch Erfahrungswerte, Fehlerfälle aus Felddaten und identifizierten Fehlerfällen aus Sicherheitsanalysen befüllt. Die Initiatoren dienen einer initialen Befüllung einer Vorlage, in der auch die Vorgaben der ISO 26262 und des Unternehmensstandards berücksichtigt werden. Auf die Struktur des Datenmodells wird in Abbildung 5.6 eingegangen. Ein Review stellt die Einhaltung der Standards und der ISO 26262 sicher, vereinfacht spätere Inspektionen und kann nach einer initialen Freigabe auf Basis der Änderungen durchgeführt werden. Darüber hinaus wird unternehmensweit eine identische Ausgangsbasis und ein gleicher Detaillierungsgrad der Analyse erzeugt, da auch ein Rückfluss weiterer Initiatoren aus der Analyse und Kommunikation an andere Projekte berücksichtigt werden kann.

Im Anschluss erfolgt der projektspezifische Teil der Analyse. In diesem ist es zu Beginn wichtig **Ziel und Umfang der Analyse** klar zu **definieren** und das Vorgehen zu kommunizieren. Formale Elemente wie Verantwortliche, Schnittstellen, Ansprechpartner und Version sollten ebenso wie wichtige Begriffsdefinitionen dokumentiert werden.

Im nächsten Schritt erfolgt das **Anpassen des generischen Katalogs** entsprechend des Ziels. Das bedeutet Umfänge ohne Relevanz werden gestrichen (vgl. [Wan17]). Die Form der Dokumentation muss Anpassungen dabei

effizient ermöglichen. Basierend auf einer Systemanalyse, wie von [Sch16], beispielsweise unter Verwendung des Abhängigkeitsmodells in Abbildung 3.1 und [Eri16] beschrieben, wird der Katalog erweitert. Die Systemanalyse ist insbesondere für neue Entwicklungsumfänge relevant und sichert die Vollständigkeit. Identifizierte Erweiterungen werden darüber hinaus in die Datenbank der Initiatoren eingepflegt, um die Erkenntnisse projektübergreifend zur Verfügung zu stellen.

Der nächste Schritt ist die eigentliche Analyse des Systems. Basierend auf der angepassten Liste erfolgt die Bewertung der einzelnen Initiatoren gemeinsamer Ausfälle, die **Bewertung der Risiken und Ableitung von Maßnahmen**. Zur Bewertung können dabei beispielsweise Tests, weitere Analysemethoden, Simulationen oder Experteneinschätzungen genutzt werden, um die Identifikation und Bewertung der gemeinsamen Ausfälle effizient zu gestalten. Die Dokumentation muss diesen Schritt durch die Berücksichtigung klarer Prüfkriterien unterstützen. Das systematische Vorgehen stellt eine Nachvollziehbarkeit und Anwendbarkeit sowie Effizienz sicher. Die Planbarkeit ist durch ein schrittweises Vorgehen gegeben.

Abbildung 5.6 zeigt das Datenmodell anhand eines Klassendiagramms zur Umsetzung der identifizierten Anforderungen, reduziert auf die wesentlichen Aspekte. Formale Aspekte wie eine Versionsverwaltung oder der Titel sind nicht Teil des Datenschemas, da diese nur einmalig definiert werden. Die Datenstruktur und die Umsetzung der identifizierten Anforderungen werden im Folgenden erklärt. Die Basis der Datenstruktur bilden die Entitäten **Koppelfaktoren, Common-Cause Initiatoren, Risiken und Maßnahmen**. Diese entsprechen auch in der Reihenfolge dem Vorgehen aus Abbildung 5.5. Die **Koppelfaktoren** beinhalten die Faktoren nach ISO 26262 beziehungsweise [Sch16] und die Zuordnung dieser zu den Themenbereichen, welche nach ISO 26262 in einer Analyse berücksichtigt werden müssen [Sch16]. Damit wird deren Abdeckung im Rahmen der Analyse sichergestellt. Darüber hinaus werden jedem Eintrag Beispiele für ein besseres Verständnis zugeordnet. Den eigentlichen Katalog der Initiatoren stellt die Entität **Common-Cause Initiatoren** dar. Die Koppelfaktoren werden diesen zugeordnet und in der

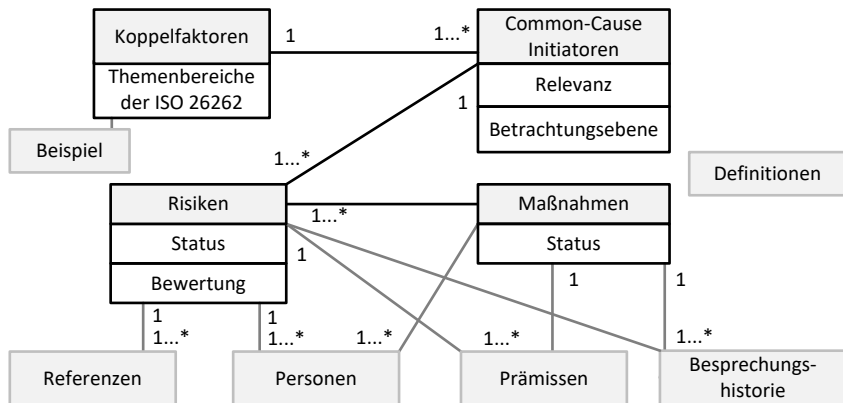


Abbildung 5.6: Datenmodell für die Dokumentation der Analyse gemeinsamer Ausfälle

unternehmensweiten Datenbank abgelegt. Darüber hinaus verfügt jeder Eintrag über ein Attribut der **Betrachtungsebene** und der **Relevanz**. Betrachtungsebenen dienen der effizienten Auswahl. Es wird entsprechend Moestl und Ernst [ME15] zwischen funktionalen, Software-, Hardware- und gegebenenfalls auch Systemumfängen unterschieden. Initiatoren, die nicht dem Betrachtungsumfang entsprechen können über die Relevanz aussortiert und die Analyse damit projektspezifisch angepasst werden. Die Bewertung erfolgt in den Entitäten der **Risiken** und **Maßnahmen**. Hierfür ist das Attribut **Status** notwendig, um diesen transparent darzustellen. Darüber hinaus wurden weitere Entitäten in Form von **Referenzen**, **Prämissen**, **Historie** und **Personen** berücksichtigt. Damit wird eine Vollständigkeit der Daten erzeugt. Für kritische Risiken werden Maßnahmen abgeleitet und diesen zugeordnet. Ebenso werden Begriffsdefinitionen berücksichtigt. Die Datenstrukturierung erlaubt außerdem eine Auswahl von Datensätzen und einen spezifischen Export dieser als Kommunikationsgrundlage.

## 5.3 Validierung des Vorgehens anhand einer Fallstudie

Eine Validierung erbringt den Nachweis, dass das Vorgehen inklusive der Datenstruktur dem Ziel und in diesem Fall den identifizierten Anforderungen entspricht [Gol11]. Hierfür wird eine Fallstudie anhand eines Anwendungsfalls aus industriellen Fahrzeugentwicklung verwendet. Im Folgenden werden zuerst die Rahmenbedingungen und im Anschluss die Durchführung aufgezeigt.

Den gewählten Anwendungsfall stellt eine Regelfunktion für die hochautomatisierte Fahrzeugführung dar. Deren Aufgabe ist die Folgeregulierung einer vorgehenden Trajektorie durch die Bestimmung des Lenkmoments und der Radmomente für den Antrieb und die Bremsen. Die Probanden stellen zwei Sicherheitsverantwortliche des Umfangs dar. Diese wurden gewählt, da in der Studie von dieser Rolle die meisten Anforderungen abgedeckt wurden. Außerdem vertritt der Sicherheitsverantwortliche die Analyse gegenüber dem Zentralteam und dem Projektleiter und ist damit Ansprechpartner für die anderen Stakeholder. Die Validierung erfolgt jeweils getrennt.

Zu Beginn der Validierung wurden die Anforderungen aus Kapitel 5.1.3 anhand der Abbildungen 5.2, 5.3 und 5.4 vorgestellt und erläutert. Im Anschluss erfolgte die Präsentation des Vorgehens aus Abbildung 5.5. Die Datenstruktur wurde anhand eines Datenbanksystems implementiert und die Initiatoren entsprechend bereits abgeschlossener Analysen vorab befüllt. Die Durchführung der Analyse durch die Sicherheitsverantwortlichen erfolgte unter Anleitung, entsprechend des Vorgehens. Abschließend wurde die Erfüllung der einzelnen Anforderungen nochmals verifiziert, womit das Ziel der Analyse validiert wird.

Die Analyseschritte im Rahmen der Validierung für das Fahrzeugführungsregelsystem werden im Folgenden vorgestellt. Im ersten Schritt werden die Unabhängigkeitsanforderungen für das Gesamtsystem, entsprechend dem in Kapitel 2.3 vorgestellten Fail-Operational-System, identifiziert. Die

Umschaltung als Reaktion auf einen Fehler erfolgt diskret zwischen den Kanälen. Im Falle komplexer Degradationskonzepte, wie der Umschaltung als Maßnahmen für Fehlertoleranz, können notwendige Unabhängigkeiten unter Umständen nicht direkt aus der Architektur abgeleitet werden, daher ist die Ableitung der Unabhängigkeit mit Hilfe von Analysen sinnvoll. Weil Mehrfachfehler relevant sind, wird, wie in Kapitel 5.2 beschrieben, eine Fehlerbaumanalyse (FTA) genutzt. Und-Verbindungen stellen dabei Fehlerkombinationen dar, welche bei Einzelfehlern nicht direkt zum Ausfall des Systems führen, sofern die Fehler unabhängig voneinander sind. Abbildung 5.7 zeigt einen schematischen Fehlerbaum für das Fail-Operational-System, das in Kapitel 2.3 beschrieben wurde. Ein Ausfall einer Komponente eines Kanals führt zu dessen Deaktivierung.

Die Und-Verbindung, die zum Ausfall im Rückfallbetrieb 3 führt, verbindet den Ausfall in Nominalkanal und den Ausfall im Rückfallkanal, welche hervorgehoben und weiter detailliert wurden. Für die primäre Fahrzeugregelung bedeutet dies, ein gemeinsamer Ausfall mit der sekundären Sensorik, der sekundären Fahrzeugregelung und der sekundären Lenkung führt zu einer Verletzung des Fail-Operational-Verhaltens. Dabei sind insbesondere die identischen Betrachtungsumfänge, wie gleiche funktionale Anteile, kritisch, weswegen im Folgenden die Unabhängigkeit der primären und sekundären Fahrzeugregelung betrachtet wird.

Den nächsten Schritt des Vorgehens stellt die Erstellung einer generischen Vorlage dar. Für die Validierung wurden, wie beschrieben, die Beispiele aus der ISO 26262 und Initiatoren aus vergangenen Analysen genutzt und in das Datenbanksystem eingepflegt. Dadurch werden die geforderten Themen aus der ISO 26262 adressiert. Durch Systemanalyse, Erfahrungen und Feldbeobachtungen wird die Datenbank iterativ ergänzt, was im Rahmen der Validierung exemplarisch erfolgt ist.

Das Ziel der Analyse stellt die Identifikation von Risiken und Maßnahmen bezüglich eines gemeinsamen Ausfalls der Regelfunktion auf Nominal- und Rückfallkanal sowie der damit einhergehende Nachweis der Unabhängigkeit



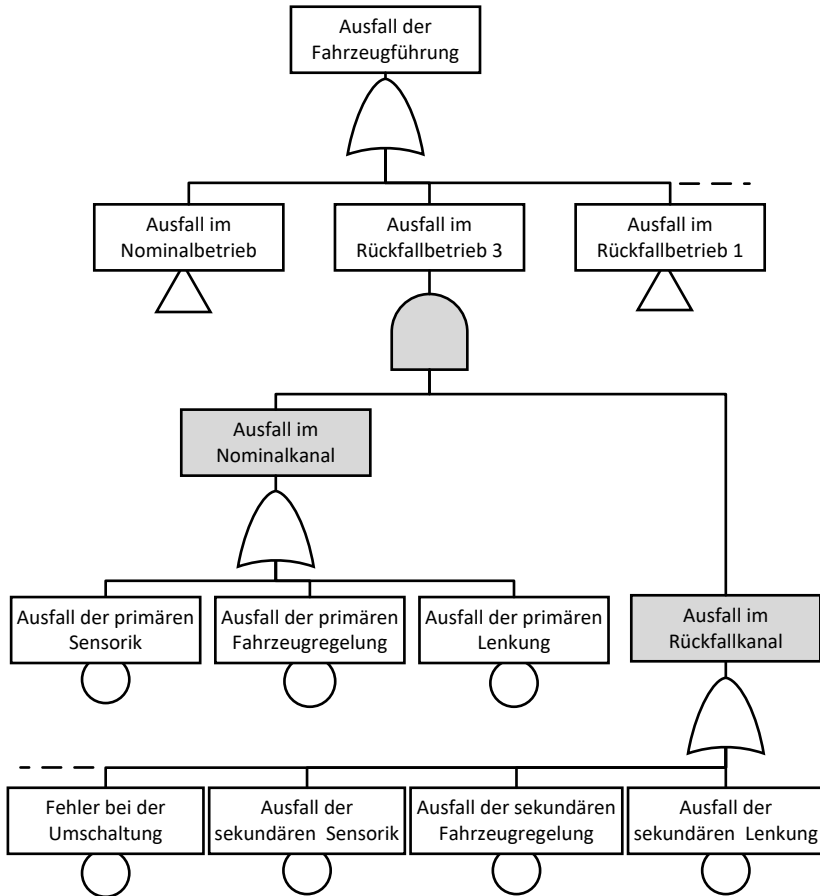


Abbildung 5.7: Fehlerbaum zur Identifikation von Unabhängigkeiten

dar. Schnittstellen sind durch Ein- und Ausgangssignale definiert. Tabelle 5.1 zeigt einen Ausschnitt der Analyse anhand der Entitäten *Common-Cause Initiators*, *Risiken* und *Maßnahmen*. Diese sind entsprechend des Datenschemas, eingebettet als Untertabellen, dargestellt. Begriffe wurden dabei analog der Norm verwendet. Die Selektion der Datensätze bezüglich des definierten Umfangs erfolgt durch die Auswahl der relevanten Initiatoren und eine

Ergänzung um Erkenntnisse der Systemanalyse. Zuerst wird bezüglich des funktionalen Betrachtungsumfanges gefiltert. Im Anschluss wird die Relevanz für den jeweiligen Umfang bewertet. Im Rahmen der Validierung erfolgte dies auf Basis von Experteneinschätzungen, beziehungsweise dem Abgleich der Funktionsmodelle. Im Anwendungsfall wurden identische Eingangssignale von den Probanden als relevant bewertet, dagegen muss der Code Generator nicht betrachtet werden, da dieser nur für Funktionen im Nominalkanal zum Einsatz kommt. Identische Busanschlüsse sind nicht vorhanden. Die Erweiterung der Checkliste erfolgt auf Basis einer Systemanalyse. Genutzt wurde hierfür das Abhängigkeitsmodell nach Schnellbach [Sch16] (vgl. Abbildung 3.1), welches im Rahmen der Validierung, insbesondere bei modellbasierter Entwicklung, eine effiziente und einfache Analysemethode darstellt. Dabei wurden zusätzlich Fahrzeugparameter als gleiche Information (engl. shared Information) identifiziert.

Den nächsten Schritt stellt die Identifikation der Risiken für jeden Initiator dar. Im Falle der identischen Eingangssignale wurden daher alle relevanten Signale analysiert. Das Geschwindigkeitssignal wurde als kritisch eingestuft, da ein Fehler zu einer Degradation und zum Abschalten beider Kanäle führen kann. Entsprechend wurde eine Maßnahme definiert. In der sekundären Fahrzeugregelung wird ein diversitäres Signal, welches im Rückfallkanal gebildet wird und damit unabhängig vom Nominalkanal ist, genutzt. Eine Prämisse, die entsprechend dokumentiert wird, ist, dass auch im Rückfallkanal die Information unabhängig und mit ausreichender Genauigkeit und Integrität zur Verfügung steht. Die Prämissen werden bei Änderungen iterativ abgeglichen, um eventuelle Widersprüche und damit eine erneut notwendige Bewertung zu identifizieren. Dieser Schritt wurde durch die Probanden als zwingend erforderlich bewertet, da nach deren Erfahrung Änderungen eine Quelle für unerkannte gemeinsame Ausfälle darstellen. Im Fall des Signals für die Querschleunigung dagegen erfolgt zwar eine Degradation, jedoch keine Verletzung eines Sicherheitsziels.

Tabelle 5.1: Struktur und Beispiel der Fallstudie Fahrzeugführung

Common-Cause Initiator	Betrachtungsumfang	Coupling Factor Klasse	Relevant für Umfang	Begründung	Prämisse	[...]
Identischer Code Generator	Funktional	Systematic Coupling	✗	Code Generator nur auf Nominalkanal	keine Modellbasierte Entwicklung auf Rückfallkanal	
Identische Busanschlüsse	Hardware	Communication	✗			
Identische Eingangssignale	Funktional	Shared information	✓			
	<b>Risiko</b>	<b>Bewertung</b>	<b>Kritikalität</b>	<b>Status</b>	<b>[...]</b>	
	identische Fahrzeuggeschwindigkeit	Ausfall führt zum Ausfall beider Kanäle	hoch	bewertet		
		<b>Maßnahme</b>	<b>Prämisse</b>	<b>[...]</b>		
		Rückfallkanal nutzt diversitäres Signal	Odometrie auf Rückfallkanal bestimmt unabhängige Fahrzeuggeschwindigkeit			
	Identische Querbeschleunigung	Signale sind divers	keine	bewertet		
Identische Fahrzeugparameter	Funktional	Shared Information	✗			

Das Vorgehen wurde inklusive dem Datenmodell von den Probanden als einfach anwendbar und nachvollziehbar bewertet. Bei einem Gegenprüfen der Anforderungen bestätigten die Probanden deren Erfüllung. Die Umsetzung mittels eines Datenbanksystems bietet einen Mehrwert gegenüber Tabellenblättern, da durch die Verwendung von eingebetteten Tabellen die Übersichtlichkeit verbessert und eine Mehrfachzuordnung möglich wird. Auch Struktur und Aufbau des Datenmodells wurden positiv beurteilt. Schwierigkeiten wurden bezüglich der Synchronisation der Initiatoren identifiziert. Eine identische Detaillierungstiefe von Common-Cause Initiatoren und Risiken stellt möglicherweise, insbesondere zu Beginn, eine Herausforderung dar. Allerdings wurde bereits im Rahmen der Validierung ein einheitliches Maß gefunden. Für eine projektübergreifende Umsetzung wird daher ein unternehmensweiter Austausch aller Anwender notwendig sein. Zusätzlich wurden für eine einfachere Planbarkeit und Statusübersicht Metriken vorgeschlagen, welche auf einem Datenblatt automatisiert bestimmt und dargestellt werden können. Das Vorgehen basiert dabei auf den in Kapitel 5.1 identifizierten Anforderungen, deren Validität bereits diskutiert wurde. Eine Einschränkung auf spezifische Systemkonzepte ist nicht gegeben, wobei sich Vorteile des Vorgehens insbesondere bei Unabhängigkeitsanforderungen verteilter Systemumfänge ergeben. Den nächsten Schritt stellt die Verwendung des Vorgehens da und damit eine umfangreiche Validierung durch alle Stakeholder dar.

## 5.4 Quantifizierung von gemeinsamen Ausfällen

Neben der qualitativen Betrachtung werden zur Analyse von gemeinsamen Ausfällen, insbesondere bei der Analyse von Industrieanlagen, quantitative Ansätze genutzt. Diese berücksichtigen die Wahrscheinlichkeit gemeinsamer Ausfälle in der quantitativen Gesamtbetrachtung [Eri16]. Neben einer realistischen Gesamtausfallrate kann durch quantitative Betrachtungen auch ein Maß für ausreichende Unabhängigkeit bei Invertierung der Berechnung abgeleitet werden. In der Automobilindustrie werden solche Methoden bisher nicht genutzt. Die ISO 26262 [Int18] verweist, sofern eine Quantifizierung notwendig ist, auf Experteneinschätzungen, da keine verlässliche Datenbasis verfügbar ist [Int18].

Zur Bestimmung der Wahrscheinlichkeiten gemeinsamer Ausfälle gibt es in der Literatur verschiedene Ansätze. O'Connor und Mosleh [OM16] unterscheiden Basic-Parameter-Modelle, Ratio-Modelle und Schock-Modelle. Das Basic Parameter Modell wird bei bekannten Ausfallraten aufgrund von gemeinsamen Ausfällen verwendet. Ratio-Modelle dagegen schätzen die Wahrscheinlichkeit auf Basis einer Korrelation von gemeinsamen Ausfällen und den Ausfallraten der Einzelsysteme ab. Schock-Modelle basieren auf der Annahme eines Ausfalles, dem, ebenso wie einer Belastung, eine Wahrscheinlichkeitsverteilung zu Grunde liegt. [OM16] Im Folgenden werden quantitative Methoden anhand eines Ratio-Modell-Ansatzes, der  $\beta$ -Faktor Methode, erklärt und deren Anwendung in der Automobilindustrie diskutiert. Der Ansatz wird in der Norm IEC 61508 [Deu01] zur Bestimmung von Abhängigkeiten vorgeschlagen und die Anwendung ist relativ einfach.

Nach der IEC 61508 ist eine Korrelationsbeziehung von systematischen und zufälligen Fehlern in der Praxis durch Systemkomplexität und den Instandhaltungsaufwand gegeben. Damit kann von Fehlerraten zufälliger Ausfälle auf systematische Fehler geschlossen werden. Der  $\beta$ -Faktor stellt den Anteil der gemeinsamen Ausfälle (engl. Common Cause) im Verhältnis

zu den Gesamtausfällen des Systems dar:  $\beta = \frac{\lambda_{CC}}{\lambda}$ . Für das mehrkanalige Fail-Operational-System bedeutet das, die Wahrscheinlichkeit eines gemeinsamen Ausfalls  $\lambda_{CC}$  wird unter Berücksichtigung des  $\beta$ -Faktors mit  $\lambda_{CC} = \beta \lambda_{kanal}$  bestimmt. Wobei die Ausfallwahrscheinlichkeit eines Kanals  $\lambda_{kanal}$  ist. Die Wahrscheinlichkeit eines unabhängigen Ausfalls eines Kanals ist demnach  $(1 - \beta) \lambda_{kanal}$ . Abbildung 5.8 zeigt ein einfaches Beispiel eines mehrkanaligen Systems anhand dessen die quantitative Bestimmung von gemeinsamer Ausfälle gezeigt wird.

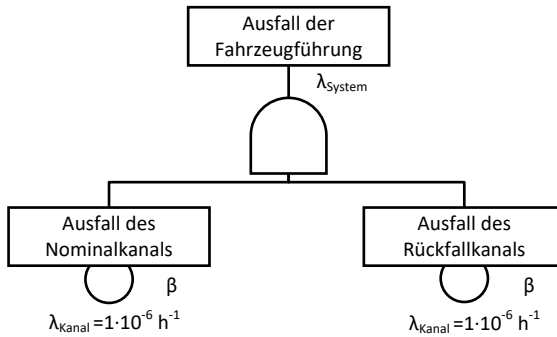


Abbildung 5.8: Beispielhafter Fehlerbaum für die quantitative Bestimmung gemeinsamer Ausfälle

Dargestellt sind der Ausfall des Nominalkanals und des Rückfallkanals mit der jeweiligen Ausfallrate  $\lambda_{kanal}$ . Der Ausfall der Fahrzeugführung ergibt sich durch den Ausfall beider Kanäle mit der Ausfallrate  $\lambda_{system}$ . Bei unabhängigen Basisereignissen ergibt sich die Ausfallrate des Systems zu  $\lambda_{system} = \lambda_{kanal}^2 = 1 \cdot 10^{-12} h^{-1}$ . Wird bei der Berechnung ein  $\beta$ -Faktor berücksichtigt verringert sich die Verfügbarkeit des Gesamtsystems. Beispielhaft wird, entsprechend der IEC 61508 der kleinste mögliche Faktor, ein  $\beta$ -Faktor von 1% angenommen. Die Gesamtausfallrate wird durch  $\lambda_{system} = ((1 - \beta) \lambda_{kanal})^2 + \beta \lambda_{kanal} = 1 \cdot 10^{-8} h^{-1}$  bestimmt und unterscheidet sich damit deutlich von der Rate ohne Berücksichtigung der gemeinsamen Ausfälle. [Jon12]

Im Falle unterschiedlicher Ausfallraten der abhängigen Systeme kann zur

Bestimmung des  $\beta$ -Faktors eine gewichtete Kalkulation oder eine Worst-Case Abschätzung erfolgen [Bel17].

Zur Ableitung der Zusammenhänge ist eine Bestimmung der quantitativen Faktoren, wie dem  $\beta$ -Faktor, oder die Quantifizierung der Wahrscheinlichkeiten von gemeinsamen Ausfällen notwendig. Nach Jones [Jon12] kann zum einen eine Abschätzung der Wahrscheinlichkeiten und zum anderen eine Ableitung von Fehlerdaten jeweils für beide Fälle erfolgen. Eine Abschätzung kann auf Experteneinschätzung oder, wie in der IEC 61508 präsentiert, auf einer Checkliste basieren [Jon12].

Die Checkliste zur Bestimmung des  $\beta$ -Faktors, welche in der IEC 61508 vorgestellt wird, ist auf Industrieanlagen und nicht auf Fahrzeugsysteme ausgelegt und daher nur in wenigen Punkten für die Analyse einer Fail-Operational-Fahrzeugsführung anwendbar [Deu01]. Eine Experteneinschätzung erfordert eine detaillierte Systemanalyse, um die Fehlerursachen zu berücksichtigen und Nachvollziehbarkeit zu gewährleisten sofern keine Fehlerdaten verwendet werden [Jon12].

Bezüglich der Analyse und der Datensammlung von gemeinsamen Ausfällen wurde insbesondere in der Nuklearindustrie durch langjährige Untersuchungen eine Datenbasis geschaffen [HHHH16] und Richtlinien für eine solche Datensammlung entwickelt [NEA04]. Diese werden im Folgenden kurz vorgestellt und im Anschluss deren Übertragbarkeit auf die Automobilindustrie diskutiert.

Eine Systembeschreibung der betrachteten Anlagen inklusive der Anzahl, Herstellerinformationen und Testprozeduren werden dokumentiert. Dabei werden nur Anlagen mit einer Laufzeit von mindestens fünf Jahren zur Datenerhebung genutzt. Außerdem werden Daten zu den aufgetreten Fehlerfällen gesammelt. Die gemeinsamen Ausfälle werden anhand der Fehlermodi, der Fehlerursache, der Art der funktionalen Fehlerauswirkung und der Erkennungen spezifiziert. Es erfolgt zudem eine Klassifizierung anhand der Ursache und der Auswirkung, der Koppelmechanismen, des zeitlichen Verlaufs und der Verbesserungsmaßnahmen. [NEA04] Für die Fehlerursache werden dabei verschiedene Kategorien, wie Design und Umweltfaktoren

unterschieden. Diese entsprechen den Aspekten, die nach der ISO 26262 in einer Analyse abhängiger Fehler zu berücksichtigen sind.

Eine entsprechende Datenanalyse wäre auch in der Automobilindustrie denkbar. Fehlerspeichereinträge und weitere Felddaten werden heutzutage bereits in den Fahrzeugen gesammelt. Eine Analyse gemeinsamer Ausfälle erfordert die Ablage von Fehlerevents mit exaktem Auftretenszeitpunkt, Ursache und Auswirkungen, um eine Identifikation gemeinsamer Ausfälle zu ermöglichen. Dies beinhaltet auch Fahrsituationen und Umweltfaktoren, welche Fehlerursachen darstellen können. Die Ablage muss dabei mit einer Taktung im Bereich von Millisekunden erfolgen, um Aufschlüsse zu ermöglichen. Des Weiteren müssen, wenn der Fehlerspeichereintrag nicht der initialen Fehlerursache entspricht, die Fehlerketten nachvollziehbar sein, um die Ursachen und Koppelmechanismen identifizieren zu können. Dies erfordert zum einen genaue Kenntnisse des Verhaltens im Fehlerfall aber auch eine umfangreiche Ablage von Daten und verursacht damit Kosten. Des Weiteren stellen die Entwicklungszyklen in der Automobilindustrie eine Einschränkung dar. Neue Fahrzeugarchitekturen mit neuen Technologien kommen jährlich auf den Markt und einzelne Derivate unterscheiden sich deutlich bezüglich der Randbedingungen, Architekturen und Systeme. Daten, die daher in Fahrzeugen über Jahre gesammelt werden, sind unter Umständen für Fahrzeuge der nächste Generation nicht aussagekräftig. Ein weiterer Unterschied der Automobilindustrie zur Nuklearindustrie ist, dass menschlicher Einfluss und Instandhaltung im Fahrzeug einen deutlich geringen Einfluss auf Fehler haben, als das Design, Einsatzbedingungen und die Produktion (vgl. [Int+06, Int18]). Relevante Daten für die Automobilindustrie erfordern, im Vergleich zur Nuklearindustrie, deutlich detailliertere Analysen der Fahrzeugarchitektur, inklusive der Hardware und des Funktionsnetzes. Außerdem ist eine adäquate Klassifizierung notwendig. Diese muss auch organisatorische Aspekte berücksichtigen, wofür es nicht reicht, Felddaten über eine Fahrzeuggeneration zu sammeln, sondern über einen deutlich längeren Zeitraum und mehrere Entwicklungszyklen, wobei auch hier Veränderungen berücksichtigt werden müssen.



Trotz der Vorteile einer Quantifizierung ist eine umfangreiche Anwendung quantitativer Methoden zur Analyse gemeinsamer Ausfälle in der Automobilindustrie nicht ohne weiteres möglich. Ein solches Vorgehen ohne eine ausreichende Datengrundlage ist nicht sinnvoll einsetzbar. Eine entsprechende Datengrundlage ist in der Automobilindustrie heutzutage nicht vorhanden. Die Datensammlung über lange Zeiten steht dabei mit den kurzen Entwicklungszyklen und damit einhergehenden Änderungen im Konflikt. Nichtsdestotrotz ist für einzelne Argumentationen eine Quantifizierung möglich, wenn entsprechende Daten und Experteneinschätzungen verfügbar sind, insbesondere wenn es um die Bewertung einer ausreichenden Unabhängigkeit geht.



KAPITEL  
6

# FORMALE VERIFIKATION DER UMSCHALTLOGIK

Fehlertoleranz und damit das Fail-Operational-Verhalten der Fahrzeugführung erfordert eine korrekte Funktion der Umschaltlogik (vgl. Kapitel 4.4). Für den Nachweis der funktionalen Sicherheit ist daher der Ausschluss systematischer Fehler in der Logik, die eine Umschaltung im Fehlerfall verhindern, notwendig. Die Analyse liefert damit, wie in Rahmen der Sicherheitsargumentation (vgl. Kapitel 4.4) gezeigt, zusätzlich einen Beitrag zum Nachweis des Übergangs in den sicheren Zustand und der Einhaltung der Fehlertoleranzzeit im Rahmen des Degradationskonzepts. Da die Umschaltlogik aus einzelnen, verteilten Zustandsautomaten besteht, die im Zusammenspiel die Funktion erfüllen, muss der Nachweis im Verbund (vgl. Kapitel 2.3) erfolgen. Formale Methoden werden dabei von der ISO 26262 für höhere Integritätslevel, ASIL C und D, empfohlen. Vollmer [Vol20] sieht im Kontext hochautomatisierter Fahrfunktionen eine Prüfung der Anwendbarkeit formaler Methoden, aufgrund der Möglichkeit einer vollständigen Prüfung gegen die Spezifikationen, als zwingend erforderlich an.

Für den Nachweis der Korrektheit wird im folgenden Kapitel bounded Model-Checking genutzt. Die Methode bietet, wie in Kapitel 2.4.3 beschrieben, die Möglichkeit einer vollständigen Verifikation des System gegen die Spezifikation und damit den Nachweis der Erfüllung der Sicherheitsanforderungen innerhalb eines definierten Intervalls. Das Intervall kann dabei begrenzt werden, da das System nach einer bestimmten Zeit einen stationären Zustand erreicht sofern die möglichen Fehlerzeiträume eingeschränkt werden können. Die Analyse erfolgt automatisiert, was aufgrund der Anzahl der Umschaltsequenzen und -varianten und der Fehlerkombinationen unabdingbar ist. Beispielsweise kann der Ausfall mehrerer Komponenten zum Ausfall dieser in beliebiger Reihenfolge und mit beliebigen Latenzen oder Gleichzeitigkeit führen. Eine manuelle Analyse, zum Beispiel eine FMEA, wäre daher nicht durchführbar.

Im Folgenden werden die einzelnen Schritte des Model-Checkings der Verifikation einer Fail-Operational-Umschaltlogik entsprechend den beschriebenen Vorgehensweisen in Kapitel 2.4.3 vorgestellt. Dabei werden auch die Vorgaben der ISO 26262 adressiert. Für das Model-Checking wurde das Werkzeug NuSMV<sup>1</sup> verwendet. Auch andere Werkzeuge, wie SPIN<sup>2</sup> [Hol08], UPAAL<sup>3</sup> [BDL04] und der Design Verifier in Matlab, Simulink [The19] wurden in Betracht gezogen. Mit dem Design Verifier konnten die Spezifikation, insbesondere temporale Zusammenhänge, nur teilweise abgebildet werden, darüber zeigten sich sehr lange Rechenzeiten. Die Modellierung in UPAAL erfolgt, wie auch beim Desing Verifier, graphisch, was Anpassungen durch Änderungen aufwändig macht. UPAAL ist darüber hinaus für kommerzielle Zwecke nicht lizenzfrei nutzbar. SPIN nutzt explizites, erschöpfendes Model-Checking. NuSMV dagegen basiert auf symbolischem Model-Checking und unterstützt sowohl bounded, als auch unbounded Model-Checking sowie synchrone und asynchrone Implementierung. Da eine große Anzahl an Zuständen zu erwarten war, fiel die initiale Wahl auf

---

<sup>1</sup><http://nusmv.fbk.eu/>

<sup>2</sup><http://spinroot.com>

<sup>3</sup><http://upaal.org>

NuSMV, das die Möglichkeit der Begrenzung der Suchtiefe bietet. Auch die Möglichkeit der asynchronen Implementierung stellte einen Auswahlgrund dar. Im Rahmen dieser Arbeit konnte gezeigt werden, dass NuSMV eine geeignete Wahl für die Verifikation der Umschaltlogik darstellt und die Verifikation umsetzbar ist.

Zu Beginn erfolgt die Modellierung des Systems und die Formulierung der Systemspezifikationen mittels temporaler Logik (vgl. Kapitel 6.1). Abgeleitet aus den Vorgaben der ISO 26262 werden die relevanten Fehlerkombinationen und die Implementierung in Kapitel 6.2 vorgestellt. Im Anschluss, in Kapitel 6.3 erfolgt die Validierung. Für einen Sicherheitsnachweis gemäß ISO 26262 ist zudem die Qualifikation des Software-Werkzeugs notwendig. Die Ergebnisse und die Anwendbarkeit der Methode für den Nachweis der funktionalen Sicherheit gemäß ISO 26262 werden zum Abschluss diskutiert.

## 6.1 Modellierung des Systems und formale Spezifikation

Im ersten Schritt erfolgt, entsprechend dem in Kapitel 2.4.3 beschriebenen Vorgehen, die Modellierung des Systems. Im Anschluss wird die formale Spezifikation der Sicherheitsanforderungen vorgestellt. Die Schritte stellen die manuelle Vorbereitungsphase dar.

### 6.1.1 Modellierung der Umschaltlogik

Das zu verifizierende Fail-Operational-System wurde in Kapitel 2.3 vorgestellt. Auf die, für die Modellierung der Umschaltlogik relevanten Aspekte, wird an dieser Stelle detaillierter eingegangen.

Zweck der Umschaltlogik ist die Aktivierung und Deaktivierung sowie die Rekonfiguration des Systems nach Fehlern und damit das Sicherstellen des Fail-Operational-Verhaltens. Die Logik besteht aus einzelnen Zustandsautomaten, die durch den Austausch der Zustände und deren Berücksichtigung in Transitionsbedingungen gekoppelt sind und im Verbund die Umschaltung koordinieren (vgl. Kapitel 2.3). Darüber hinaus werden auch externe Signale,

wie Aktivierungs- und Deaktivierungsaufforderungen und die Meldungen funktionaler Fehler, in Transitionsbedingungen genutzt. Es ist jeweils ein Zustandsautomat auf jedem relevanten Steuergerät partitioniert. Deren Kommunikation erfolgt über ein Bussystem, bestehend aus mehreren Kommunikationsbussen, das den Zustand der jeweiligen Zustandsautomaten an die Übrigen übermittelt. Die Berechnung der Zustände erfolgt zyklisch, und ist im Allgemeinen weder über Steuergeräte hinweg noch mit Kommunikationsbussen synchronisiert. Die Funktionen auf einem Steuergerät kommunizieren im Falle einer Fehlfunktion, was eine Integritätsverletzung einschließt, den Fehlerzustand an den Zustandsautomaten auf dem entsprechenden Steuergerät. Dieser schaltet auf den Zustand *Fehler*, woraufhin die Zustandsautomaten des Kanals entsprechend die Funktionen deaktivieren. Sofern der Kanal zuvor aktiv war, reagieren die Zustandsautomaten des anderen Kanals auf die Deaktivierung und schalten auf *Aktiv*. Sowohl die Kommunikation als auch die Energieversorgung ist auf jedem der Kanäle unabhängig umgesetzt.

Im Folgenden werden die Modellierung der Zustandsautomaten und der Architektur im verwendeten Werkzeug, NuSMV, erklärt. NuSMV ist, wie beschrieben, ein frei verfügbares, symbolische Model-Checking Werkzeug, das sowohl Model-Checking mit binären Entscheidungsdiagrammen als auch SAT Ansätze für bounded Model-Checking unterstützt. Syntax und Semantik der Modellierungssprache sind auf die Anwendung für zustandsbasierte Systeme zugeschnitten. Außerdem werden verschiedene Datentypen sowie boolesche Ausdrücke, LTL und CTL abhängig vom genutzten Ansatz unterstützt. [CCG+02]

Jeder Zustandsautomat wird als Modul, eine Kapselung von Variablen und Spezifikationen ähnlich einer Funktion, implementiert. Dem Modul werden Eingangsvariablen übergeben, der eigene Zustand stellt den Ausgang dar. [CCJ+10] Die beispielhafte, abstrahierte Implementierung eines Zustandsautomaten als Modul, analog zu Abbildung 2.10b, wird in Quellcode 6.1 gezeigt. Die Notation orientiert sich dabei an der Syntax des

```

1  MODULE MFS1 (FF1,FF2,EPS1,BRS1,Fehlerfunkt, Aktivierung)
2  VAR
3    FS1 : { Init, Bereit, Fehler, Aktiv };
4
5  ASSIGN
6    init FS1 := Init;
7    next FS1 :=
8      case
9        FS1 = Init & !Fehlerfunkt : {Bereit};
10       FS1 = Bereit & Aktivierung & ((FF1 & FF2) = Bereit) : {Aktiv};
11       FS1 = Bereit & Fehlerfunkt : {Fehler};
12       FS1 = Aktiv & !Aktivierung : {Bereit};
13       FS1 = Aktiv & ((Fehlerfunkt | FF1 | BRS1 | EPS1) = Fehler) : {Fehler};
14       FS1 = Fehler & !Fehlerfunkt : {Init}
15     true : {FS1};
16   esac;

```

Quellcode 6.1: Modellierung eines Zustandsautomaten als Modul

Model-Checking Werkzeugs (vgl. [CCJ+10]), wurde aber aus Gründen der Übersicht vereinfacht. Die Schreibweise  $(FF1 \ \& \ FF2) = Bereit$  ist dabei gleichbedeutend mit  $(FF1 = Bereit) \ \& \ (FF2 = Bereit)$

Zu Beginn wird das Modul  $M_{FS1}$  für die primäre Fahrstrategie inklusive der Eingangssignale definiert. Die Variablen FF1 beziehungsweise FF2 entsprechend den Zustandsautomaten der primären und sekundären Fahrzeugführung, EPS und BRS stellen das Lenk- und Bremssystem dar. Der Wert der Variable kann jedem Zustand des Zustandsautomaten entsprechen. Die Zustände und Zustandsübergänge werden im Anschluss spezifiziert. Dabei wird mittels einer Fallunterscheidung, abhängig vom Ausgangszustand und den anliegenden Bedingungen der Folgezustand bestimmt. Beispielsweise erfolgt der Übergang von *Bereit* zu *Aktiv*, wenn eine Aktivierung angefordert und sowohl FF1 als auch FF2 in *Bereit* sind.

Neben der Logik der Zustandsautomaten muss auch die Architektur des Systems im Modell berücksichtigt werden. Das umfasst die Partitionierung, sowie die Kommunikation und die Energieversorgung. Relevant ist das Fehlerver-

halten beziehungsweise die Auswirkungen auf das System. Jede individuelle Komponente reagiert auf einen kritischen Fehler in der Komponente mit einem Fail-Silent Verhalten, einem Abschalten. Eine analoge Reaktion erfolgt bei einem Fehler in der Versorgung der Komponenten, beispielsweise Unterspannung. Die jeweiligen Maßnahmen, wie Sicherheitsabschaltungen im Fehlerfall, werden im Sicherheitskonzept der Komponenten berücksichtigt (vgl. Kapitel 4.4). Eine abgeschaltete Komponente sendet kein Signal mehr. Daher werden Steuergeräteausfälle entsprechend der Partitionierung und Energieversorgung implizit über die Kommunikation modelliert. Jede einzelne Kommunikationsverbindung wird unabhängig vom Bus eingeführt. Die Empfänger reagieren auf fehlende Eingänge und behandeln ausbleibende Signale analog eines empfangenen Fehlerzustands. Zusätzlich werden aus Gründen der Robustheit Entprellungszyklen implementiert, die im Modell berücksichtigt werden. Die Entprellung führt dazu, dass für eine bestimmte Zyklenzahl der Zustand *Fehler* oder kein Signal gesendet werden muss, bevor der Empfänger diese als *Fehler* interpretiert und darauf reagiert [ST12]. Für die Berücksichtigung der Zeiten, wie der Entprellzeit und der Fehlertoleranzzeit in Abhängigkeit des Umschaltzeitpunktes werden Zähler genutzt, die entsprechend des Rechenzyklus die Zeit beinhalten. Zur Darstellung von Steuergeräte-, Bus-, oder Energiebordnetzfehlern, werden die relevanten Kommunikationsverbindungen manipuliert. Eine beispielhafte Implementierung einer Kommunikationsverbindung zwischen dem Zustandsautomat der Fahrstrategie (FS) und der Fahrzeugführung (FF) auf den Nominalkanal wird in Quellcode 6.2 gezeigt. Dabei wird nur der Fehlerfall der Energieversorgung dargestellt. Wenn ein Fehler der Energieversorgung länger als die definierte Entprellzeit, im Beispiel drei Zyklen, anliegt, wird ein Fehlersignal ausgegeben. Andernfalls wird der Zustand beibehalten, der im fehlerlosen Fall dem Zustand der primären Fahrzeugführung entspricht. Die Initialisierung des Modells umfasst neben den Modulen auch einen Bus, der alle Kommunikationsverbindung beinhaltet. Die Kommunikationssignale ( $\text{Komm}_{\text{Empfänger}_{\text{Sender}}}$ ) werden entsprechend der Eingangssignale den Modulen bei der Initialisierung übergeben.



```

1  init KommFS1FF1 := Init;
2  next KommFS1FF1 :=
3  case
4    FehlerEnergie & Entprellzeit ≥ 3 : {Fehler};
5    FehlerEnergie & Entprellzeit < 3 : {KommFS1FF1};
6    true : {FF1};
7  esac;

```

Quellcode 6.2: Beispiel einer Kommunikationsverbindung

### 6.1.2 Spezifikation der Sicherheitsanforderungen

Die formale Spezifikation der Sicherheitsanforderungen ist der nächste Schritt für die Verifikation des Fail-Operational-Systems. Sicherheitsanforderungen abgeleitet aus Sicherheitszielen (vgl. Tabelle 2.1), beschreiben das Systemverhalten, welches notwendig ist, um funktionale Sicherheit zu gewährleisten. Diese werden in formaler Logik beschrieben und genutzt, um das System mittels Model-Checking dagegen zu verifizieren.

Die Sicherheitsziele, beschrieben in Tabelle 2.1, umfassen die Aktivierung, die nur bei Bereitschaft aller Komponenten erfolgen darf, die Deaktivierung bei Anforderung und die Umschaltung im Fehlerfall unter Berücksichtigung der Fehlertoleranzzeit.

Für die Beschreibung der Systemzustände vor und nach den Umschaltvorgängen werden vorab die Betriebszustände, der Nominalbetrieb (NB) und die Rückfallbetriebe (RB1 beziehungsweise RB2, RB3), definiert. Diese werden jeweils durch die Zustände der Zustandsautomaten beschrieben. Das Beispiel in Quellcode 6.3 zeigt exemplarisch die Definition des primären Rückfallbetriebs. Die einzelnen Komponenten werden wiederum, entsprechend Kapitel 2.3 mit Fahrstrategie (FS), Fahrzeugführung (FF), Bremsregelsystem (BR) und Electric-Power Steering (EPS) bezeichnet. Im Rückfallbetrieb 1 sind demnach alle Komponenten auf dem Nominalkanal aktiv (Zeile 4). Auf dem Rückfallkanal darf keine Komponente aktiv und mindestens eine Komponente muss nicht in *Bereit* sein (Zeile 6), also fehlerhaft sein oder gewesen sein.

```

1  init RB1 := false;
2    next RB1 :=
3      case
4        (FS1 & FF1 & EPS1) = Aktiv
5        & (FF2 & BR2 & EPS2) != Aktiv
6        & !(FF2 & BR2 & EPS2 = Bereit) : true
7      esac;

```

Quellcode 6.3: Definition des primären Rückfallbetriebs (RB1)

Unter Verwendung der Definitionen aller Betriebszustände, werden im Anschluss die Sicherheitsanforderungen für die Verifikation formal spezifiziert. Hierfür wird lineare temporale Logik (LTL), wie in Kapitel 2.4.3 eingeführt, genutzt. Auch der der Model-Checking Ansatz, das bounded Model-Checking muss berücksichtigt werden, da Spezifikation so formuliert werden müssen, dass mit begrenzter Suchtiefe eine definitive Aussage getroffen werden kann. Dabei werden verschiedene Formen von Anforderungen, Sicherheitsanforderungen und Lebendigkeitsanforderungen, unterschieden [BK08]. Sicherheitsanforderungen im Sinne des Model-Checkings sind Anforderungen, welche durch eine finite Sequenz erfüllt werden. Im Rahmen des Model-Checkings wird geprüft, ob eine solche Sequenz beziehungsweise inwiefern eine Sequenz, die die Anforderung verletzt, erreichbar ist. Eine Lebendigkeitsbedingung dagegen stellt eine Bedingung dar, für welche die finite Sequenz zu einer inifiniten Sequenz erweitert werden kann, welche die Bedingung erfüllt. Dies ist meist für iterative Abläufe oder bei Freiheitsgraden, die keine Auswirkung auf den Systemzustand haben der Fall. Die Spezifikation beinhalten dabei eine Bedingung und ein Prüfkriterium, welches erfüllt sein muss, sofern die Vorbedingung wahr ist.

Die Aktivierung darf nur bei Bereitschaft aller Komponenten und einer gleichzeitigen Aktivierungsanforderung erfolgen. Dabei handelt es sich um eine Lebendigkeitsanforderung im Sinne des Model-Checkings.

$G (NB \rightarrow (O ((FS1 \& FF1 \& \dots \& EPS2) = \text{Bereit} \& \text{Aktivierung} = 1)))$

### Spezifikation 6.1: Aktivierung bei Bereitschaft

Die Anforderung muss für jeden Zeitpunkt gelten ( $G$ ). Der erste Term stellt die Bedingung dar, dass das System im Nominalbetrieb ( $NB$ ) ist. Wenn das der Fall ist müssen zuvor einmalig und gleichzeitig ( $O$ ) alle Komponenten bereit gewesen sein und eine Aktivierungsanforderung vorgelegen haben. Der zweite Term, das Prüfkriterium, bezieht sich daher auf die Vergangenheit.

Sicherheitsziele, die eine Umschaltung im Fehlerfall erfordert, stellen keine umsetzbare Anforderung dar, da weder Bedingung noch Prüfkriterium ausreichend eindeutig definiert ist. Im Rahmen des funktionalen Sicherheitskonzeptes wurden diese daher detailliert und werden im Folgenden erklärt. Die erste abgeleitete Anforderung lautet: *Die Umschaltlogik muss bei einem Fehler auf dem Nominalkanal in den Rückfallbetrieb 2 umschalten*. Eine finite Sequenz, welche die Anforderungen erfüllt, kann zu einer infiniten Sequenz erweitert werden, weswegen es sich um eine Lebendigkeitsbedingung handelt. Die Anforderung ist global gültig. Im ersten Teil wird wiederum die Bedingung, im zweiten Teil das Prüfkriterium definiert. Die Bedingung tritt zeitlich vor dem Prüfkriterium ein. Fehler, die in der Vorbedingung berücksichtigt werden und zu einer Umschaltung führen, sind funktionale Fehler, welche nicht entprellt werden, oder Architekturfehler, für die eine Entprellung berücksichtigt wurde. Dabei führen nur die Fehler auf dem Nominalkanal zum Rückfallbetrieb 2. Sofern ein Fehlerfall eintritt soll der Rückfallbetrieb 2 ( $RB2$ ) innerhalb der Fehlertoleranzzeit ( $FTTI$ ) erreicht werden. Die Umschaltung erfordert eine gewisse Zeit, weswegen ein Prüfintervall von  $[FTTI-5, FTTI+5]$  definiert wurde. Dieses muss die Entprellzeit beinhalten, um sicher zu stellen, dass der Zustand stationär ist. Sofern kein Intervall spezifiziert ist, kann bei begrenzter Suchtiefe keine definitive Aussage getroffen werden.

G (((FS1 | FF1 | EPS1) = Fehler | Fehler<sub>NK</sub>.Entprellzeit = 3)  
→ (G [FTTI-5, FTTI+5] (RB2)))

Spezifikation 6.2: Umschaltung in den Rückfallbetrieb 2 bei Einfachfehler auf Nominalkanal

Für die Prüfung der Reaktion bei Mehrfachfehlern, in diesem Fall zwei unabhängigen Fehlern, wurde die Anforderung um den Ausschluss von Einfachfehlern erweitert.

G (((BR1 | FS1 | FF1 | EPS1) = Fehler | Fehler<sub>NK</sub>.Entprellzeit = 3)  
& !(BR1 xor FS1 xor FF1 xor EPS1) = Fehler  
xor Fehler<sub>NK</sub>.Entprellzeit = 3))  
→ (G [FTTI-5, FTTI+5] (RB2)))

Spezifikation 6.3: Umschaltung in den Rückfallbetrieb 2 bei Doppelfehlern auf Nominalkanal

Dies ist notwendig, da Sequenzen, bei denen der Zweitfehler nicht anliegt oder entprellt wird, also keine entsprechende Umschaltung aufgrund des Zweitfehlers erfolgen soll, ausgeschlossen werden müssen. Analog wurden auch für den Rückfallbetrieb 1 und 3 Anforderungen spezifiziert: *Die Umschaltlogik muss bei einem Fehler auf dem Rückfallkanal in den Rückfallbetrieb 1 umschalten.*

Darüber hinaus darf *die Umschaltlogik [...] einen deaktivierten Kanal nicht erneut aktivieren* (vgl. Tabelle 2.1), da dessen Fehlerfreiheit erst bei einer Diagnose nach einem Klemmenwechsel garantiert werden kann. Das bedeutet *der Nominalbetrieb [darf] nach einem Rückfallbetrieb nicht nochmals aktiviert werden*. Die Anforderung stellt im Sinne des Model-Checkings eine Sicherheitsanforderung dar. Eine temporale Eingrenzung durch ein Intervall ist daher nicht notwendig, wobei die Suchtiefe natürlich begrenzt ist. In der Anforderung werden, die nicht gewünschten Übergänge jeweils durch Bedingung und Prüfkriterium berücksichtigt.

$$G ((RB1 \rightarrow !RB2) \& (RB2 \rightarrow !RB1) \& ((RB1 \mid RB2) \rightarrow !NB))$$

#### Spezifikation 6.4: Verhindern der Reaktivierung eines Kanals

Es muss zu jedem Zeitpunkt ein definierter Betriebsmodus aktiv sein, das bedeutet es darf nur maximal ein Betriebsmodus aktiv sein.

$$G (!(NB \& RB1) \& !(NB \& RB2) \& !(RB1 \& RB1))$$

#### Spezifikation 6.5: Exklusivität der Betriebsmodi

Der Ausfall oder ein Zustand während einer Umschaltung stellen dabei keine Verletzung dar, weswegen eine Formulierung mit *exklusiven Oder* nicht zielführend ist.

Darüber hinaus werden Spezifikationen berücksichtigt, die ein Erreichen des Zielbetriebes für einen beliebigen Fehlerfall prüfen. Beispielhaft wird das Erreichen des Rückfallbetriebs 2 vorgestellt.

$$G (((BR1 \mid FS1 \mid FF1 \mid EPS1 \mid FF2 \mid BR2 \mid EPS2) = Fehler \\ \mid Fehler_A.Entprellzeit = 3) \\ \rightarrow (G [FTTI-5, FTTI+5] (RB2)))$$

#### Spezifikation 6.6: sekundärer Rückfallbetrieb Zielbetrieb im Fehlerfall

Die Bedingung wird nicht nur beim Eintreten eines Fehlers auf dem Nominalkanal erfüllt, sondern für jeden Fehler, der eine Reaktion des Systems erfordert. Diese Anforderungen werden entsprechend für Zweifachfehler erweitert und dienen der Validierung (vgl. Kapitel 6.3).

## 6.2 Identifikation relevanter Fehlerfälle und Implementierung

Im nächsten Schritt wird aus den Vorgaben der ISO 26262 und der Sicherheitsargumentation der notwendige Umfang der Analyse bezüglich der zu prüfenden Fehlerfälle und derer Kombinationen abgeleitet. Darauf aufbauend wird die Identifikation relevanter Fehler und die Implementierung, inklusive der Zuordnung von Spezifikationen zu den entsprechenden Fehlerfällen, vorgestellt.

Der Industriestandard ISO 26262 beschreibt Fehlertoleranz, als die Fähigkeit, eine Funktion auch in Gegenwart einer oder mehrerer Fehler bereitzustellen. Die Literatur beschränkt sich bei der Definition auf einzelne Fehler (vgl. Kapitel 2.3). Unabhängige Mehrfachfehler mit einer Ordnung größer zwei werden aufgrund der Eintretenswahrscheinlichkeit in der ISO 26262 als sicher angesehen und müssen, sofern das Sicherheitskonzept dies nicht explizit erfordert, nicht berücksichtigt werden. Für Zweifachfehler erfolgt die Berücksichtigung auf Basis einer Bewertung des Risikos anhand der Diagnoseabdeckung und dem Zeitintervall, in dem der latente Fehler anliegen kann. [Int18] Das bedeutet, dass Zweifachfehler nach den Vorgaben der ISO 26262 nicht vollständig berücksichtigt werden müssen. Im Sinne einer Risikominimierung soll das System allerdings, sofern die Möglichkeit besteht, auch nach Zweifachfehlern funktionsfähig bleiben. Darüber hinaus erfordert die Argumentation der Sicherheit eine Unabhängigkeit gewisser Fehler (vgl. Kapitel 4.4). Die notwendigen Unabhängigkeiten werden mittels eines Fehlerbaums abgeleitet (vgl. Kapitel 3.2 und Kapitel 5.2). Dabei wird die Umschaltlogik berücksichtigt, weswegen das Verhalten nicht nur bei tolerierten Fehlern, sondern auch bei Fehlern, die zum Systemausfall führen, verifiziert werden muss. Eine Verifikation von Fehlerkombinationen dient zudem dem Systemverständnis, da das Verhalten mit dem erwarteten Verhalten abgeglichen wird. Daher werden Einfach- und Zweifachfehler in jeglicher Kombination in der Verifikation berücksichtigt

Im nächsten Schritt wird die Identifikation relevanter Fehlerfälle, ausgehend von der Funktion, der Hardware und der Systemarchitektur analog einer FMEA vorgestellt. Im Anschluss erfolgt die Zuordnung der entsprechenden Anforderungen zu den jeweiligen Fehlerfällen. Sofern Fehlerfälle identifiziert werden, die im Modell nicht berücksichtigt wurden, ist eine Anpassung notwendig. Dies bezieht sich insbesondere auf die Spezifikation der Kommunikationsausfälle (vgl. Spezifikation 6.2).

Fehler, die durch Funktionen an die jeweiligen Zustandsautomaten gemeldet werden, müssen berücksichtigt werden. Die relevanten Meldemechanismen werden auf Funktions- und Softwareebene analysiert (vgl. Kapitel 4.4). Auf der Hardwareebene sind auf den Steuergeräten, wie bereits beschrieben, Maßnahmen implementiert, die zum Abschalten des Steuergerätes bei kritischen Fehlern führen. Dies gilt analog für kritische Fehler der Basissoftware. Daher ist der Steuergeräteausfall der relevante Fehlermodus auf Hard- und Softwareebene. Architekturfehler betreffen die Spannungsversorgung und die Kommunikation. Auch für die Spannungsversorgung sind Sicherheitsmechanismen implementiert, welche zu einem Abschalten der Versorgung im Fehlerfall führen. Fehler können dabei in lokalen oder kommunalen Versorgungsmodulen der Komponenten auftreten, was zu einem individuellen oder entsprechenden gemeinsamen Ausfall führt. Daher sind bezüglich der Spannungsversorgung ein vollständiger Ausfall eines Bordnetzes und ein lokaler Spannungsausfall zu berücksichtigen. Gleiches gilt für die Kommunikation. Sowohl der Ausfall eines Kommunikationsbusses als auch lokale Ausfälle, stellen zu berücksichtigende Fehlerszenarien dar. Lokale Ausfälle betreffen Sender, Empfänger und einzelne Nachrichten. Die Ausfälle von Sender und Empfänger werden durch die Ausfälle einzelner Komponenten abgedeckt. Ein Ausfall einzelner Nachrichten stellt einen weiteren Fehlerfall dar, der aufgrund von Sicherheitsmaßnahmen, wie Check-Summen, auch die Verfälschung von Nachrichten abdeckt.

Für die Verifikation ist zudem die Zuordnung der Spezifikation zu den Fehlerfällen und die Konfiguration und der Aufruf des Model-Checking Werkzeugs erforderlich.

Nicht jede Anforderung ist für jeden Fehlerfall relevant. Eine Zuordnung von Anforderungen zu Fehlerfällen dient der Laufzeitorientierung durch eine geringe Anzahl an zu prüfenden Anforderungen und vereinfacht die Formulierung der Anforderungen, da in den Bedingungen nicht alle Fehlerfälle und Ausschlusskriterien berücksichtigt werden müssen. Zudem ermöglicht die Zuordnung die Validierung anhand der Spezifikation des Zielbetriebs für beliebige Fehlerfälle.

Um die für den jeweiligen Fehlerfall, beziehungsweise die Fehlerkombination relevante Untermengen der Anforderungen zu definieren, wurde eine Matrix der Fehlerkombinationen und den entsprechenden Betriebszuständen nach der Umschaltung genutzt. Tabelle 6.1 zeigt den Aufbau der Matrix. In der

Tabelle 6.1: Aufbau der Matrix für die von Fehlerkombinationen und Betriebszustand

Zweitfehler	funkt. Fehler Lenkung 1	Energieversorgung Rückfallkanal	[...]
Erstfehler			
funkt. Fehler Lenkung 1	Rückfallbetrieb 2	Systemausfall	[...]
Energieversorgung Rückfallkanal	Systemausfall	Rückfallbetrieb 1	[...]

Matrix sind vertikal die Erstfehler und horizontal die Zweitfehler aufgetragen. Damit ergeben sich auf der Hauptdiagonale Einfachfehler und auf den Nebendiagonalen die Zweifachfehlerkombinationen. Die Untermenge der Anforderungen ist dabei entsprechend der Bedingung und des Prüfkriteriums vom Zielzustand nach dem Eintreten eines Fehlers abhängig. Dabei sind die Reihenfolge der Fehler und deren Auftretenszeitpunkt relevant. Um den Zustandsraum auf die relevanten Aspekte einzugrenzen wurden dies eingeschränkt, wie Abbildung 6.1 zeigt.

Nach dem Start des Model-Checkings ist eine gewisse Hochlaufzeit notwendig bis der Nominalbetrieb aktiv ist. Nach der Aktivierung kann ein Erstfehler für einige Zeitschritte auftreten. Einen Zeitschritt danach beginnt das Intervall für das mögliche Auftreten eines möglichen Zweitfehlers.



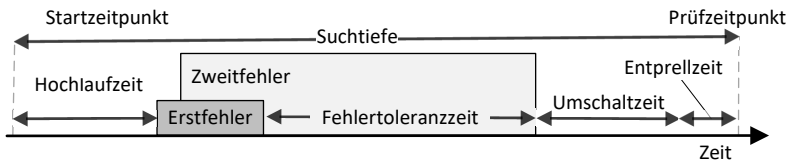


Abbildung 6.1: Definierte Fehlerzeitpunkte im Rahmen der Prüfung

Das Zeitintervall ab dem Zeitpunkt des Erstfehlers entspricht dabei der Fehlertoleranzzeit, im Rahmen dessen eine Umschaltung erfolgen muss. Dadurch wird sichergestellt, dass ein Zweitfehler zu jedem Zeitpunkt der Umschaltung geprüft wird. Sowohl Erst- als auch Zweitfehler können dabei beliebig oft für eine beliebige Zeitdauer anliegen, wobei immer der Erstfehler vor dem Zweitfehler auftritt. Nach dem jeweiligen Fehlerintervall bleibt der Fehlerzustand konstant bis zum Prüfzeitpunkt. Die Prüfung erfolgt in einem stationären Zustand, der nach einer zusätzlichen Umschaltzeit, im Maximalfall die Fehlertoleranzzeit und die Entprellzeit eintritt. Aus der Summe der Einzelzeiten ergibt sich damit auch die Suchtiefe, da das System diskret im Zyklus der Buskommunikation berechnet wird. Durch die Prüfung zu dem diskreten Zeitpunkt, entsprechend der Fehlertoleranzzeit, wird auch das Erreichen des sicheren Zustands innerhalb dieser explizit nachgewiesen.

Implementiert wurde die Auswahl des Modells und der Spezifikationen anhand eines Steuerprogramms, welches für jede Zelle die Fehlerfälle, das Modell und die Spezifikationen entsprechend dem Fehlerfall einliest und integriert. Durch das Steuerprogramm erfolgt auch der Aufruf des Model-Checkers, dem dabei die Parameter für das bounded Model-Checking, der entsprechende Befehl und die Suchtiefe, übergeben werden. Zudem erfolgt eine Parallelisierung, eine Aufteilung der Fehlerfälle auf verfügbare Kerne, um die Rechenzeit zu verkürzen. Die Zustände der Module, der einzelnen Zustandsautomaten, werden dabei synchron bestimmt. Durch die beliebigen Fehlerzeitpunkte und -Kombinationen werden allerdings Übergänge entsprechend einer asynchronen Implementierung erzeugt.

## 6.3 Validierung und Werkzeugqualifizierung

Ziel der Validierung ist der Nachweis, dass Modell und Spezifikationen sowie die Prüfergebnisse des Model-Checkings dem realen Verhalten der Umschaltlogik entsprechen [Sau99].

Die Validierung des Modells und der formalen Spezifikationen wird im Folgenden vorgestellt. Zu diesem Zweck werden Expertenreviews, Fehlerinjektionen und Tests genutzt. Darüber hinaus erfordert die ISO 26262 eine Qualifikation des Werkzeugs bei Verwendung der Ergebnisse als Nachweis in einer Sicherheitsargumentation, welche am Ende des Kapitels adressiert wird.

### 6.3.1 Validierung des Modells

Die Basis für den Abgleich von Modell und Systemverhalten stellt dabei die Matrix in Tabelle 6.1 dar. Diese wurde durch Simulationen und Tests des Programmcodes unter anderem auf der Zielhardware validiert.

Die Validierung des Modelles erfolgt durch das Prüfen des Modellverhaltens beim Auftreten der identifizierten Fehlerfälle. Hierfür wird der Zielbetrieb beim Model-Checking von Einzel- und Zweifachfehlern mit dem in der Matrix beschriebenen Systemverhalten verglichen. Zu diesem Zweck wurde die Spezifikation 6.6 und analog Spezifikationen für jeden Zielbetrieb berücksichtigt, welche beim Auftreten beliebiger Fehler den Zielzustand prüfen. Diese wurden bei jedem Prüflauf des entsprechenden Zielbetriebs integriert, wodurch eine Abweichung vom Systemverhalten detektiert wird. Darüber hinaus wurden die umgesetzten Systemspezifikationen im Modell dokumentiert und das Modell einem Review unterzogen.

Den größten Aufwand stellt die Validierung der formalen Spezifikationen dar. Zu diesem Zweck erfolgt zum einen die Validierung anhand von Fehlerinjektionen, basierend auf Äquivalenzklassentests [Hof13], und zum anderen ein inhaltliches Review zur Validierung der Vollständigkeit. Kritisch bezüglich der Sicherheit sind falsch-negative Prüffälle des Model-Checkings,

also Verletzungen von Anforderungen, welche nicht erkannt werden. Die formulierten Spezifikationen bestehen, wie beschrieben, aus Bedingungen und Prüfkriterien. Für beide Terme ist im Rahmen der Validierung zu zeigen, dass diese korrekt formuliert sind, sodass vorliegende Fehler entdeckt werden. Die Terme müssen daher für die entsprechenden Äquivalenzklassen wahr beziehungsweise falsch sein. Tabelle 6.2 zeigt das Validierungskonzept für die formalen Spezifikationen, basierend auf mehrdimensionale Äquivalenzklassentests [Hof13]. Darauf aufbauend werden Fehlerinjektionen in Form von Manipulationen der Spezifikationen und Reviews genutzt. Die Ableitung von Testfälle erfolgt auch in Literatur durch Negieren der Spezifikationen (vgl. [AHDR18][AVR19a], [AVR19b]).

Tabelle 6.2: Äquivalenzklassenkonzept für die Validierung der formalen Spezifikationen

Bedingung \ Prüfkriterium	wahr	falsch
	wahr	Prüfkriterium & Bedingung
falsch	-	!Prüfkriterium & !Bedingung

Für die Validierung der formalen Spezifikationen wird eine Menge an Fehlerkombination ausgewählt, die alle Spezifikationen abdecken. Im ersten Schritt erfolgt eine Prüfung ohne Fehlerinjektion, eine Manipulation der Spezifikationen, sogenannter Trap-Bedingungen (vgl. [AVR19b]), bei dem diese entsprechend nicht verletzt werden. Im nächsten Schritt wird das Prüfkriterium negiert. Entsprechend der Äquivalenzklasse ergibt sich damit eine Verletzung aller Spezifikationen. Das gilt, wenn für alle Sequenzen, in denen das Prüfkriterium nicht erfüllt wird, die Bedingung wahr ist. Das Vorgehen wird anhand eines Beispiels, der Umschaltung auf den Rückfallkanal, gezeigt. Wie beschrieben stellt der Term  $G ((BR1 | FS1 | FF1 | EPS1 | Fehler_{NK}.Entprellzeit = 3) = Fehler)$  die Bedingung dar. Das Prüfkriterium lautet  $(G [FTTI-5, FTTI+5] (RB2))$ . Eine Manipulation des Zielbetriebs zu  $!RB2$  würde demnach zu einer Verletzung führen, wenn der Term

((BR1 | FS1 | FF1 | EPS1 | Fehler<sub>NK</sub>.Entprellzeit = 3) = Fehler) wahr ist. Der Validierungsschritt zeigt zum einen, dass die Bedingung korrekt-positiv wahr ist und, dass das Zielkriterium das Prüfkriterium bei Nichterfüllung zu einer Verletzung führt. Zu zeigen ist darüber hinaus, dass die Bedingung nur in den relevanten Fällen wahr ist. Dies erfolgt durch die zusätzliche Negierung der Bedingung, was bei den vorherigen Prüffällen, keine Verletzung erzeugen sollte. Eine Negierung der Bedingung ausgehend von einem wahren Prüffall generiert keine Erkenntnis, weswegen auf darauf verzichtet wird. Da die Validierung anhand exemplarischer Prüffälle erfolgt wird auf die Vollständigkeit der Fehlerfälle in den Bedingungen zusätzlich mit einem Review durch Experten bestätigt.

### 6.3.2 Werkzeugqualifizierung nach ISO 26262

Neben der Validierung erfordert die ISO 26262 die Qualifizierung der für den Sicherheitsnachweis verwendeten Software-Werkzeuge [Int18]. Im Folgenden werden der Prozess und die notwendigen Schritte zur Verwendung einer Model-Checking Software für den Sicherheitsnachweis mittels formaler Verifikation vorgestellt.

Jedes Werkzeug, das zur Entwicklung sicherheitsrelevanter Produkte oder zur Erstellung von Arbeitsprodukten im Sicherheitslebenszyklus verwendet wird, muss nachweislich dafür geeignet sein. Fehler dürfen durch das Werkzeug nicht in das Produkt eingebracht werden und das Werkzeug muss einen sicherheitsgerichteten Prozess nach den Vorgaben der ISO 26262 ermöglichen. Zur Ableitung der notwendigen Qualifikationsmaßnahmen wird ein Konfidenz-Level (engl. Tool Confidence Level (TCL)) bestimmt. Dieses ist abhängig von den Auswirkungen eines Fehlers (engl. Tool Impact Level (TIL)) und dessen Entdeckungswahrscheinlichkeit (Detektion, engl. Tool Detection Level (TDL)). [Int18] Abbildung 6.2 zeigt das generische Vorgehen entsprechend ISO 26262 zur Untersuchung und Qualifizierung von Software-Werkzeugen für den Sicherheitslebenszyklus.

Das Werkzeug wird als Black-Box von der Eingabe durch den Anwender bis

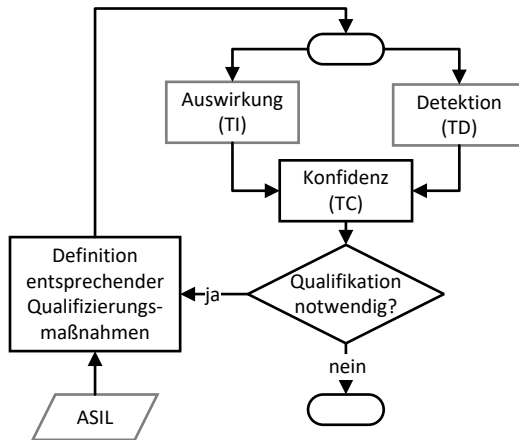


Abbildung 6.2: Vorgehen bei der Analyse und Qualifizierung von Software-Werkzeugen nach ISO 26262 [Int18]

zur Ausgabe betrachtet. Dabei wird die beabsichtigte Verwendung, ohne Fehlbedienung oder inkorrekte Eingangsgrößen, analysiert. Die Auswirkung, das Tool-Impact Level (TCL), ist relevant, wenn das Werkzeug durch eine Fehlfunktion Fehler im Produkt erzeugt oder diese nicht erkannt werden. Das Tool-Detection Level wird in Abhängigkeit der Wahrscheinlichkeit der Fehlervermeidung beziehungsweise der -entdeckung in drei Stufen unterteilt. Die Bewertung erfolgt dabei für entsprechende Fehlfunktionen, abgeleitet von den relevanten Anwendungsfällen. Abhängig von der Bewertung werden Maßnahmen zur Qualifizierung definiert, wobei das notwendige Integritätslevel (ASIL) des Produktes berücksichtigt wird. Diese können aus Entwicklungs- und Absicherungsmaßnahmen bestehen und werden bei einer erneuten Bewertung der Auswirkung der Detektion berücksichtigt.

Der Anwendungsfall des Model-Checking Werkzeugs zur Verifikation der Umschaltlogik ist *das Prüfen des Verhaltens der Zustandsautomaten auf Verletzung von Spezifikationen entsprechend der Zielzustände*. Daraus ergeben sich die folgenden Fehlerfälle:

- **falsch-positive Prüfung**

Das Model-Checking findet Verletzungen, obwohl diese nicht vorliegen.

→ nicht sicherheitskritisch (TI1, TD1 → TCL1)

→ keine Qualifizierung notwendig,

- **falsch-negative Prüfung**

Das Model-Checking findet Verletzungen nicht, obwohl diese vorliegen.

→ sicherheitskritisch (TI2, TD3 → TCL3)

→ Qualifizierung notwendig.

Der falsch positive Prüffall bedeutet einen Mehraufwand bei der Auswertung, ist allerdings nicht sicherheitskritisch. Der falsch negative Prüffall dagegen ist sicherheitskritisch. Sowohl für die Auswirkung, TIL, als auch die Detektionswahrscheinlichkeit, TDL, erfolgt die höchste Einstufung, da bei einer fehlerhaften Analyse Produktfehler unerkannt bleiben und die fehlerhafte Analyse ohne Validierung oder Qualifizierung nicht erkannt wird. Nach der ISO 26262 ergibt sich damit das höchste Konfidenz-Level (TCL3). Zur Qualifizierung ist entweder eine Entwicklung des Software-Werkzeugs nach ISO 26262 unter Berücksichtigung des ASILs oder dessen Validierung erforderlich. [Int18] Da mit NuSMV ein bereits existierendes Werkzeug genutzt wird, wird eine Validierung durchgeführt.

Im Rahmen der Validierung des Software-Werkzeugs wird nachgewiesen, dass die Software den Anforderungen und dem Zweck genügt. Diese unterscheidet sich damit bezüglich des Ziels von der Validierung der Implementierung, wobei die einzelnen Validierungsschritte sich teilweise überschneiden können.

Ausgehend vom kritischen Fehlerfall werden die Fehlerursachen identifiziert. Ursachen für eine falsch negative Prüfung stellen das Nicht-stattfinden des Model-Checkings und die inkorrekte Anwendung der Prüfkriterien dar. Der Nachweis, dass das Model-Checking durchgeführt wurde erfolgt wiederum durch das Prüfen des Zielzustandes (vgl. Spezifikation 6.6), wie im Rahmen der Modell-Validierung bereits adressiert. Dies ist nur unter der Prämisse, dass Fehler auch entdeckt und die Prüfkriterien demnach korrekt ausge-

wertet werden, gültig. Um Dies nachzuweisen, werden Testfälle, basierend auf Fehlerinjektionen, definiert. Dies umfasst die Manipulation der Matrix, um zu validieren, dass das Einlesen und der Abgleich und der Aufruf des Model-Checkers, umgesetzt durch das Steuerprogramm, korrekt erfolgt. Außerdem werden Fehlerinjektionen im Modell der Zustandsautomaten und in der Modellierung der Architektur genutzt. Damit wird nachgewiesen, dass das Model-Checking Werkzeug vorliegende Fehler identifiziert. Ziel der Qualifizierung ist nicht die vollständige Testabdeckung, sondern der Nachweis der Korrektheit des Software Werkzeugs. Auch wenn die Testfälle der Modellvalidierung und die Qualifizierung sich überschneiden, sollte die Dokumentation getrennt erfolgen, um die Abdeckung der ISO 26262 nachvollziehbar zu zeigen. Eine fehlerhafte Anwendung wird durch Einschränkung der Anwendung auf Experten minimiert. Die Validierung des Werkzeugs zeigt eine hinreichende Qualifizierung entsprechend der ISO 26262, weswegen das Werkzeug für den Sicherheitsnachweis genutzt werden kann.

## 6.4 Ergebnisse und Anwendbarkeit des Model-Checkings

Das vorgestellte Vorgehen wurde im Rahmen der Sicherheitsanalyse eines Fail-Operational-Fahrzeug-Systems in der Fahrzeugentwicklung bei der BMW AG, einem Anwendungsfall aus der Industrie, angewandt. Die Ergebnisse und Erkenntnisse bezüglich der Anwendbarkeit werden in diesem Kapitel vorgestellt. Hierfür wird das System und dessen Komplexität anhand der, für das Model-Checking relevanten Eigenschaften, vorgestellt. Im Anschluss wird auf die Ergebnisse der Verifikation mittels Model-Checking eingegangen. Zum Schluss werden die Anwendbarkeit und Einschränkungen der Methode diskutiert.

Im Rahmen dieser Arbeit wurde das Vorgehen zur Analyse einer Umschaltlogik für ein Fail-Operational-Fahrzeugsystem genutzt. Das untersuchte System besteht, analog zu der vorgestellten Architektur (vgl. Abbildung

2.10), aus sieben gekoppelten Zustandsautomaten mit mehr als dreißig einzelnen Zuständen. Die Zustandsautomaten sind jeweils auf einem Steuergerät partitioniert und bilden zwei funktionale Kanäle (vgl. Abbildung 2.9). Im Rückfallbetrieb sind drei verschiedene Systemkonfigurationen möglich, wobei in einem Rückfallbetrieb Systeme beider Kanäle aktiv sind. Damit kann eine Priorisierung der Komponenten berücksichtigt werden [Kue18]. Neben den Kommunikationsverbindungen innerhalb der jeweiligen Kanäle erfolgt eine Kommunikation über Bussysteme zwischen den Kanälen, woraus sich über 25 einzelne Nachrichten, als Kommunikationssignale modelliert, zwischen den Zustandsautomaten ergeben. Inklusiv der Fehler der Energieversorgung ergibt sich eine Größe Fehlermatrix von  $42 \times 42$ . Damit sind 42 Einfachfehler und 1722 Fehlerkombinationen zu prüfen. Die Freiheitsgrade spannen dabei einen Zustandsraum mit  $6 \cdot 10^{40}$  Zuständen auf. Durch die zusätzliche Berücksichtigung der Priorisierung der Aktoren und von Notfallmanövern, welche abhängig von der Fehlerreihenfolge sind, ist die Matrix der Fehler Fehlerkombinationen und Zielzuständen asymmetrisch. Die Anzahl der relevanten Spezifikation variiert zwischen fünf im Falle eines Systemausfalls und sechzehn für Einfachfehler, die zu einem Rückfallbetrieb führen.

Mittels Model-Checking konnte die Verifikation für einen Sicherheitsnachweis der Umschaltlogik gemäß ISO 26262 durchgeführt werden. Damit wurde der Nachweis erbracht, dass die Logik der Zustandsautomaten den formulierten Sicherheitsanforderungen entspricht, was durch funktionale Tests bestätigt wurde. Alle im Rahmen der Validierung induzierten Fehler führten zu entsprechenden Verletzungen von formalen Spezifikationen, womit die korrekte Implementierung nachgewiesen wurde. Die Erfüllung der Norm und die Konformität des Vorgehens wurde durch ein unabhängiges Review und Validierungen bestätigt. Neben diversen Verletzungen, deren Ursache in der Formulierung der Sicherheitsanforderungen lag und deren iterative Überarbeitung erforderte, wurde eine Sequenz identifiziert, welche eine Verletzung der Spezifikation zur Folge hat. Bei diesem Fehlerfall handelte es sich um eine mögliche Aktivierung beider Kanäle, sofern die Kommunikation zwischen den Kanälen abbricht und zu dem Zeitpunkt



wieder einsetzt, zu dem die Aktivierungsanforderung an den Nominalkanal gesendet wird. Der Rückfallkanal liest den Zustand des Nominalkanals vor der Aktivierung ein und wechselt ebenfalls in *Aktiv*. Eine weitere Analyse der Sequenz hat gezeigt, dass die Eintretenswahrscheinlichkeit gering ist, hat aber zu einer Korrektur in der Software geführt. Zusätzlich konnten im Rahmen der Modellierung unklare Formulierungen der Anforderungen und entsprechende Anpassungen identifiziert werden, wodurch die Qualität dieser verbessert wurde.

Die Evaluation der Anwendbarkeit stellt ein weiteres Ergebnis der Fallstudie dar und wird durch den Aufwand der Implementierung und die Rechenzeit bestimmt. Die Modellierung der Zustandsautomaten stellt den geringsten zeitlichen Aufwand dar, da die zustandsorientierte Systembeschreibung eine einfache Übertragung in die Syntax der Modellierungssprache ermöglichte. Die Modellierung der Architektur erforderte ein generisches Konzept und eine entsprechende Abstraktion, konnte aber auf alle relevanten Fehler analog übertragen werden. Die Übersetzung der prosaisch formulierten Anforderungen in linear temporale Logik (LTL), die Validierung und Optimierung stellen, wie in der Literatur (vgl. Kapitel 3.3.1) beschrieben, den größten zeitlichen Aufwand dar. Für die Formulierung der Spezifikationen sind sowohl Systemverständnis als auch Kenntnisse des Model-Checking Verfahrens notwendig. Außerdem besteht eine Abhängigkeit der Formulierung von der Modellierung, beispielsweise von Variablen, was bei Änderungen im Modell auch Anpassungen der Spezifikationen erfordert. Zum Nachweis der Korrektheit wurden die Spezifikation, mitunter iterativ, angepasst und validiert. Aufgrund der Komplexität des untersuchten Systems waren mitunter Detaillierungen der Spezifikationen und die Berücksichtigung von Sonderfällen, welche beispielsweise durch Reaktionszeiten entstehen können, notwendig. Zusätzliche Konzepte, wie zeitgesteuertes Auftreten der Fehler, und die Optimierung des Model-Checking Ansatzes, beispielsweise die Verwendung des bounded Model-Checking, haben im Rahmen der Implementierung zusätzlichen Aufwand generiert. Hierbei können die Erkenntnisse dieser Arbeit jedoch als Basis für zukünftige Arbeiten dienen.

Die Verteilung der Rechenzeiten bei den Prüfungen von Einfach- beziehungsweise Zweifachfehler werden in Abbildung 6.3 und 6.4 anhand von Kastendiagrammen gezeigt. Dabei wurden alle Einfachfehler und 40 Zweifachfehler ausgewählt, welche die Zielbetriebe zu gleichen Anteilen berücksichtigen. Für alle Prüffälle erfolgte die Verifikation vollständig. Das genutzte System verfügte über 6 Kerne (4 GHz, 16 GB RAM, Windows 10, 64 bit), die parallel für die Prüfung der einzelnen Spezifikationen des Fehlerfalls genutzt wurden.

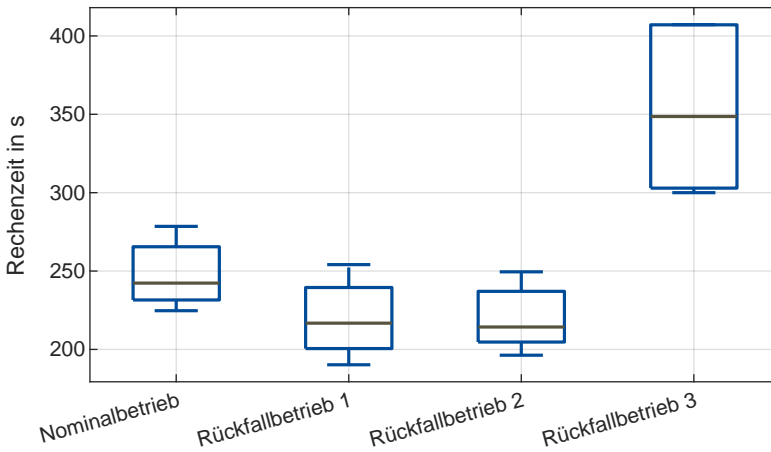


Abbildung 6.3: Verteilung der Rechenzeiten beim Prüfen von Einfachfehler in Abhängigkeit des Zielbetriebs

Dabei sind die Rechenzeiten in Sekunden vertikal aufgetragen. Die Verteilung der Rechenzeiten für die einzelnen Betriebsmodi ist anhand von Kastendiagrammen dargestellt. Einfachfehler führen nicht zum Systemausfall und sind daher in Abbildung 6.3 nicht dargestellt. Die Ursachen der Rechenzeitunterschiede sind zum einen die formalen Spezifikationen und zum anderen die Zustandssequenzen. Die Verteilung der Rechenzeiten bei Einfachfehler zeigt nur geringe Unterschiede, für die einzelnen Zielbetriebe. Für die Umschaltung in die Rückfallbetriebe 1 und 2, sowie keine Umschal-

tung ergeben sich durchschnittliche Rechenzeiten zwischen 200 und 250 s. Die maximalen Abweichungen liegen dabei unter 30 s. Die Prüfung von Umschaltungen in den Rückfallbetrieb 3 erfordern längere Rechenzeiten, da längere Umschaltsequenzen erforderlich sind und zeigt Abweichungen von 50 s.

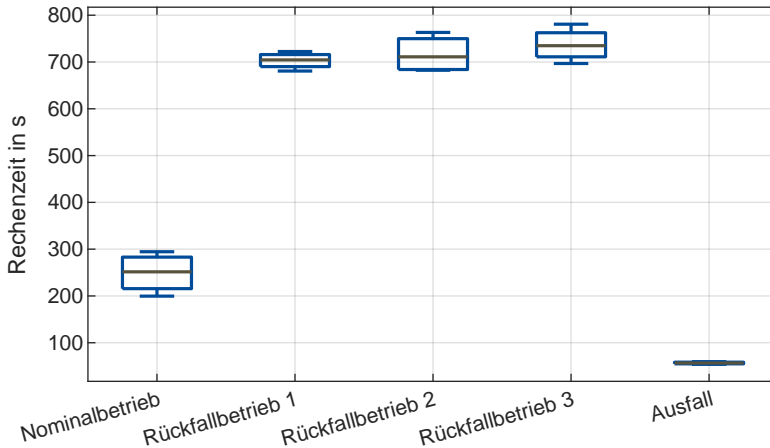


Abbildung 6.4: Verteilung der Rechenzeiten beim Prüfen von Zweifachfehler in Abhängigkeit des Zielbetriebs

Im Falle der Zweifachfehler ergeben sich längere Rechenzeiten, bedingt durch im allgemeinen längere Sequenzen bei Umschaltungen. Zudem sind deutlichere Unterschiede abhängig vom Zielbetrieb zu sehen. Diese sind wiederum bedingt durch die Länge der Zustandssequenzen und die Anzahl und den komplexeren Aufbau der formalen Spezifikationen. Fehlerfälle, die zum Ausfall führen werden innerhalb von circa 50 s geprüft. Die Streuung liegt dabei unter 1 s. Die Berechnung von Fehlerfällen, die nicht zu einer Umschaltung führen erfordert eine relative geringe Rechenzeit, Vergleich mit der Rechenzeit bei Einfachfehler. Dies liegt daran, dass auch die Umschaltsequenzen vergleichbar sind. Prüffälle, die zu einer Umschaltung führen, erfordern eine Rechenzeit zwischen 680 und 780 s (oder 11 und 13

Minuten). Deutliche Unterschiede zwischen den Zielbetrieben zeigen sich dabei nicht, wobei diese im Mittel von den Zielbetrieben Rückfallbetrieb 1 über den Rückfallbetrieb 2 und Rückfallbetrieb 3 hinweg steigen. Dies ist aufgrund der steigenden Länge der Umschaltsequenzen plausibel. Da der Anstieg der Rechenzeiten aufgrund der Umschaltsequenzen jedoch im Vergleich zu deren absoluten Werten relativ gering ausfällt, legt dies den Schluss nahe, dass die Auswirkungen der Spezifikationen, welche für die Rückfallbetriebe analog formuliert werden, größer ist. Dabei ist zu berücksichtigen, dass die Rechenzeit exponentiell steigt, wenn der Arbeitsspeicher die Kapazitätsgrenze erreicht, da im Werkzeug dann Zustände auf den Festspeicher ausgelagert werden. Dies ist aber in keinem Verifikationsfall relevant gewesen.

Insgesamt zeigten sich Rechenzeiten im Fallbeispiel akzeptabel für einen Nachweis der funktionalen Sicherheit. Eine entwicklungsbegleitende, direkte Prüfung von Änderungen dagegen ist ohne weitere Optimierung nicht möglich, da eine komplette Prüfung circa zwei Tage erfordert. Eine Optimierung der Rechenzeit allerdings stand nicht im Fokus der Arbeit. Für eine zyklische Integration der Prüfung in die Entwicklung stellt die Rechenzeit kein Hindernis dar.

Zusammenfassend hat sich gezeigt, dass die Methode für die Verifikation und den Nachweis der funktionalen Sicherheit von Fail-Operational-Umschaltlogiken gemäß ISO 26262 geeignet ist und der Sicherheitsnachweis für ein reales und komplexes System erbracht werden kann. Die Anzahl der Fehler und Varianten sowie die Komplexität können mit Model-Checking adressiert und das System analysiert werden. Es konnte das Erreichen eines Zielzustands innerhalb der Fehlertoleranzzeit nachgewiesen werden. Der Aufwand verlagert sich im Vergleich zu einer FMEA von der Identifikation der Fehlerkette, auf die Modellierung und Validierung, erfordert mit Ausnahme des Reviews keine Beteiligung mehrerer Experten und spart daher Kapazitäten. Die Qualität der Anforderungen konnte durch deren Übersetzung in formale Spezifikationen verbessert werden.

Nichtsdestotrotz unterliegt das Vorgehen einigen Einschränkungen, welche

bei der Diskussion der Generalisierbarkeit beachtet werden müssen. Der Modellierungsaufwand war unter anderem begrenzt, weil die Umschaltlogik als Zustands-basiertes System der Syntax des Model-Checking Werkzeugs entsprochen hat und der notwendige Abstraktionsgrad daher gering war. Sofern dies nicht der Fall ist, steigt nicht nur der Aufwand, sondern auch das Risiko einer fehlerhaften Modellierung. Analoges gilt auch für die Definition der formalen Spezifikation, welche auch im Anwendungsfall einen hohen Zeitaufwand erforderten. Die ISO 26262 empfiehlt den Einsatz formaler Methoden für ASIL C und D Systeme, fordert diesen aber nicht zwingend. Die Vorteile eines Korrektheitsnachweises müssen für jedes System gegenüber dem Aufwand unter Berücksichtigung möglicher Alternativen abgewogen werden. Wie von Vollmer [Vol20] dargelegt, ist dies im Rahmen hochautomatisierter Fahrfunktionen und insbesondere für ASIL C und D Umfänge sinnvoll. Auch die mögliche Wahl des Model-Checking Ansatzes stellt eine Einschränkung der Generalisierbarkeit dar. Im Anwendungsfall konnte aufgrund des stationären Systemverhaltens eine Grenze der Suchtiefe bestimmt und damit bounded Model-Checking verwendet werden. Dies war eine notwendige Voraussetzung für das Verhindern der Zustandsraumexplosion, welche eine vollständige Verifikation unter Umständen verhindert. Im Rahmen dieser Arbeit wurde dabei ein möglicher Lösungsweg für die Verifikation einer Fail-Operational Fahrzeugführung aufgezeigt. Die Optimierung der Implementierung, der Rechenzeiten oder die Wahl des Werkzeugs stellen weitere Forschungsmöglichkeiten dar, die bisher nicht erschöpfend behandelt wurden und bei der Verifikation weiterer Systeme berücksichtigt werden können. Bezüglich der Generalisierbarkeit müssen auch notwendige Abstraktionen bei der Modellierung und deren Auswirkungen auf die Validität und Aussagekraft der Ergebnisse berücksichtigt werden. Insbesondere die Berücksichtigung des zeitlichen Verhaltens verteilter Systeme erfordert eine Abstraktion. Im Anwendungsfall erfolgt die Berechnung der einzelnen Module und der Kommunikation synchron, was gegenüber dem asynchronen Verhalten im realen System eine Vereinfachung darstellt. Auf Basis einer Expertenscheinschätzung wurde die Näherung durch Implementierung beliebiger Fehlerkombinationen und Fehlerzeitpunkten als ausreichend bewertet.

Entsprechende Vereinfachungen müssen abhängig vom Anwendungsfall bewertet werden. Weiterhin basiert das berücksichtigte Fehlerverhalten auf Sicherheitskonzepten der Komponenten, die eine Reaktion, beispielsweise ein Abschalten, entsprechend nachweisen müssen. Fehler der unterlagerten Sicherheitskonzepte, würden sich in der Analyse nicht zeigen, jedoch die Validität der Ergebnisse einschränken. Bei der formalen Verifikation handelt es sich darüber hinaus um eine theoretische Sicherheitsanalyse auf Basis funktionaler Beschreibungen und Spezifikationen. Diese ersetzt keine Absicherung unter Verwendung der Zielhardware, welche im Industriestandard in jedem Fall zusätzlich gefordert wird [Int18].

KAPITEL 

# FORMALER NACHWEIS DES VERHALTENS IM SYSTEMVERBUND

Der Nachweis der funktionalen Sicherheit in einem Fail-Operational-System erfordert den Nachweis der Unabhängigkeit von Nominal- und Rückfallkanal (vgl. Kapitel 4.4). Neben der in Kapitel 5 bereits adressierten Analyse der Unabhängigkeit bezüglich gemeinsamer Ausfälle ist eine Analyse von Ausfällen durch Fehlerpropagationen für den Nachweis der funktionalen Sicherheit von Fail-Operational-Systemen notwendig (vgl. Kapitel 4.4). Ein Fehler, der von einem Kanal in den anderen Kanal propagiert, kann bei entsprechender Diagnose und Meldung an die Umschaltlogik zur Deaktivierung beider Kanäle führen und muss daher verhindert werden. Die entsprechende Analyse muss im Verbund erfolgen, da die Propagation über mehrere Funktionen und Komponenten hinweg möglich ist.

Im Rahmen der Analyse wird damit das Fail-Operational-Verhalten unter Berücksichtigung des Systemverbunds verifiziert, das dem Übergang in einen sicheren Zustand unter Einhaltung der Fehlertoleranzzeit auf Systemebene entspricht.

Eine Fehlerpropagation erfordert dabei eine Kopplung der Systeme und Komponenten [Sch16]. Im gezeigten Fail-Operational Systemen, wie auch in alternativen Architekturen, wird redundante Hardware genutzt (vgl. Kapitel 2.3). Die Energieversorgung erfolgt getrennt in den jeweiligen Kanälen. Auch weitere Hardware-Umfänge, wie Speicher und Prozessoren sind Steuergeräte-spezifisch umgesetzt. Damit stellen entsprechend des Abhängigkeitsmodells von Schnellbach (vgl. Abbildung 3.1) die Kommunikation und geteilte Informationen die relevanten Fehlerpropagationsmechanismen dar, welche in der Analyse adressiert werden müssen.

Im diesem Kapitel wird eine Erweiterung des Model-Checking Ansatzes aus Kapitel 6 zur Analyse der Fehlerpropagation und damit zum Nachweis des Fail-Operational-Verhaltens vorgestellt. Das Fehlerverhalten von Softwarekomponenten und die Fehlerpropagation wird anhand der Signalübertragung zwischen den Softwarekomponenten modelliert (vgl. Kapitel 7.1). Im Anschluss wird ein rekursiver On-the-Fly Model-Checking Algorithmus, mit dem der Systemzustand anhand der Zustände der Umschaltlogik geprüft wird, vorgestellt. Die Implementierung erfolgt dabei objektorientiert in Python<sup>1</sup> ohne Bindung an ein vorhandenes Werkzeug. Die Validierung in Kapitel 7.3 erfolgt anhand eines Minimalbeispiels. Die Analyse einer realen Signalwirkkette des Fahrzeugsystems mittels der Model-Checking Methode in Kapitel 7.4 zeigt die Anwendbarkeit der Methode. Außerdem werden Systemoptimierungen für eine höhere Robustheit bezüglich Verletzungen des Fail-Operational-Verhaltens abgeleitet.

---

<sup>1</sup><https://www.python.org/>



## 7.1 Modellierung des Systemverbunds

Die Modellierung des Systemverbunds umfasst das für die Fehlerpropagation relevante Verhalten des Systems inklusive der Architektur und stellt den ersten Schritt der Analyse dar (vgl. Kapitel 2.4.3 und 3.4). Gegenüber der Analyse der Umschaltlogik in Kapitel 6 werden die Softwaresignale und Elemente berücksichtigt, die die Signalverarbeitung, Software-Komponenten der Applikationssoftware, darstellen. Abbildung 7.1 zeigt das Klassendiagramm des relevanten Systemverbunds entsprechend der UML-Notation [Gol11].

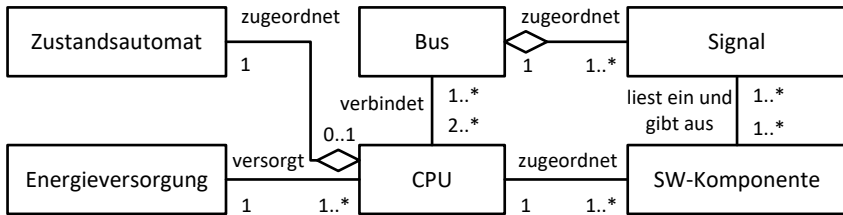


Abbildung 7.1: Klassendiagramm des für die Fehlerpropagation relevanten Systemmodells

Central Processing Units (CPUs) stellen die Rechenplattformen von Steuergeräten dar. Auf den für die Fahrzeugführung unbedingt notwendigen CPUs ist, wie beschrieben, jeweils ein Zustandsautomat partitioniert (vgl. Kapitel 2.3). Darüber hinaus können auch weitere Steuergeräte, wie eine Hinterachslenkung oder Sensorik, die Signale an diese senden, modelliert werden. Software-Komponenten (SW-Komponenten) sind einer CPU zugeordnet. Im Gegensatz zu den Zustandsautomaten können mehrere SW-Komponenten auf einer CPU partitioniert sein. Eine CPU ist an mindestens einen Bus angebunden, der mindesten zwei CPUs verbindet. Jede CPU hat eine Anbindung an die Energieversorgung. Signale stellen die Information dar, welche zwischen Software-Komponenten über einen Bus ausgetauscht werden und von diesen verarbeitet werden.

Im Folgenden wird die Modellierung der einzelnen Klassen vorgestellt. Insbesondere ist das Verhalten der Software-Komponenten und die Kommunikation für die Signalpropagation relevant. Die Modellierung der Software-Komponenten erfolgt anhand modularer Fehlerbäume in Kapitel 7.1.1. Im Anschluss wird in Kapitel 7.1.2 die Modellierung der Architektur und der Kommunikation inklusive des zeitlichen Verhaltens vorgestellt.

### 7.1.1 Software-Komponenten als modulare Fehlerbäume

Die Modellierung des Fehlerverhaltens der Software-Komponenten erfolgt auf Basis modularer Fehlerbäume. Zu diesem Zweck werden die relevanten Signalarten und die entsprechenden Fehlermodi identifiziert und der Aufbau der modularen Fehlerbäume sowie die Zusammenhänge der Fortpflanzung inklusive des zeitlichen Verhaltens abhängig von Signalart und Fehlermodi vorgestellt.

Im Software-Verbund werden Nutzsignale und Qualifier-Signale unterschieden. Nutzsignale stellen Signale dar, deren Informationen zur Bestimmung funktionaler Stellgrößen verwendet werden. Beispiele solcher Nutzsignale sind kontinuierliche Signale, wie ein Geschwindigkeitssignal, und diskrete Signale wie die Zustände der Zustandsautomaten oder Aktivierungsanforderungen, auf Basis derer ein Ausgangssignal bestimmt wird. Darüber hinaus werden Qualifier-Signale verwendet, auf Basis derer keine direkte Berechnung erfolgt, sondern Informationen über den Zustand eines Nutzsignals übermittelt werden. Diese stellen diskrete Abstufungen der Signalgüte oder deren Integrität dar. Ein Qualifier-Signal ist dabei einem Nutzsignal zugeordnet.

Im nächsten Schritt werden die relevanten Fehlermodi der Signale identifiziert. Die Verifikation des Fail-Operational Verbundes erfolgt auf Basis der Zustände der Umschaltlogik, die dem Systemzustand des Fail-Operational-Systems entsprechen. Da diese durch diskrete Meldung ausgelöst werden, werden verschiedene Abweichungen äquivalent behandelt und die Fehlermodi können entsprechend diskretisiert werden. Analoge Vorgehen werden

auch in der Literatur präsentiert (vgl. Kapitel 3.4, [RPV+01]). Im Sollzustand befinden sich die Werte eines Nutzsymbols innerhalb der Spezifikationen. Die Spezifikation berücksichtigen dabei die Anforderungen an das Signal im, für das Fail-Operational-Verhalten relevanten, hochautomatisierten Fahrbetrieb. Abweichungen des Signals außerhalb dieser Spezifikationen stellen dabei ein fehlerhaftes, inkorrektes Verhalten dar. Durch Fehler, wie Ausfälle von Sendern oder Nachrichtenverfälschung und entsprechenden Sicherheitsmaßnahmen (engl. Cyclic-Redundancy-Check (CRC)), ist es außerdem möglich, dass kein verwertbares Signal beim Empfänger ankommt. Daher werden für Nutzsymbole die Modi *korrekt*, *inkorrekt* und *kein Signal* berücksichtigt. Für Qualifier-Symbole werden die Zustände *valide*, *nicht valide* und *kein Signal* definiert. Analog zu den Nutzsymbolen, gibt es eine Mindestsignalgüte oder Mindestintegrität für den hochautomatisierten Fahrbetrieb. Wird diese erfüllt ist das Signal *valide*, andernfalls ist das Signal *nicht valide*. Analog würde eine Signalverfälschung durch end-to-end Absicherung, ebenso wie Ausfälle des Busses oder des Senders zum Fehlermodus *kein Signal* führen. Ein fälschlicherweise als *valide* oder *invalide* gekennzeichnetes Signal muss mit entsprechender Integrität auf Komponentenebene verhindert werden und wird im Verbund nicht betrachtet.

Die Modellierung der Fehlerlogik von Software-Komponenten erfolgt mittels modularer Fehlerbäume. Diese bieten die Möglichkeit der Kapselung einzelner Module, wodurch diese in einer Fehlerlogik mehrfach berücksichtigt werden können (vgl. Kapitel 3.4). Somit bietet der Ansatz die Möglichkeit einer übersichtlichen Modellierung der Software-Architektur und einer zyklischen Berechnung des Verhaltens im Software-Verbund. Der modulare Fehlerbaum bildet das Verhalten der Software-Komponente von deren Eingang zum Ausgang abhängig von der Signalart und dem Fehlermodus ab. Für die Modellierung der Software-Komponenten sind dabei drei logische Verknüpfungen relevant: Das logische *Und*, das *Oder* und ein *bedingtes Oder*. Letzteres stellt dabei keine logische Verknüpfung im eigentlichen Sinne dar, da das *bedingte Oder* durch *Und*- und *Oder*-Kombinationen darstellbar ist, aber aufgrund der mehrfachen Relevanz individuell modelliert

wurde. Abbildung 7.2 zeigt die Logik der modularen Fehlerbäume für Softwarekomponenten mit *Oder*-Kombinationen der Eingänge.

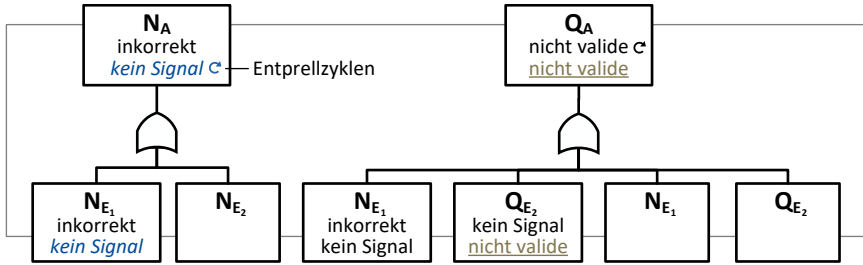


Abbildung 7.2: Logik der modularen Fehlerbäume für Software-Komponenten

Dargestellt ist die Fehlerlogik anhand einer Software-Komponente mit je einem Nutzsignal ( $N_A$ ) und einem Qualifier-Signal ( $Q_A$ ) als Ausgang. Eingänge stellen je zwei Nutzsignale ( $N_{E_{1/2}}$ ) und zwei Qualifier-Signale ( $Q_{E_{1/2}}$ ) dar. Die Zusammenhänge der Fehlermodi zwischen Eingang und Ausgang sind durch identische Formatierungen dargestellt, der fehlerfrei Zusammenhang wird nicht dargestellt. Die Bestimmung der Nutzsignale erfolgt anhand von Nutzsignalen und wird unabhängig von Qualifier-Signalen berechnet. Der Fehlermodus *inkorrekt* eines Nutzsignal am Ausgang entsteht durch ebendiesen Fehlermodus an einem Eingangssignal. Analog verhält sich die Fortpflanzung, wenn ein Nutzsignal nicht empfangen wird (*kein Signal*). Dabei erfolgt zur Erhöhung der Robustheit, analog den Zustandsautomaten (vgl. Kapitel 6.1), eine Entprellung für eine bestimmte Anzahl an Zyklen. Im Falle eines ausbleibenden Nutzsignals am Eingang wird erst nach der Entprellzeit kein Nutzsignal mehr ausgegeben. In der Zwischenzeit erfolgt die Berechnung mittels Ersatzwerten, beispielsweise mit dem letzten gültigen Wert. Für die Bestimmung des Qualifier-Signals sind sowohl Nutz- als auch Qualifier-Signale relevant. Das Qualifier-Signal zeigt dem Empfänger an, dass das Signal den Güteanforderungen nicht mit ausreichender Integrität entspricht. Ein Qualifier-Signal gibt den Wert *nicht valide* aus, sofern bei einem Nutzsignal am Eingang *inkorrekt* anliegt oder dieses, respektive

ein Qualifier-Signal nicht empfangen wird. Dies erfordert eine Diagnose beziehungsweise eine Entprellung mit einer entsprechenden Zyklenzahl. Wenn der Software-Komponente durch ein Qualifier-Signal am Eingang gemeldet wird, dass das Signal *nicht valide* ist, wird diese Information direkt innerhalb eines Rechenzyklus am Ausgang weiter gegeben. Zusätzlich zu der dargestellten Logik werden redundante Signale, beispielsweise mehrere Sensoreingänge, mittels logischer *Und*-Verknüpfungen modelliert. Im Falle der *bedingten Oder*-Verknüpfungen werden, entsprechend einer Signalumschaltung, abhängig vom Auslöser nur Teile der Oder-Verknüpfung berücksichtigt, Umschaltbedingungen stellen dabei Qualifier-Signale dar. Implementiert werden die Verbindungen durch Wenn-Dann-Konstrukte. Entsprechend der definierten Fehlermodi kann für jede Nachricht, Nutzsignal und Qualifier-Signal, jeder Fehlermodus in beliebiger Fehlerkombination injiziert und analysiert werden.

Fehler der Software-Komponenten, beispielsweise ein Abbruch, führen zum Abbruch der Berechnung. Berücksichtigt wird dies, indem die Software-Komponente keine Signale mehr ausgibt, beziehungsweise alle Ausgangssignale im selben Rechenzyklus auf den Fehlermodus *kein Signal* gesetzt werden. Die funktionale Meldung an die Zustandsautomaten erfolgt anhand von Qualifier-Signalen am Ausgang. Sofern ein Signalzustand *nicht valide* oder äquivalent, nach einer Entprellung, *kein Signal* vom Zustandsautomaten empfangen wird, erfolgt die Transition im zugehörigen Zustandsautomat. Die Zustandsautomaten werden gesondert, vergleichbar zu Kapitel 6.1.1 modelliert, die Fehlerfälle entsprechen denen der Software-Komponenten.

### 7.1.2 Modellierung der Architektur und Kommunikation

Neben den Softwarekomponenten müssen auch weitere, für die Fehlerpropagation relevante, Umfänge im Modell berücksichtigt werden. In diesem Kapitel wird zuerst auf die Modellierung des Fehlerverhaltens der Hardwarearchitektur, welches größtenteils analog zu Kapitel 6.1 berücksichtigt wird, und im Anschluss auf die Modellierung der Kommunikation ein-

gegangen. Bezüglich der Kommunikation ist insbesondere das zeitliche Verhalten, mit der Latenz- und Übertragungszeit und deren Divergenz, relevant für die Fehlerauswirkung, da es zwischen Fehlerfortpflanzung und Sicherheitsmechanismen, wie der Umschaltung und Deaktivierung, zu Wettlaufbedingungen (engl. Race-Conditions) kommen kann [Fri21]. Außerdem sind die Latenzzeiten relevant für die Analyse der Einhaltung der Fehlertoleranzzeit.

Die Hardwarearchitektur und die daraus resultierenden Fehlerfälle werden analog zum Model-Checking Ansatz für die Verifikation der Umschaltlogik in Kapitel 6.1 berücksichtigt. Relevant sind komplette und lokale Ausfälle der Energieversorgung, ebenso wie Ausfälle der Steuergeräte und Kommunikationsbusse. Dies wird über die Deaktivierung aller gesendeten Nachrichten anhand des entsprechenden Fehlermodus, der Zuordnung der Signale zu Software-Komponenten und deren Zuordnung zu Steuergeräten modelliert (vgl. Abbildung 7.1). Ausfälle von Busverbindungen führen zum Verlust aller Nachrichten auf dem jeweiligen Bus.

Die Kommunikation der einzelnen Software-Komponenten erfolgt sowohl innerhalb eines Steuergeräts (intra-CPU-Kommunikation), als auch Steuergeräte übergreifend (inter-CPU-Kommunikation). In Fahrzeugführungen und Fahrwerks-Systemen wird üblicherweise eine Sender-Receiver Kommunikation genutzt. Dabei stellt die Sender-Software-Komponente Signale kommunal bereit und die Empfänger (engl. Receiver) entscheiden individuell über die Nutzung der Information. [AUT06, Rei14]

Bei der inter-CPU-Kommunikation werden die Signale innerhalb eines Steuergeräts bereitgestellt. Ausgangssignale werden über einen virtuellen Bus, das Runtime Environment (RTE), zur Verfügung gestellt [AUT06]. Im darauffolgenden Rechenzyklus lesen die empfangenden Software-Komponenten die Signale ein. Latenzen sind dabei für die Analysen nicht von Bedeutung, da diese deutlich unter der Taktzeit der Rechenzyklen der CPUs liegen, sofern jede Software-Komponente innerhalb eines Zyklus maximal einmal berechnet wird.

Intra-CPU Kommunikation bezeichnet die Nachrichten Kommunikation über einen Bus. Busübertragungen können deterministisch, wie beim Flexray, oder zufällig, basierend auf einer Arbitrierung, wie beispielsweise bei CAN-Bus Varianten erfolgen [Rei14, ST12]. Für die Übertragungszeit ist neben der Busübertragung die Synchronisierung zwischen den Bussen und den CPUs relevant [Fri21]. Für die unterschiedlichen Konstellationen werden ausgehend von allgemeinen Zusammenhängen die minimalen und maximalen Übertragungszeiten auf Basis von Berechnungen und Expertenabschätzungen bestimmt. Diese stellen, im Verlauf durch Analysen bestätigt, in Bezug auf Wettlaufbedingungen den kritischen Fall dar. Die Übertragungszeit ( $T_{Ue}$ ), das Zeitintervall vom Ausgeben beim Sender bis zum Einlesen einer Information durch die empfangende Software-Komponente, ergibt sich durch die Wartezeit auf einen Sendeplatz im Buszyklus ( $T_{W_{Bus}}$ ), die Übertragungszeit des Busses ( $T_{Ue_{Bus}}$ ) und die Verarbeitungszeit beim Empfänger ( $T_{Ue_E}$ ).

$$T_{Ue} = T_{W_{Bus}} + T_{Ue_{Bus}} + T_{Ue_E}$$

Die Übertragungszeit des Busses und Verarbeitungszeit beim Empfänger liegen zusammen innerhalb eines Rechenzyklus, weswegen auch eine Synchronisation zwischen Bus und Empfänger keine Auswirkungen hat. Die Wartezeit variiert mit der Bus- und Steuergerätekonstellation. Die minimalen Übertragungszeiten ergeben sich sowohl im deterministischen Fall als auch im nichtdeterministischen Fall des Busses bei einem direkten Versenden und Verarbeiten durch den Empfänger. Für alle Konstellationen wurde daher ein Rechenzyklus als minimale Übertragungszeit gewählt.

Bei einem deterministischen Bus mit synchronisiertem Sender ist in jedem Buszyklus ein Sendeplatz reserviert. Die maximale Übertragungszeit ergibt sich damit durch den Buszyklus, der Übertragungs- und Empfangszeit abzüglich der Rechenzeit des Senders. Bei 10 ms Buszykluszeit und 1 ms Rechenzeit sowie Übertragungs- und Empfangszeit ergeben sich 10 ms. Sofern keine Synchronisation zwischen Sender und Buszyklus implementiert ist, entspricht die maximale Wartezeit dem kompletten Buszyklus. Für den nicht deterministischen Fall kann keine Synchronisation erfolgen. Daher ist

die maximale Übertragungszeit wiederum die Summe aus dem maximalen Buszyklus, der auch im nichtdeterministischen Fall gewährleistet werden muss, und der Übertragungszeit.

Die Implementierung der Kommunikation erfolgt über Stapelspeicher (engl. Stacks) mit Transitionen nach dem Last-In-First-Out-Prinzip. Signale werden zu den Stapeln inklusive deren Berechnungszeitpunkt nach jedem Rechenzyklus hinzugefügt. Durch Abgleich des Bereitstellungszeitpunkts und der relevanten Übertragungszeit ( $T_{Ue}$ ) mit der globalen Rechenzeit wird über die Propagation der Nachrichten entschieden. Die möglichen Transitionen werden dabei zu jedem Rechenzyklus aktualisiert, wobei der Bereitstellungszeitpunkt sich erst bei einem erfolgreichen Senden aktualisiert. Somit wird immer die aktuelle Nachricht über den Bus versendet. Dabei werden Nutzsignale und die respektiven Qualifier-Signale zusammen, einem Nachrichtenpaket entsprechend, versendet. Die Berechnung des Modells erfolgt dabei wiederum kaskadiert über die Zuordnungen entsprechend Abbildung 7.1. Aus dem Hauptprogramm werden alle CPUs aktualisiert, daraufhin alle partitionierten Software-Komponenten und die zugeordneten Signal-Transitionen bestimmt.

## 7.2 Analyse der Umschaltung im Verbund

Die Analyse der Fehlerpropagation und damit die Verifikation des Fail-Operational-Verhaltens erfolgt mittels explizitem Model-Checking. Hierfür werden zuerst die relevanten Zustandsräume, der Prüfraum und der Fehlerraum, abgeleitet und erweitert ausgehend von Kapitel 6.1 und 6.2, definiert. Für die spezifische Analyse unter Eingrenzung der Fehler und Prüffälle wurde ein individueller Algorithmus an Stelle eines verfügbaren Model-Checking Werkzeug genutzt, welcher Freiheiten bezüglich der Modellierung und modellabhängigen Optimierung der Suche bietet. Damit wurde allerdings auch auf die Verwendung entsprechend leistungsoptimierter Suchalgorithmen verzichtet. Dies stellt jedoch bezüglich der Aussage zur generellen Lösbarkeit des Problems keine Einschränkung dar. Der verwendete Algorithmus basiert



auf einem rekursiven On-the-Fly Model-Checking Ansatz und wird inklusive der Optimierungsansätze in diesem Kapitel vorgestellt.

Der Fail-Operational-Betrieb, welcher durch die Fehlerpropagation nicht beeinträchtigt werden darf, wird durch die Umschaltlogik und deren Zustände definiert. Diese steuert die Aktivierung und Deaktivierung der Funktionen. Die Meldung an die Zustandsautomaten in den relevanten Fehlerfällen wird auf der Ebene der jeweiligen Funktionen analysiert und sichergestellt und kann daher als Prämisse genutzt werden. Die Fehlerfreiheit der Logik wurde durch die Verifikation in Kapitel 6 adressiert. Die in Kapitel 6.1 vorgestellten Spezifikationen gelten daher analog für das Verhalten und damit die Analyse des Verbunds. Die Verifikation der Umschaltlogik hat dabei das Fail-Operational-Verhalten unter der Prämisse der Unabhängigkeit der Fehler gezeigt. Im Rahmen der Verbundanalyse ist nachzuweisen, dass Fehlerpropagation zu ebendiesem Verhalten führt, weswegen eine Prüfung des Zielbetriebs (vgl. Tabelle 6.1 und Spezifikation 6.6) hinreichend für den Nachweis des Fail-Operational-Verhaltens ist. Die Prüfung des korrekten Zustands nach der Fehlertoleranzzeit, was anhand eines globalen Zählers geprüft wird, zeigt dabei außerdem deren Einhaltung im Systemverbund. Kritisch für gemeinsame Ausfälle von Nominal- und Rückfallkanal sind Einfachfehler, welche zur Deaktivierung mehrerer Betriebsmodi entgegen des Umschaltkonzepts (vgl. Kapitel 6.2) beziehungsweise beider Kanäle führen. Darüber hinaus ist auch gemäß ISO 26262 die Berücksichtigung von Einfachfehlern für die Analyse der Unabhängigkeit hinreichend. Mittels der Verifikation der Umschaltlogik (vgl. Kapitel 6), wurde zudem nachgewiesen, dass die Zustandsautomaten, sofern sie im Fehlerzustand sind, im gleichen Fahrzyklus nicht wieder aktiviert werden können. Intermittierende Fehler müssen daher nicht berücksichtigt werden.

Der On-the-Fly Model-Checking Algorithmus für die Analyse basiert auf einer begrenzten Tiefensuche mit expliziter Zustandsbeschreibung. Das Verfahren wurde aufgrund des geringeren Speicherbedarfs gewählt, da der Zustandsraum vorab nicht vollständig bestimmt werden muss (vgl. Kapitel

3.3.2.2). Für die Analyse sind dabei zwei Zustandsräume, der Such- und der Prüfraum, relevant. Der Prüfraum wird, wie beschrieben, durch die Zustände der Umschaltlogik definiert. Der Suchraum wird durch das Modell und dessen Variablen sowie die Freiheitsgrade der Kommunikation und die Fehlerfälle aufgespannt. Ein Zustand des Suchraums ist exakt einem Zustand im Prüfraum zugeordnet. Umgekehrt ist die Zuordnung nicht eindeutig, da verschiedene Systemzustände zu einem Zustand der Umschaltlogik führen können.

Das angewandte Suchverfahren, gezeigt in Algorithmus 7.1, wird im Folgenden inklusive der implementierten Optimierungen vorgestellt.

Im ersten Schritt erfolgt die Initialisierung des Modells. Die Softwarekomponenten, Partitionierungen und Verbindungen werden aus Konfigurationsdateien generiert. Die Initialisierung der Software-Komponenten beinhaltet auch die Erstellung der Nutzsignale und der Qualifier-Signale. Die Konfigurationsdateien können aus Datenbanken und Analysewerkzeugen exportiert werden. Die Automatisierung ist erforderlich, da eine manuelle Erstellung aufgrund der Datenmengen ineffizient und fehleranfällig ist. Zudem werden die zu untersuchenden Fehlerfälle definiert.

Im nächsten Schritt erfolgt die Analyse. Dabei wird eine begrenzte, rekursive Tiefensuche genutzt, die ab einer gewissen Suchbreite parallelisiert wird. Rekursion bezeichnet eine Prozedur, die sich wiederum selbst aufruft und wird im Algorithmus zur Generierung und zum Durchsuchen des Zustandsbaumes genutzt. Die Parallelisierung dient dabei der Laufzeitoptimierung (vgl. Kapitel 3.3.2.1). Bereits untersuchte Zustände und die Fehlertoleranzzeit dienen als Terminierungsbedingung der Suchtiefe. Die Prozedur ist in Algorithmus 7.1 vereinfacht dargestellt.

Beim Aufruf der Prozedur wird das Modell  $M$ , der aktuelle Systemzustand  $Z_{\text{akt}}$ , die Liste bereits untersuchter Zustände  $Z_{\text{bekannt}}$ , die Spezifikationen  $\psi_{\text{Sys}}$  und der aktuelle Pfad ( $\text{Pfad}_{\text{akt}}$ ) übergeben.

Zuerst erfolgt ein Vergleich des aktuellen Zustands ( $Z_{\text{akt}}$ ) mit den bereits

---

**Algorithmus 7.1** paralleler Tiefensuch Model-Checking Algorithmus zur Analyse der Fehlerpropagation

---

```
1: procedure TIEFENSUCHE ( $M, Z_{\text{akt}}, Z_{\text{bekannt}}, \psi_{\text{Sys}}, \text{Pfad}_{\text{akt}}$ )
2:   if  $Z_{\text{aktuell}} = Z_{\text{untersucht}}$  then
3:     return
4:   else if then add  $Z_{\text{aktuell}}$  to  $Z_{\text{bekannt}}$ 
5:     if Fehlertoleranzzeit erreicht or Modellzustand=stationär then
6:       if  $\psi_{\text{Sys}}$  verletzt( $\text{Pfad}_{\text{aktuell}}$ ) then
7:         save Gegenbeispiel( $\text{Pfad}_{\text{aktuell}}$ ) return
8:       end if
9:     end if
10:  end if
11:  if Parallelisierung and  $\text{mod}(\text{Tiefe}_{\text{akt}}/\text{Tiefe}_{\text{Sync}})=0$  then
12:    Synchronisation( $Z_{\text{bekannt}}$ )
13:  end if
14:  if  $\text{Anzahl}_{\text{Folgezustaende}} < \text{Anzahl}_{\text{Kerne}}$  or Parallelisierung then
15:    for all Folgezustaende of berechne Folgezustaende( $Z_{\text{akt}}$ ) do
16:      TIEFENSUCHE( $M, Z_{\text{akt}}, Z_{\text{bekannt}}, \text{Spezifikationen}, \text{Pfad}_{\text{akt}}$ )
17:    end for
18:  else if  $\text{!Parallelisierung}$  and  $\text{Anzahl}_{\text{Folgezustaende}} \geq \text{Anzahl}_{\text{Kerne}}$  then
19:    for all Folgezustaende of berechne Folgezustaende( $Z_{\text{akt}}$ ) do
20:      PROZESS (TIEFENSUCHE ( $M, Z_{\text{akt}}, Z_{\text{bekannt}}, \psi_{\text{Sys}}, \text{Pfad}_{\text{akt}}$ )
21:    end for
22:  end if
23: end procedure
```

---

untersuchten Zuständen ( $Z_{\text{bekannt}}$ , vgl. Zeile 2). Für bereits untersuchte Zuständen und alle Folgezustände ist die Analyse und daher eine Verifikation respektive die Generierung eines Gegenbeispiels bei Verletzung der Spezifikationen bereits erfolgt. Ist dies der Fall wird der rekursive Aufruf beendet. Die Zustände werden dabei zur Reduktion des Speicherbedarfs auf ein Minimum an Informationen limitiert, die unter Berücksichtigung der Vorgängerzustände eine eindeutige Beschreibung und die Bestimmung der Folgezustände ermöglichen. Berücksichtigt werden daher die Zustände der Umschaltlogik, die Zustände der Transitionen im Stapelspeicher und die Ausgangssignale und Fehlerzustände, abweichend vom Initialzustand

(vgl. Kapitel 3.3.2.1, [WKP09]). In Kombination mit den vorangegangenen Zuständen, dem Pfad  $\text{Pfad}_{\text{akt}}$ , wird der Zustand damit vollständig beschrieben. Modellzusammenhänge, wie die Architektur, Eingangssignale, oder Meta-Informationen, wie die Simulationszeit werden nicht abgelegt, da diese aus dem Modell rekonstruiert werden können. Beim Speichern der Zustände erfolgt darüber hinaus eine verlustlose Komprimierung um den Speicherbedarf nochmals zu reduzieren (vgl. Kapitel 3.3.2.1). Caching wird verwendet, um sicher zu stellen, dass ein Speicherüberlauf nicht auftreten kann (vgl. Kapitel 3.3.2.2). Dabei wird der Zustand, dessen Aufruf am weitesten zurück liegt (engl. least-recently used) ersetzt. Zustände bis zu einer gewissen Suchtiefe werden nicht ersetzt, da diese als Ausgangspunkt für viele Folgezustände hohes Potential für eine Effizienzsteigerung bieten, die Wahl der entsprechenden Tiefe erfolgt dabei empirisch. Der eigentliche Vergleich prüft, ob der aktuelle Zustand  $Z_{\text{akt}}$  einem bereits untersuchten Zustand  $Z_{\text{bekannt}}$  vollständig entspricht. Dies geschieht auf Basis der komprimierten Zustände, was eine Laufzeitverbesserung bewirkt. Auch ein Vergleich auf Basis von Untermengen ( $Z_{\text{bekannt}} \subseteq Z_{\text{akt}}$ ) ist unter der Prämisse, dass Signale individuell mit Sicherheitsmaßnahmen versehen sind, möglich und kann durch den Vergleich von Pfaden und Gegenbeispielen verifiziert werden. Die Anzahl der zu untersuchenden Zustände wird dadurch minimiert, jedoch erfordert der Abgleich von Teilmengen ein Dekomprimieren der Zustände, was zu erheblichen Laufzeitverlängerungen geführt hat.

Sofern der Zustand unbekannt ist (vgl. Zeile 2), wird der aktuelle Zustand auf dem Laufwerk abgelegt. Der Arbeitsspeicher wird für das Modell und die rekursive Prozedur reserviert. Eine Verletzung der Spezifikation wird geprüft, sofern die Fehlertoleranzzeit entsprechend der Sicherheitsanforderungen erreicht ist, oder das System einen stationären Zustand erreicht hat (vgl. Algorithmus Zeile 5). Ein stationärer Zustand liegt vor, wenn keine Transition im System mehr zu einem veränderten Systemzustand führen kann. Im Falle einer Verletzung der Spezifikationen im Zielzustand wird ein Gegenbeispiel auf dem Laufwerk abgelegt, das den kompletten Pfad, die Sequenz der besuchten Zustände, beinhaltet.

Die eigentliche Berechnung und der rekursive Aufruf erfolgt in den Zeilen 15 und 16 respektive in den Zeilen 19 und 20 und den Folgezeilen. Darin werden alle Folgezustände des aktuellen Zustands bestimmt (engl. On-The-Fly). Die minimalen und maximalen Übertragungszeiten der Kommunikation aller relevanten Signale und für die jeweiligen Konfigurationen werden dabei berücksichtigt und spannen den Zustandsraum auf. Nutzsignale und die zugehörigen Qualifier-Signale werden in Kombination variiert.  $n$  Signale auf  $m$  Bussen führen zu  $2^{n+m}$  Kombinationen. Die Variation der Latenzen erfolgt daher nur für die relevanten Signale, Signale mit ausführbaren Transitionen, die eine Abweichung zum Initialzustand darstellen und damit den Systemzustand verändern und über einen Bus versendet werden. Ein Signal, das im aktuellen Zustand dem Initialzustand entspricht, und sich auch nicht ändert, hat daher keine Auswirkungen auf den Suchraum. Das Modell wird jeweils bis zur nächsten Änderung des Systemzustands ( $Z_{\text{akt}}$ ) berechnet. Die Berechnung erfolgt in der kleinsten zu berücksichtigenden Zeiteinheit des Systems, den Rechenzyklen der Steuergeräte. Für alle Folgezustände wird die Prozedur rekursiv aufgerufen. Suchtiefen, die Anzahl der Terminierungen aufgrund bereits untersuchter Zustände und die Anzahl analysierter Zustände werden für eine Auswertung dokumentiert.

Eine weitere Optimierung stellt die Parallelisierung zur Reduktion der Rechenzeit dar (vgl. Kapitel 3.3.2.1). Die Berechnungen werden auf die verfügbaren Kerne verteilt, um die Ausnutzung der Rechenleistung zu erhöhen. Die Parallelisierung erfolgt dabei, sofern die Suchbreite der Anzahl der Kerne entspricht und die Berechnung nicht bereits parallel läuft (vgl. Algorithmus Zeile 20). Dies kann schon bei geringen Suchtiefen, sobald Transitionen mehrerer Signale relevant sind, der Fall sein. Die parallelen Prozesse laufen dabei unabhängig voneinander. Damit bereits untersuchte Zustände nicht zeitlich in weiteren Prozessen analysiert werden ist eine Synchronisation dieser notwendig. Zu diskreten Suchtiefen wurde daher eine Synchronisation implementiert, die die Menge der bekannten Zustände aktualisiert und wiederum im aktuellen Prozess verwendet (vgl. Algorithmus Zeile 12). Hierfür ist allerdings ein Serialisieren der Zugriffe notwendig.

## 7.3 Validierung und Qualifizierung

Die Validierung des Systems erfolgt anhand eines Minimalbeispiels, abgeleitet aus der Software-Architektur einer Fahrwerksfunktion. Im Rahmen der Validierung wird zum einen anhand von Testfällen das Modellverhalten durch den Abgleich mit dem Erwartungswert und zum anderen die Identifikation von Verletzungen der Spezifikation durch Fehlerinjektion adressiert. Die falsch-negative Identifikation von Verletzungen ist dabei zugleich der für die Qualifizierung des Werkzeugs gemäß ISO 262626 notwendigerweise zu betrachtende Fehlerfall.

Das genutzte Minimalbeispiel einer vereinfachten, funktionalen Wirkkette für die Querregelung deckt alle relevanten Elemente, Mechanismen und Zusammenhänge ab und ermöglicht dabei eine Nachvollziehbarkeit des Verhaltens. Abbildung 7.3 zeigt die Architektur des Systems anhand eines Blockdiagramms. Die Wirkkette umfasst dabei Sensoren für die Bestimmung der Gierrate und zur Umsetzung der Lenkanforderungen mittels dem Lenkungsaktuator sowie entsprechende Regelschleifen für den Nominal- und den Rückfallkanal.

Signalverbindungen sind in Abbildung 7.3 durch die Verbindung von Ports dargestellt, einzelne Signale sind angetragen. Redundante Signale wurden in den Sensoranteilen berücksichtigt. Zwischen dem Nominal und Rückfallkanal werden dabei insbesondere für die Diagnose, aber auch für die Umsetzung funktionaler Aspekte, Signale kommuniziert. In der Komponente *Signalkoordinator* erfolgt die Umschaltung der Signale auf Meldung der Umschaltlogik. Die Umschaltlogik mit den Zustandsautomaten ist nicht dargestellt, wurde aber entsprechend der Partitionierung in Abbildung 2.9 vollständig berücksichtigt und an die Ausgänge der Qualifier-Signale angebunden. Ebenso wurden die Software-Komponenten Steuergeräten zugeordnet, wodurch auch die Anbindung von Kommunikations- und Energiebordnetz erfolgt. In der Abbildung 7.3 wird sowohl die intra- als auch inter-CPU dargestellt.

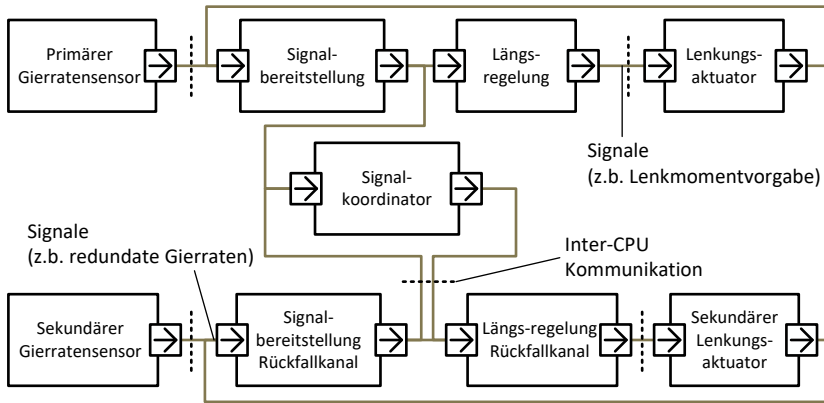


Abbildung 7.3: Blockdiagramm der vereinfachten Wirkkette als Validierungsbeispiel

Das Ziel der Analyse ist die Verifikation des Fail-Operational-Verhaltens im Software-Verbund unter Berücksichtigung der Fehlerpropagation und der Fehlertoleranzzeit. Die Validierung wird in zwei Schritten durchgeführt. Im ersten Schritt wird das Modellverhalten im zweiten Schritt die Identifikation von Verletzungen des Fail-Operational-Verhaltens validiert.

Für die Modellvalidierung erfolgt die Stimulation des Systems anhand der identifizierten Fehlerfälle und im Anschluss ein Abgleich des Verhaltens mit Erwartungswerten. Die Erwartungswerte stellen dabei wiederum die Zielbetriebe entsprechend Tabelle 6.1, erweitert um zusätzlich relevante Steuergeräte, die Software-Komponenten und Signale, dar. Der Zielbetrieb für Erweiterungen kann anhand der Schnittstellen, beispielsweise der Steuergeräte, auf denen die Partition der Software-Komponenten erfolgt, bestimmt werden. Signale werden der sendenden Software-Komponenten zugeordnet. Im Falle von Abweichungen sind zusätzliche Analysen oder Systemtests notwendig, da die Ursache nicht zwangsweise auf Modellierungsfehler, sondern gegebenenfalls auch auf das Systemdesign, zurück zu führen ist.

Auch daher empfiehlt sich die Verwendung eines Validierungsbeispiels mit überschaubarem Umfang. Die Validierung der Spezifikationen entfällt im Vergleich zu Kapitel 6.3, da als Prüfkriterium einzig die Zielbetriebe genutzt werden. Die Auswahl der Testfälle deckt dabei alle Fehlerfälle entsprechend der relevanten Klassen und Fehlermodi ab (vgl. Abbildung 7.1). Zusätzlich erfolgt der Abgleich mit dem Zielbetrieb auch in jedem Prüffall im Rahmen der Analyse. Die Modellvalidierung erfolgt unter der Prämisse, dass Abweichungen des Zielbetriebs erkannt und aufgezeigt werden, was im zweiten Validierungsschritt nachgewiesen wird.

Die Validierung der Analyse erfolgt durch Fehlerinjektionen, gezielte Manipulationen des Modellverhaltens, welche eine Abweichung von Erwartungswert und Modellverhalten bei einer entsprechenden Stimulation zu Folge haben. Die Fehlerinjektionen erfolgen in der Fehlermatrix, womit der Erwartungswert des Zielzustands verändert wird. Da im Rahmen der Modellvalidierung ein der Matrix entsprechendes Verhalten gezeigt wurde, muss eine Änderung der Erwartungswerte zu Verletzungen und Gegenbeispielen führen. Zusätzlich, separat davon, werden die Manipulationen einzeln, in jeder Klasse injiziert, so dass der Zielzustand nicht erreicht wird. Die Prüffälle decken dabei wiederum alle Fehlerfälle entsprechend der relevanten Klassen und Fehlermodi ab.

Entsprechend dem Model-Checking der Umschaltlogik in Kapitel 6.3 ist der für die Werkzeug-Qualifizierung nach ISO 26262 relevante, kritische Fehlerfall die falsch-negative Detektion von Prüffällen. Der Fehlerfall wurde bereits im Rahmen des zweiten Validierungsschritt adressiert, womit der notwendige Nachweis abgedeckt. Zusätzlich können, vergleichbar mit dem Vorgehen in Kapitel 6.3, weitere Fehlerinjektionen im Modell genutzt werden.



## 7.4 Ergebnisse und Diskussion des Ansatzes

Die Untersuchung der Anwendbarkeit der Methode erfolgt anhand der Analyse einer Fahrwerksfunktion aus der Industrie, einer Querführung im hochautomatisierten Fahrmodus. Darüber hinaus konnten mit der Analyse kritische Systemeigenschaften identifiziert und daraus Optimierungen für ein robustes Systemverhalten bezüglich der Fehlerpropagation abgeleitet werden. Die Diskussion der Methode unter Berücksichtigung der Einschränkungen und der Generalisierbarkeit erfolgt am Ende des Kapitels.

### 7.4.1 Anwendbarkeit der Methode

Das analysierte System stellt eine Querführung im hochautomatisierten Fahrbetrieb dar, von der auch das Validierungsbeispiel in [Abbildung 7.3](#) abgeleitet wurde. Auf Basis von Sensordaten wird eine Trajektorie bestimmt, welche unter Berücksichtigung der Fahrzeugeigenbewegung durch die Aktorik, wie Lenksysteme an Vorder- und Hinterachse, ein Bremssystem und einen Antrieb, eingeregelt wird. Das System ist dabei, entsprechend der Architektur in [Abbildung 2.9](#), zweikanalig ausgeführt und beinhaltet eine dezentrale Umschaltlogik (vgl. [Abbildung 2.10](#)). Es umfasst, inklusive der Zustandsautomaten, 35 Software-Komponenten auf 15 Steuergeräten. Sieben davon sind Sensoren, fünf Steuergeräte umfassen sowohl Sensorik als auch Aktorik-Anteile. Die Kommunikation erfolgt über fünf Bussysteme, von denen zwei deterministisch sind. Synchronisiert dazu ist jeweils ein Steuergerät auf welchem die eigentlichen Regelanteile auf Fahrzeugebene partitioniert sind. Kommuniziert werden jeweils über einhundert Nutz- und Qualifier-Signale, sowie die Signale zwischen den Zustandsautomaten. Das System umfasst alle relevanten Mechanismen und stellt einen entsprechend komplexen Systemumfang dar, wie er in Serienfahrzeugen zum Einsatz kommt.

Die Anwendbarkeit der Methode umfasst, analog zu Kapitel 6.4, die beiden Aspekte Modellierung beziehungsweise Implementierung und die Rechenzeit.

Die Erstellung und die Validierung der Klassen- und Objektstruktur stellt den größten Aufwand dar. Dies allerdings ist nur initial erforderlich. Die eigentliche Modellierung, die Erstellung der Objekte, erfolgt vorwiegend automatisiert. Hierfür müssen eine entsprechende Verfügbarkeit und Zugänglichkeit der Informationen gegeben sein. Im Anwendungsfall wurden Konfigurationsdateien aus Architekturdatenbanken abgeleitet, für welche die Schnittstelle ausgelegt ist. Sofern die Struktur der Dateien dem Datenschema entspricht, ist der Aufwand zur Generierung und Anpassung gering. Zusätzlich mussten einzelne abweichende Parameter der Software-Komponenten, wie Entprellzeiten oder Meldungen an die Zustandsautomaten, manuell angepasst werden, da ein entsprechender Export nicht verfügbar war. Die Validierung der formalen Spezifikationen ist nicht erforderlich, sofern dies im Rahmen der Verifikation der Umschaltlogik bereits erfolgt ist. Der Implementierungsaufwand, ist daher abgesehen von der initialen Erstellung der Klassen- und Objektstruktur, im Vergleich zu anderen Analysen, wie einer FMEA, gering.

Die Rechenzeit ist ein weiterer Einflussfaktor, der die Anwendbarkeit bestimmt. Abbildung 7.4 zeigt die Verteilung der Rechenzeiten bei exemplarischen Analysen für verschiedene Fehlerarten. Die genutzte Hardware ist identisch zu Anwendung in Kapitel 6.4 (6 Kerne, 4GHz, 16GB RAM, Windows 10, 64 bit).

Die Rechenzeit ist vertikal angetragen. Horizontal ist die Anzahl der Zustände zu sehen. Es wird zwischen geprüften (o) und bereits besuchten Zuständen (x), Zuständen, bei denen eine Terminierung der Suchtiefe erfolgt, unterschieden, die für einen Prüffall jeweils mit einer Linie verbunden dargestellt sind. Gezeigt sind die Analysen verschiedener Fehlerursachen: Steuergeräteausfälle, Ausfälle von Software-Komponenten, Kommunikations- und Energie-Bordnetzausfälle, Signalfehler und Signalausfälle. Die darge-

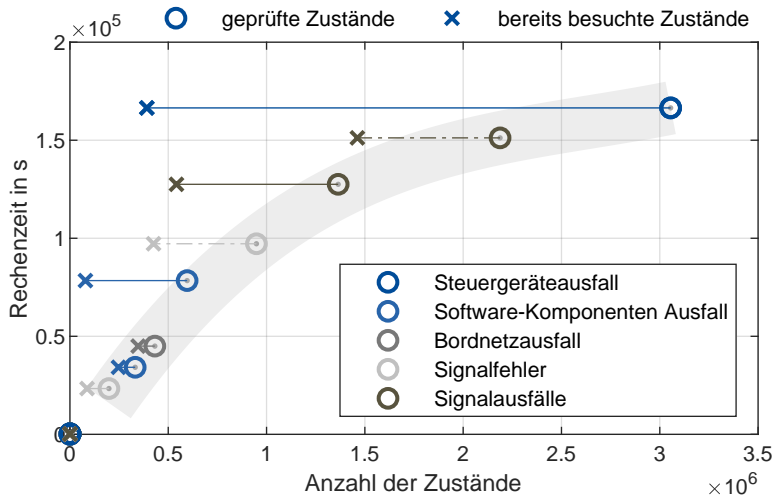


Abbildung 7.4: Rechenzeiten in Abhängigkeit der geprüften und bereits besuchten Zustände

stellten Fehlerfälle umfassen dabei Fehlermodi jeweils im Nominal- und im Rückfallkanal und wurden gewählt, um die Abhängigkeit zwischen der Rechenzeit und der Anzahl der Zustände aufzuzeigen. Die Prüfungen erfolgten, mit Ausnahme zweier Fälle, vollständig und zeigen Abstufungen der Rechenzeiten. Im Falle der unvollständigen Prüfungen, dargestellt durch unterbrochene Linien, lässt die Anzahl der besuchten Zustände nur eine begrenzte Aussage zu. Die Menge bereits bekannter Zustände war zu Beginn eines jeden Prüflaufes leer.

Das Diagramm zeigt, dass die Rechenzeit mit der Zahl der geprüften Zustände steigt. Der Anstieg ist nicht linear wie durch den Verlauf des grauen Bereichs im Diagramm zu sehen ist. Die Fehlerfälle auf dem Rückfallkanal stellen die Prüfungen mit geringerer Rechenzeit dar, da entsprechend die Anzahl der besuchten Zustände geringer ist. Neben der Zahl der geprüften Zustände hat auch die Suchtiefe Einfluss auf die Rechenzeit. Fehler im Rückfallkanal wirken sich nicht auf den Nominalkanal aus, die entsprechenden

Zustandssequenzen sind daher kürzer. Die längsten Rechenzeiten haben Fehler ergeben, welche im Nominalkanal und am Ende der Signalkette, an den Aktoren, diagnostiziert werden. Im Diagramm hat der Ausfall des primären Lenksystems zur längsten Rechenzeit geführt. Das liegt daran, dass sowohl Fehler als auch Sicherheitsmechanismen mehrere Nachrichten und Reaktion erfordern bis eine Umschaltung erfolgen kann und die Fehlerfälle damit entsprechend der Freiheitsgrade den größten Zustandsraum aufspannen. Die Tiefensuche erfordert gegenüber der Breitensuche geringere Rechenzeiten, weswegen die Zeit für die Berechnung eines Zustandes geringer als beim Ausfall zentraler Komponenten ist. Das zeigt sich durch den abflachenden Verlauf der Rechenzeiten, grau unterlegt, in [Abbildung 7.4](#). Die Zahl der bereits besuchten Zustände, Zustände für die die Terminierungsbedingung in [Zeile 2](#) erfüllt ist, korreliert im Allgemeinen nicht mit der Zahl der geprüften Zustände sondern ist abhängig vom jeweiligen Fehlerfall und dem resultierenden Zustandsbaum, lag aber in jedem Fall unter der Anzahl der besuchten Zustände. Fehler, die direkt zu einer Umschaltung führen, erfordern weniger Rechenzeit. Die kürzesten Rechenzeiten, mit unter 15 besuchten Zuständen, ergeben sich durch die Ausfälle der Energiebordnetze.

Insgesamt zeigen sich bei der unabhängigen Prüfung der Fehlerfälle Rechenzeiten von teilweise mehreren Tagen für einige Analysen. Auch wenn sicherlich noch Optimierungspotential vorhanden ist und die sich Rechenzeiten bei der Analyse mehrerer Fehlerfälle durch bereits bekannte Zustände reduzieren, stellt diese eine Limitierung dar.

#### 7.4.2 Optimierung der Systemrobustheit auf Basis der Analyse

Eine vollständige Verifikation des Systems ist, wie in [Kapitel 7.4.1](#) gezeigt, aufgrund der Rechenzeiten nur unter großem Zeit- und Rechenaufwand möglich. Auf Basis der Analyse des Anwendungsfalls konnten jedoch Maßnahmen und Mechanismen identifiziert werden um die Robustheit des Fail-Operational-Systems zu erhöhen. Hierfür wurden die Ursachen von Verletzungen des Fail-Operational-Verhaltens untersucht.

Potentiell kritisch sind Signale, die zwischen den Kanälen kommuniziert wer-

den, da diese unter Umständen zu einer Propagation von Fehlern und einer Deaktivierung in beiden Kanälen führen. Das Fail-Operational-Verhalten wird verletzt, wenn ein Fehler schneller propagiert als der Sicherheitsmechanismus dies verhindern kann. Die Möglichkeit der Fehlerpropagation ist dabei von der Architektur abhängig. Im einfachsten Fall werden keine Signale zwischen den Kanälen kommuniziert. Aber auch eine einseitige Kommunikation, wie im Anwendungsfall, indem nur Signale des Nominalkanals auf den Rückfallkanal gesendet werden, vereinfacht das Sicherheitskonzept. Sofern ein Kanal mehrere redundante Signale empfängt ist eine Signalumschaltung für den Fehlerfall notwendig. Bei einer wechselseitigen Kommunikation muss zusätzlich die Information über den Beginn des Rückfallbetriebs und die Unterbrechung der Kommunikation zwischen den Kanälen möglichst direkt kommuniziert werden.

Für die Analyse des kritischen Systemverhaltens wurden die maximalen Übertragungszeiten um den Faktor Zwei erhöht und die Sequenzen, die das Fail-Operational-Verhalten des Systems verletzen, analysiert. Dabei zeigt sich, dass Verletzungen auftreten, wenn die Fehlerpropagation mit geringer Latenz und die Sicherheitsreaktion, die Signal- und Kanalschaltung, mit hoher Latenz berücksichtigt wird. Dabei muss die Umschaltzeit der Signale geringer als die Diagnosezeit auf dem Rückfallkanal sein, damit dort keine Reaktion erfolgt. Daraus ergeben sich zwei mögliche Ansätze. Zum einen kann die Latenz der Sicherheitsreaktion gegenüber der Fehlerpropagation verringert werden. Zum anderen kann die Deaktivierung im Rückfallbetrieb verzögert oder ausgesetzt werden. Beide Ansätze wurden anhand exemplarischer Analysen untersucht.

Diese haben gezeigt, dass eine möglichst direkte Anbindung der Signalumschaltung sich positiv auf die Robustheit auswirkt. Die Signalumschaltung muss bei fehlerhaften Signalen auf die fehlerfreien Signale umschalten. Eine Umschaltung aufgrund der frühestmöglichen Diagnose, durch möglichst wenige Sequenzen, verringert die Zahl der Verletzungen und erhöht damit die Robustheit des Systems. Im Gegensatz dazu führt eine Umschaltung der Signale aufgrund der Zustandsautomaten zu Verzögerungen, da mehrere

Zustandsänderungen notwendig sind. Die absolute Diagnosezeit dagegen ist, sofern diese auf Nominal- und Rückfallkanal identisch ist, nicht relevant. Einen weiteren Ansatz stellt die Erhöhung der Robustheit des Rückfallbetriebs dar, durch welche Latenzen der Sicherheitsmechanismen kompensiert werden. Zu diesem Zweck kann die Entprell- und Diagnosezeit im Rückfallkanal erhöht werden. Auch der komplette Verzicht auf Diagnosen im Rückfallbetrieb ist eine Möglichkeit. Dafür allerdings muss der Wechsel auf den Rückfallbetrieb in den entsprechenden Komponenten rechtzeitig, unter der Diagnosezeit, beispielsweise über direkte Busverbindungen innerhalb der Brems- oder Lenksystemem, kommuniziert werden. Auch das Verhindern einer Deaktivierung im Rückfallbetrieb oder deren Verzögerung durch die Umschaltlogik stellt einen entsprechenden Ansatz dar, wobei dies unter Umständen zu einem fehlerhaften Betrieb bei Mehrfachfehlern führen kann.

Das Ziel der Systementwicklung ist ein fehlerfreies Verhalten. Die Kombination einer direkteren Umschaltung, also der Anbindung der Signalumschaltung an die Qualifier-Signale der relevanten Signale, und die Erhöhung der Diagnosezeiten auf dem Rückfallkanal haben auch unter Variation zu dem spezifizierten Fail-Operational-Verhalten aller analysierten Sequenzen geführt. Zur Sicherung des Fail-Operational-Verhaltens beziehungsweise der Optimierung der Systemrobustheit wird daher sowohl eine direkte Umschaltung als auch eine robuste Diagnose des Rückfallsystems empfohlen.

#### 7.4.3 Diskussion des Ansatzes

Im Folgenden wird der Model-Checking Ansatz für die Verifikation des Fail-Operation Software-Verbunds und die Analyse der Fehlerpropagation anhand der Erkenntnisse und Einschränkungen, sowie die Generalisierbarkeit des Ansatzes, diskutiert.

Der Model-Checking Ansatz bietet die Möglichkeit der systematischen Analyse der Fehlerpropagation in komplexen Systemen unter Berücksichtigung zeitlicher Aspekte, bei welchen eine manuelle Analyse impraktikabel

ist. Es wurde gezeigt, dass durch die Verifikation die Einhaltung der Fehler-toleranzzeit und des Fail-Operational-Verhaltens, unter Berücksichtigung von Fehlerpropagationen, nachweisbar ist. Insbesondere konnte durch die Ableitung von Design-Entscheidungen zum Verhindern von Fehlerpropagation sichergestellt werden. Das Vorgehen entspricht dabei, inklusive einer Qualifizierung des Software-Werkzeugs, den Anforderungen der ISO 26262. Bei einer umfangreichen Analyse ergeben sich allerdings Limitierungen durch die Rechenzeit. Wenngleich Optimierungspotential, zum Beispiel durch die Wahl eines effizienten Model-Checking Werkzeugs, besteht, stellen Rechenzeiten aufgrund von Größen des Zustandsraumes eine bekannte Einschränkung von Model-Checking Ansätzen dar (vgl. Kapitel 3.3). Für Konzeptentscheidungen sowie die Analyse einzelner Aspekte, bei denen die Zusammenhänge zu komplex für die Anwendung einer manuellen Methode sind oder eine ergänzende Analyse notwendig ist, ist die Analyse jedoch, wie die Validierung in Kapitel 7.3 und die Analyse der Systemrobustheit in Kapitel 7.4.2 gezeigt haben, geeignet und kann damit eine Ergänzung der Sicherheitsaktivitäten für die Untersuchung komplexer zeitlicher Abläufe darstellen.

Einschränkungen im Rahmen der Modellierung ergeben sich durch Prämissen und Vereinfachungen. Mit der Analyse wird der Fail-Operational Systemzustand verifiziert. Eine Vorbedingung stellen dabei Signalspezifikationen und Sicherheitskonzepten unterlagerter Funktionen und Komponenten dar. Ein fehlerhaftes Signal beispielsweise wird nur dann analysiert, wenn der Fehlermodus entsprechend spezifiziert ist. Dabei ist insbesondere eine Konsolidierung der Fehlerspezifikation notwendig, wenn Signale an mehreren Stellen verifiziert werden. Vereinfachungen im Rahmen der Modellierung betreffen, wie in Kapitel 6.4 diskutiert, auch das zeitliche Verhalten, die synchrone Berechnung und die Diskretisierung der Fehlermodi. Im Rahmen der Analyse werden unter Variation der Zeiten die Extremfälle abgedeckt, die, wie die Systemanalyse in Kapitel 7.4.2 gezeigt hat, die kritischen Fälle darstellen. Eine Absicherung auf der Zielhardware kann durch die Analyse nicht vollständig ersetzt werden [Int18].

Wie auch in Kapitel 6, wurde die Analyse anhand eines spezifischen Systems durchgeführt. Dennoch ist diese unter Berücksichtigung einiger Anpassungen generalisierbar. Das Vorgehen im Rahmen der Modellierung ist auf weitere Konzepte und Fahrzeugsysteme übertragbar, da die einzelnen Modellierungselemente, wie modulare Fehlerbäume und die Berücksichtigung der Architektur und der Kommunikation auf Basis allgemeiner Konzepte in der Automobilindustrie und des Autosar-Industriestandards [AUT06] definiert wurden. Die Identifikation der Zustände muss systemspezifisch erfolgen, wobei auch zu prüfen ist inwiefern eine Diskretisierung zulässig ist. Diese stellt eine notwendige Voraussetzung für die Bestimmung des Zustandsraumes und der Berechnung dessen dar. Der gezeigte Algorithmus kann auf weitere Analyse übertragen werden. Die Ergebnisse, welche in Kapitel 7.4.1 und insbesondere in Kapitel 7.4.2 vorgestellt wurden, zeigen generell gültige Zusammenhänge auf, müssen für detaillierte und spezifische Aussagen aber im Kontext des jeweiligen Systemkonzepts evaluiert werden.



# QUANTITATIVE ANALYSE VON FAIL-OPERATIONAL-SYSTEMEN

Die Argumentation der funktionalen Sicherheit (vgl. Kapitel 4.4) erfordert den Nachweis eines ausreichend geringen Restrisikos der Sicherheitszielverletzungen durch zufällige Hardwarefehler. In der Norm für funktionale Sicherheit, ISO 26262, wird hierfür die Auswertung der probabilistischen Metrik für zufällige Hardwarefehler (engl. PMHF) bestimmt, welche die durchschnittliche Ausfallwahrscheinlichkeit pro Stunde über die Lebenszeit darstellt [Int18]. Üblicherweise wird die Einheit Failures in Time (FIT) genutzt. Ein FIT ist äquivalent zu einem Ausfall in  $10^9$  h. Als mögliche Referenz für Zielwerte eines akzeptablen Risikos, abhängig vom ASIL, werden Grenzwerte (vgl. Tabelle 8.1), angegeben.

Darüber hinaus werden die Einzelfehlermetrik und die Metrik für latente Fehler bestimmt. Für vollständig redundante und unabhängige Systeme werden diese durch die Fehlertoleranz und eine Diagnoseabdeckung im Rückfallkanal von über 90 % erfüllt, weswegen auf diese nicht näher eingegangen wird.

Tabelle 8.1: Grenzwerte für Sicherheitszielverletzungen durch zufällige Hardwarefehler nach ISO 26262 [Int18]

ASIL	Grenzwert für das Restrisiko
D	$10^{-8} \text{ h}^{-1}$
C	$10^{-7} \text{ h}^{-1}$
B	$10^{-7} \text{ h}^{-1}$

Durch das hochautomatisierte Fahren und damit notwendige Fail-Operational-Verhalten ergeben sich gegenüber dem Betrieb in niedrigeren Automatisierungslevel Systemeigenschaften, die bei der quantitativen Analyse berücksichtigt werden müssen. In einem Fail-Operational-System, wie es in Abbildung 2.9 gezeigt wurde, ist eine individuelle Betrachtung der Komponenten und Betriebsphasen nicht hinreichend, da Fehler einer Komponente den Betrieb anderer bedingen beziehungsweise beeinflussen. Bei Fail-Silent-Systemen stellt das Abschalten den Übergang in den sicheren Zustand dar, dies erfolgt ohne Beeinflussung der anderen Komponenten. Der jeweilige Beitrag der Komponenten zur Ausfallwahrscheinlichkeit kann daher unabhängig bestimmt werden. Das Sicherheitsziel des Fail-Operational-Systems *Eine Kollision durch Verlassen der Trajektorie muss verhindert werden*, erfordert die Verfügbarkeit des Gesamtsystems, da für eine korrekte Bahnführung die Funktion der Wirkkette inklusive der Sensorik und Aktorik notwendig ist. Im Fehlerfall erfolgt eine Umschaltung der Wirkkette. Die Umschaltlogik ermöglicht dabei im Rückfallbetrieb nur bestimmte Systemkonfigurationen und erfordert eine Berücksichtigung von Vorbedingungen, da Fehler im Rückfallkanal erst nach einer Umschaltung zu einem Systemausfall führen. Darüber hinaus stellt die hochautomatisierte Fahrfunktion einen Fahrmodus dar, in dem verschiedene Betriebsmodi differenziert und berücksichtigt werden müssen. Latente Fehler können zudem außerhalb des autonomen Fahrmodus im passiven Kanal auftreten und erst bei einer entsprechenden Aktivierung wirksam werden.

Die Fehlerbaumanalyse ist der konventionelle und etablierte Ansatz für die quantitative Analyse zufälliger Hardwareausfälle in der Automobilindustrie [ESH15]. Allerdings wurden Fail-Operational-Systeme, und die damit einhergehend Gesamtsystembetrachtungen, Reihenfolgen der Ereignisse und resultierende Systemkonfigurationen sowie verschiedene Betriebsmodi bisher nicht vollständig berücksichtigt. Die Grundlagen der Fehlerbaumanalyse wurden in Kapitel 2.4.1 eingeführt. In der Literatur, insbesondere aus der Luft- und Raumfahrt, werden verschiedene Ansätze vorgestellt, um Reihenfolgen der Fehler und verschiedene Betriebsphasen, in Fehlerbäumen zu berücksichtigen (vgl. Kapitel 3.5). Diese werden in Kapitel 8.1 für ein Fail-Operational-System adaptiert. Da dynamische Vorgänge in Fehlerbäumen vereinfacht dargestellt werden (vgl. [Eri16]) erfolgt die Validierung mittels eines abstrahierten Markov-Modells in Kapitel 8.2. Markov-Modelle bilden dabei die Sequenzen der Zustände exakt ab. Die quantitative Analyse mit Markov-Modellen ist im Rahmen der Entwicklung allerdings aufgrund der Größe des Zustandsraums und dem Aufwand bei Anpassungen und Detaillierungen nicht sinnvoll [DBB93]. Durch eine Sensitivitätsanalyse des Fehlerbaums werden in Kapitel 8.3 relevante Parameter eines Fail-Operational-Systems für die Optimierung der Ausfallwahrscheinlichkeit identifiziert.

## 8.1 Modellierung mehrkanaliger Systeme mittels Fehlerbaum

Die Modellierung eines Fehlerbaums für das Fail-Operational-System (vgl. Abbildung 2.9) wird im Folgenden vorgestellt. Zu Beginn werden die relevanten Betriebsphasen und Diagnosemechanismen beschrieben. Auf Basis entsprechender Ansätze aus der Literatur (vgl. Kapitel 3.5) wird im Anschluss der Aufbau des statischen Fehlerbaums vorgestellt.

Die Betriebsmodi des hochautomatisierten Fahrens wurden bereits in Kapitel 2.3 eingeführt. Für die Ausfallwahrscheinlichkeit des Systems im hochautomatisierten Betrieb sind darüber hinaus Betriebsphasen vor dem eigentlichen Fahrmodus relevant, da sich latente Fehler in diesem auswirken können. Abbildung 8.1 zeigt eine Erweiterung der Betriebsphasen um die relevanten Betriebsmodi und Diagnosen. Die Phasendauern sind qualitativ dargestellt.

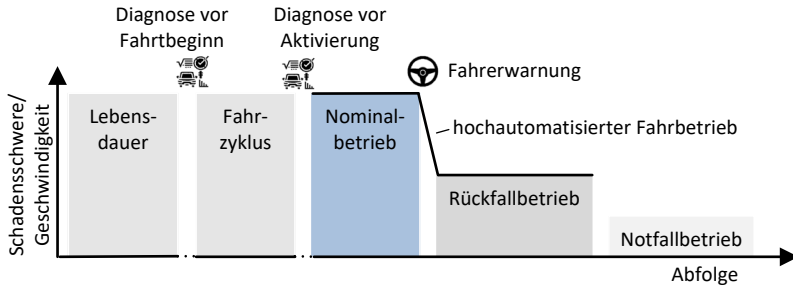


Abbildung 8.1: Relevante Betriebsphasen für die Ausfallwahrscheinlichkeit im hochautomatisierten Fahrmodus

Die längste Phase stellt die Fahrzeuglebensdauer dar. Relevante Fehler werden über die Fahrzeuglebensdauer mit einer gewissen Diagnoseabdeckung entdeckt. Ebenso erfolgt in den weiteren Betriebsphasen eine kontinuierliche Diagnose im Nominal- und im Rückfallkanal. Ein einzelner Fahrzyklus ist die Zeit zwischen zwei Klemmenwechseln, dem An- und Ausschalten des Motors. Vor jedem Fahrzyklus erfolgt ein Diagnosezyklus. Beispielsweise kann das Bremssystem mit Druck beaufschlagt werden, um die Funktionalität zu prüfen. Fehler, welche im Betrieb diagnostiziert werden können, werden auch durch den Diagnosezyklus vor Fahrtbeginn entdeckt. Das bedeutet umgekehrt ein vor Fahrtbeginn nicht entdeckter Fehler, wird auch im Fahrzyklus nicht erkannt. Vor der Aktivierung des hochautomatisierten Fahrmodus wird wiederum eine Systemdiagnose durchgeführt. Nominalbetrieb und Rückfallbetrieb wurden bereits in Kapitel 2.3 erklärt. Der Notfallbetrieb wird nicht betrachtet, da für diesen keine Sicherheitsanforderung gilt. Berücksichtigt werden außerdem nur Fehlermodi, welche für den hochautomatisierten Fahrbetrieb relevant sind.

Im nächsten Schritt erfolgt die Modellierung des Fehlerbaums unter Berücksichtigung der Abhängigkeiten. Abbildung 8.2 zeigt den Fehlerbaum mit den verschiedenen Betriebsmodi inklusive der logischen Verknüpfungen. Im Vergleich zur Ableitung der Fehlermodelle in Kapitel 4.3 ist für die quantitative Analyse die Wahrscheinlichkeit der Ereignisse relevant, weswegen jegliche Ereignisse mit einem Einfluss auf die Ausfallwahrscheinlichkeit explizit modelliert werden.

Das Sicherheitsziel kann in jedem der Betriebsmodi verletzt werden. Im Nominalkanal erfolgt dies durch nicht diagnostizierte, latente Fehler. Diese können im Nominalbetrieb, aber auch in vorherigen Phasen aufgetreten sein (vgl. Kapitel 4.3). Daher werden Fehlerereignisse für jede Phase bis einschließlich des Nominalbetriebs berücksichtigt (vgl. [VSD+02]). Entsprechend LaBand et al. [LA04] und Vario et al. [Vau01] wird für die Phasen in den jeweiligen Rückfallbetrieben der Abschluss der vorherigen Phasen als Bedingung berücksichtigt. Modelliert wird dies durch die Negierung des Ereignisses *Verletzung im Nominalbetrieb*. Ein Ausfall ist an dieser Stelle gleichbedeutend mit einer Verletzung von Sicherheitsanforderungen. Für einen Ausfall im Rückfallbetrieb muss zuerst ein Fehler, der zur Umschaltung in diesen führt, diagnostiziert werden und dann ein Ausfall einer aktiven Komponente erfolgen. Das ist im Fehlerbaum in Abbildung 8.2 durch die Verknüpfung der Diagnoseabdeckung und des Fehlers am Beispiel der primären Lenkung gezeigt. Für den sekundären Ausfall kann keine Reaktion erfolgen weswegen eine Diagnose nicht von Bedeutung ist. Da die Komponenten des Nominalkanals sowohl im Nominalbetrieb als auch im Rückfallbetrieb 1 aktiv sind, führen latente Fehler bereits zum Ausfall im Nominalbetrieb. Im Rückfallkanal können latente Fehler zum Eintritt in die Phase vorhanden sein, da die jeweiligen Komponenten zwar in einem Hot-Standby Betrieb laufen, aber nicht im Eingriff sind. Fehler wirken sich daher nicht im Nominalbetrieb aus. Abbildung 8.3 zeigt die Zusammenhänge anhand einer Detaillierung des Fehlerbaums für den Rückfallbetrieb 3 im Rückfallkanal.

Im Rückfallbetrieb 3 werden entsprechend die latenten Fehler der vorherigen Betriebsphasen und die Fehler während des Rückfallbetriebs berücksichtigt.

Für einen latenten Fehler, der vor dem Fahrzyklus aufgetreten ist, wird dessen Wahrscheinlichkeit, mit den Wahrscheinlichkeiten, dass dieser nicht diagnostiziert wurde, multipliziert (vgl. Abbildung 8.3 unterste Ebene). Dabei werden entsprechend der Betriebsphasen die relevanten Diagnosen berücksichtigt. Eine Erkennung eines latenten Fehlers im Nominalbetrieb kann nur durch die Diagnose im Nominalbetrieb erfolgen, ein latenter Fehler über die Lebensdauer dagegen durchläuft mehrere Diagnosen.

Für die Bestimmung der Ausfallwahrscheinlichkeiten sind neben der Modellierung der Betriebsphasen die entsprechenden Ausfallwahrscheinlichkeit bezogen auf die Phasendauer erforderlich. Die Wahrscheinlichkeit eines Ausfalls berechnet sich für exponentielle Fehlerverteilungen, die in der ISO 26262 angenommen werden, durch  $Q(t) = 1 - e^{-\lambda t}$  mit der Ausfallrate  $\lambda$ . Wobei durch  $t$  als das Intervall, in dem der Fehler auftreten kann definiert ist (vgl. Kapitel 2.1.2). Dieses Zeitintervall kann ein Diagnoseintervall, Reparaturintervall oder eine Betriebs- oder Phasendauer sein [BCS07]. Die obere Grenze des Intervalls stellt im Rahmen der Analyse die Fahrzeuglebensdauer dar. Zur Bestimmung der Ausfallwahrscheinlichkeit beziehungsweise der Metrik über die Lebensdauer entsprechend der ISO 26262 (PMHF), wird der jeweilige Maximalwert beziehungsweise Mittelwert, für ein durchschnittliche Ausfallwahrscheinlichkeit, bestimmt. Die Ausfallwahrscheinlichkeit nicht latenter Fehler im hochautomatisierten Fahrmodus werden über deren Phasendauer gemittelt, um gemäß ISO 26262 die mittlere Ausfallwahrscheinlichkeit pro Stunde zu bestimmen. Latente Fehler können über deren Phasendauer, nicht aber über die Betriebsphase, in der sich die Auswirkungen zeigen würden, gemittelt werden. Die Diagnoseabdeckungen werden durch absolute Wahrscheinlichkeiten berücksichtigt.

Trotz der Berücksichtigung der Betriebsphasen und des Umschaltkonzeptes müssen sind zur Berechnung Vereinfachungen im Fehlerbaum notwendig. Die Modellierung und Berechnung erfolgt mittels Isograph, Reliability Work-

bench<sup>1</sup>. Fehlerarten, wie latente Fehler, können im Modell entsprechend spezifiziert werden. Das Werkzeug erfordert für die Berechnung dynamischer Elemente jedoch eine Modellierung der Ereignisse direkt unter den übergeordneten Ereignissen. Dies ist notwendig, da die Berechnung über eine Transformation in Markov-Modelle erfolgt. Der Fehlerbaum würde dadurch unübersichtlich werden, weswegen eine Vereinfachung erforderlich ist. Dynamische Elemente werden daher nicht berücksichtigt. Ersatzweise werden Gatter mittels Und-Verknüpfungen modelliert. Auch die Exklusivität der Ereignisse, beispielsweise der Ausfälle in Rückfallbetrieb 1 und 2, stellt eine Vereinfachung im Fehlerbaum dar, welche nicht modelliert und nur mit Aufwand berücksichtigt werden kann. Im folgenden Kapitel wird die Modellierung validiert und dabei auch Abweichung durch die Näherung bewertet.

---

<sup>1</sup><https://www.isograph.com/software/reliability-workbench/>

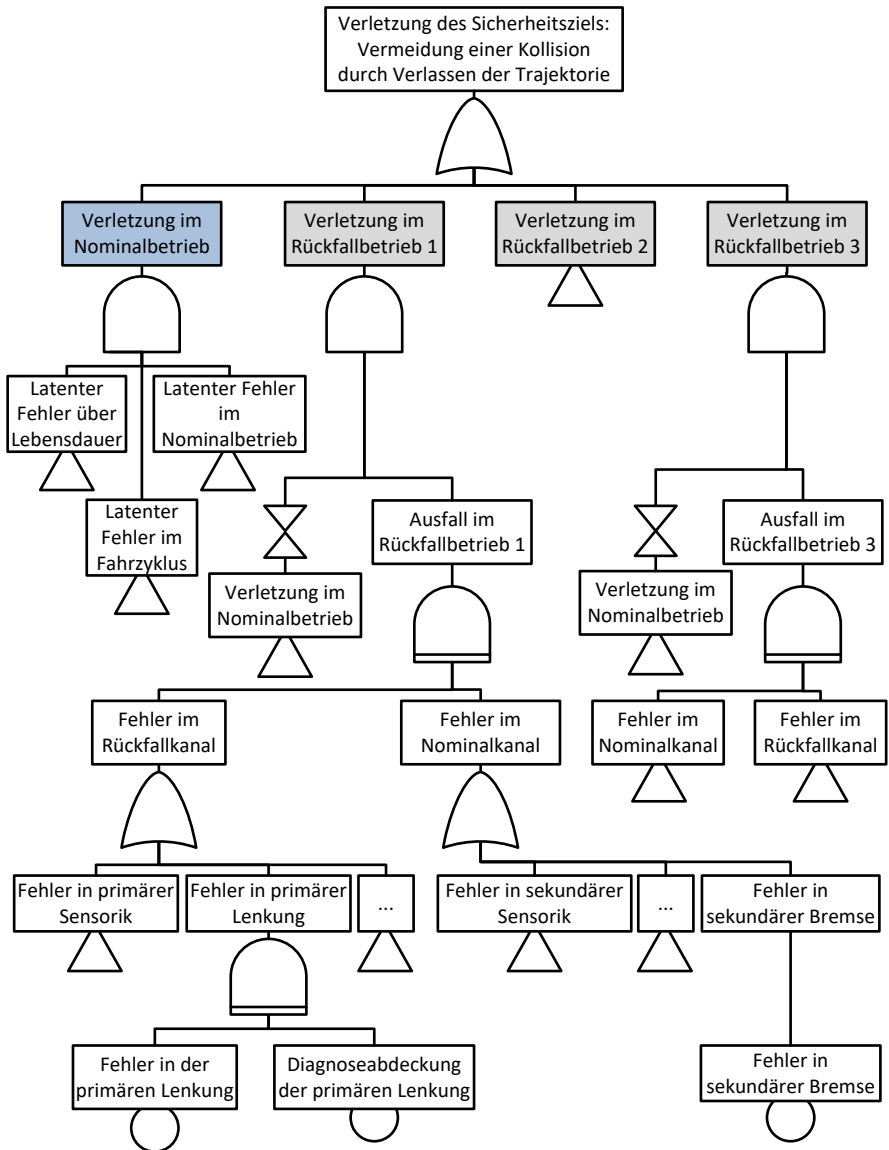


Abbildung 8.2: Aufbau eines Fehlerbaums unter Berücksichtigung der Betriebsphasen



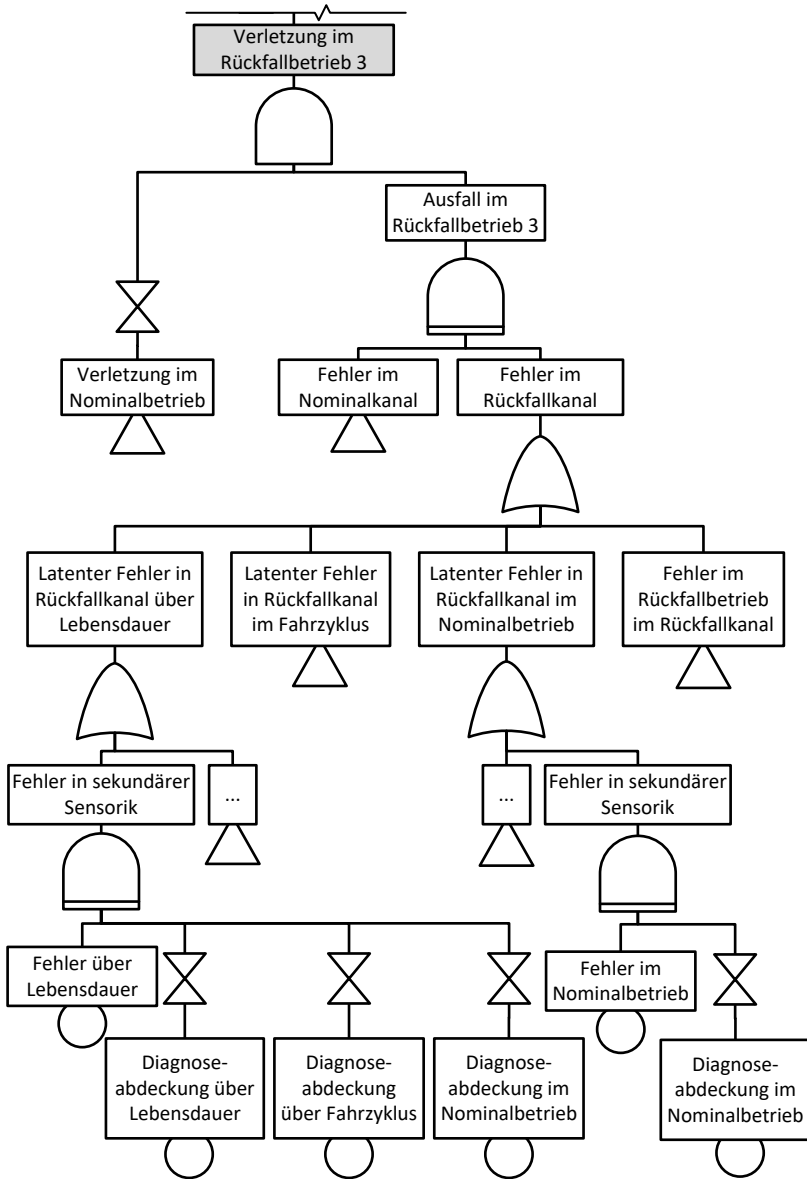


Abbildung 8.3: Detaillierung des Fehlerbaums für den Rückfallbetrieb auf dem Rückfallkanal

## 8.2 Validierung der Analyse mit Markov-Modellen

Um die Modellierung des Fehlerbaums zu validieren werden Markov-Modelle genutzt. Die Validität umfasst dabei sowohl die Modellierung als auch die Vereinfachungen durch eine statische Modellierung des Fehlerbaums. Markov-Modelle bieten die Möglichkeit Zustandsübergänge exakt zu modellieren (vgl. Kapitel 2.4.1 [Eri16]). Für einen Einsatz in der Entwicklung komplexer Systeme sind diese jedoch aufgrund der Komplexität und der Größe des Zustandsraums ungeeignet [DBB93]. Im Rahmen der Validierung konnte ein abstrahiertes Modell genutzt werden, in dem die Komponentenausfälle als Basisereignisse genutzt werden.

Die Modellierung der einzelnen Betriebsphasen und deren Übergängen mit einem Markov-Modell wird im Folgenden vorgestellt. Dabei wird zuerst das Modell und im Anschluss der Berücksichtigung der Phasen erläutert. Die Berechnung erfolgt mit realistischen Parametern, basierend auf Erfahrungswerten. Um die Generalisierbarkeit der Validität zu untersuchen wird eine Variation der Parameter für die Phasendauer, die Fehlerraten und die Diagnoseabdeckung durchgeführt. Markov-Modelle bieten darüber hinaus, die Möglichkeit die Häufigkeit der Fahrerübernahme im Modell zu bestimmen.

Zur Konstruktion eines Markov-Modelles werden die Auswirkungen der Fehlerereignisse als Zustände modelliert (vgl. Kapitel 2.4.1). Dabei werden vom Startereignis ausgehend durch Transitionen, Fehlereignisse, und resultierende Zustände das Markov-Modell konstruiert. Abbildung 8.4 zeigt den Aufbau des Markov Modells anhand eines vereinfachten Modells des Fail-Operational-Systems, entsprechend der zweiten Ebene unter dem Sicherheitsziel im Fehlerbaum. Da die Ausfallraten auf Komponentenebene spezifiziert sind, werden in der Darstellung in Abbildung 8.4 keine Wahrscheinlichkeiten gezeigt.

Auf der linken Seite ist der Initialzustand dargestellt. Der kritische Zustand ist die Verletzung des Sicherheitsziels, äquivalent zu einem Systemausfall auf der rechten Seite, in blau dargestellt. Darüber hinaus stellt die Fah-

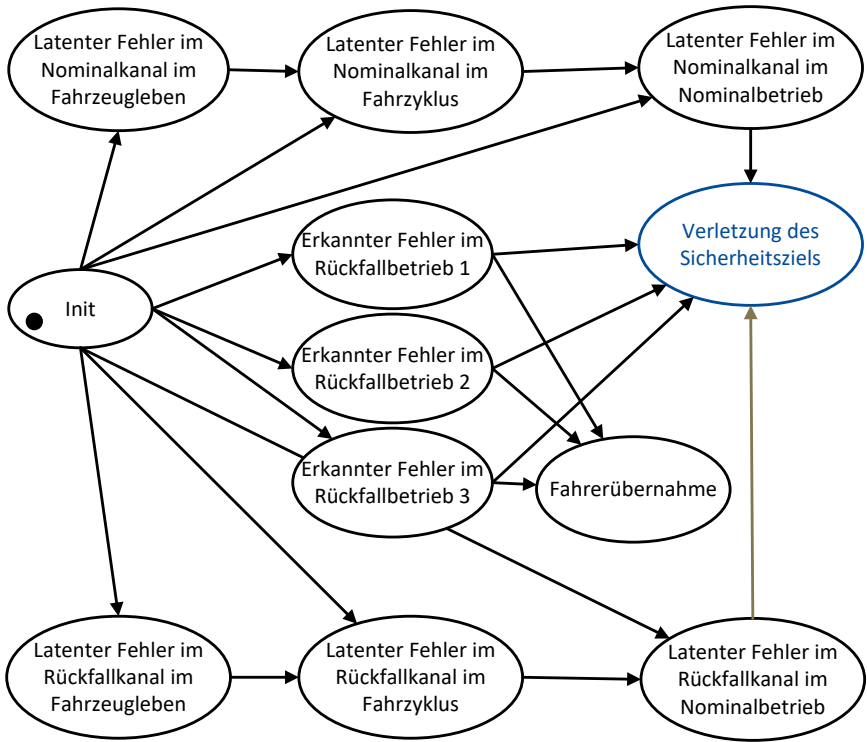


Abbildung 8.4: Schematischer Aufbau des Markov-Modells

rerübernahme einen Endzustand für die Betrachtung dar. Im Modell sind oben die Zustände mit latenten Fehlern der einzelnen Betriebsphasen im Nominalkanal dargestellt, die respektiven Zustände im Rückfallkanal sind unten zu sehen. Die Übergänge beinhalten dabei Ausfallraten und die Diagnoseabdeckung. Mittig sind diagnostizierte Fehler berücksichtigt, die zu einem Rückfallbetrieb führen. Das Modell ist dabei abstrahiert dargestellt. Zur eigentlichen Berechnung wird jedes Basisereignis, wie Komponentenausfälle, als Transition modelliert. Abbildung 8.5 zeigt schematisch eine Detaillierung des hervorgehobenen Übergangs von einem latenten Fehler auf der Rückfallebene zur Verletzung des Sicherheitsziels

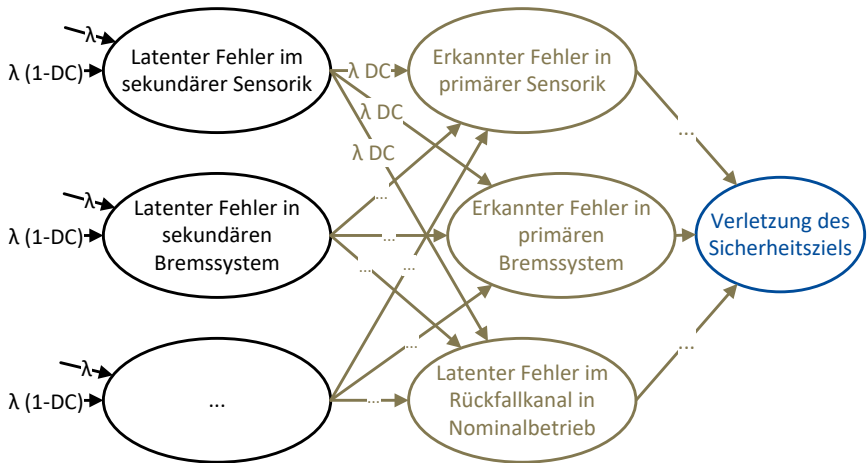


Abbildung 8.5: Detaillierung der Markov-Modells anhand der Basisereignisse

Mit dem Detaillierungsgrad der quantitativen Analyse steigt die Anzahl der Zustände und Transitionen. Für die Validierung des Fehlerbaum ist eine Detaillierung wie in Abbildung 8.5 gezeigt hinreichend, da die Fehler der Komponenten die Basisereignisse darstellen. Eine weitere Detaillierung im Markov-Modell ist aufgrund der Komplexität fehleranfällig und im Allgemeinen aufwändig, da die Struktur angepasst werden muss.

Im nächsten Schritt werden die einzelnen Betriebsphasen berücksichtigt. Ausgehend vom Modell in Abbildung 8.4 respektive 8.5 werden die relevanten Transitionen für jede Phase bestimmt. Abbildung 8.6 zeigt die Teilmodelle für die jeweiligen Betriebsphasen und die Phasenübergänge. Relevante Transitionen sind dabei dargestellt (vgl. Abbildung 8.4).

Dabei wird zwischen kontinuierlichen Betriebsphasen und den diskreten Diagnosezyklen unterschieden. Die Berechnung der kontinuierlichen Markov-Modelle erfolgt entsprechend über die jeweilige Phasendauer. Dabei werden die Wahrscheinlichkeiten des Eintretens der Zustände am Ende der jeweiligen Phase als Initialzustand für die folgende Phase genutzt.

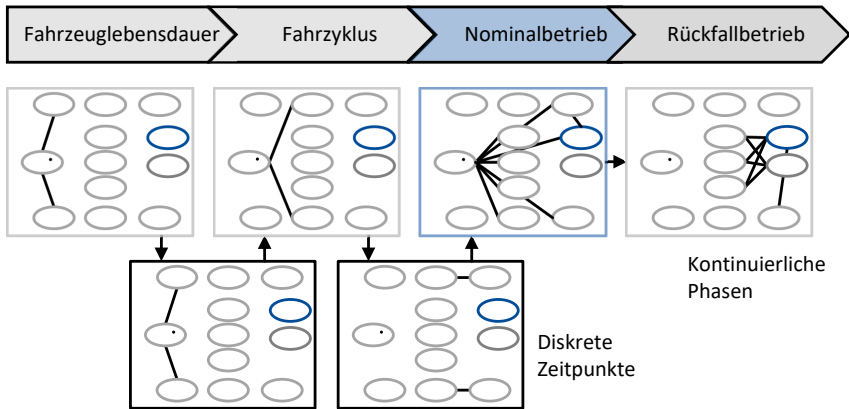


Abbildung 8.6: Berechnung des Markov-Modells über die Betriebsphasen

Nach der Betriebsphase über die Lebensdauer erfolgt daher entsprechend die Diagnose vor dem Fahrzyklus und im Anschluss der Betrieb im Fahrzyklus.

Die Berechnung der Ausfallwahrscheinlichkeit mit dem statischen Fehlerbaum wird anhand eines Abgleichs mit dem gezeigten Markov-Modell validiert. Die Validierung erfolgt dabei anhand realistischer Ausfall- und Diagnoseraten der einzelnen Komponenten sowie Phasendauern. Eine Generalisierung der Aussagen durch Variation der Parameter wird im Anschluss adressiert. Abbildung 8.7 zeigt die absolute und die relative Abweichung der Ansätze für die Berechnung der Wahrscheinlichkeit einzelner Ereignisse respektive Zustände. Berechnet wurde in den folgenden Analysen die mittlere Wahrscheinlichkeit einer Sicherheitszielverletzung des Fail-Operational-Systems entsprechend den Vorgaben der ISO 26262. Diese wird fortlaufend als Ausfallwahrscheinlichkeit bezeichnet.

Die Berechnung beider Fehlermodelle erfolgt dabei mit Industrie-üblichen Rohfehlerraten ( $10^{-6} \text{ h}^{-1}$ ) und Diagnoseabdeckungen (90 % für den herkömmlichen Betrieb, 99 % für den hochautomatisierten Betrieb) von sicherheitskritischen Komponenten. Für die Betriebsphasen wurde konservative Abschätzung getroffen. Die Fahrzeuglebensdauer wird mit 6000 Stunden,

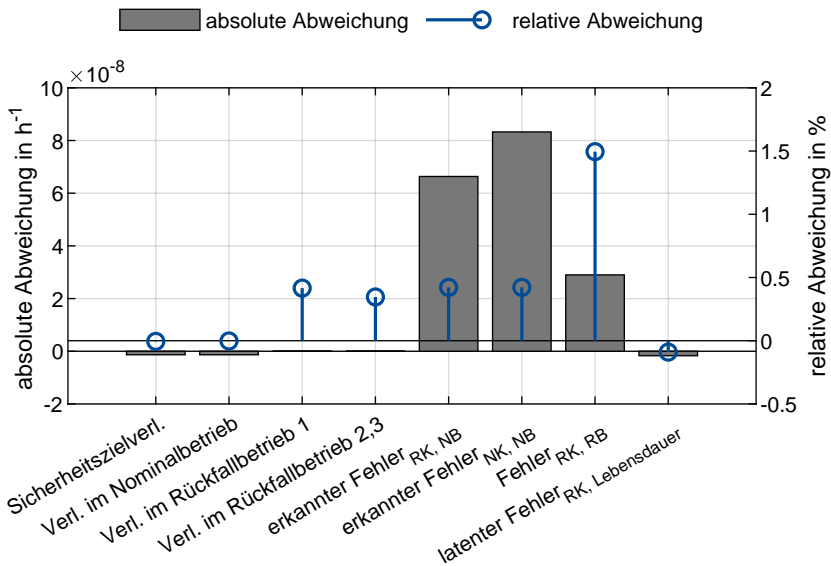


Abbildung 8.7: Abweichungen zwischen der anhand Fehlerbaum und Markov-Modell berechneten Ausfallwahrscheinlichkeit

der Fahrzyklus und der Nominalbetrieb mit jeweils 4 Stunden und der Rückfallbetrieb mit 0,01 Stunden angenommen (vgl. [Int18, Ver15b]). Dargestellt sind die Abweichungen der Ausfallwahrscheinlichkeit pro Stunde anhand der PMHF bezogen auf das Sicherheitsziel in den jeweiligen Betriebsmodi. Rückfallbetrieb 2 und 3 sind zusammengefasst, da beide den Betrieb auf dem Rückfallkanal zeigen. Zudem sind weitere repräsentative Ereignisse aufgetragen. Positive Abweiche zeigen eine höhere Ausfallwahrscheinlichkeit bei der Berechnung mit dem Fehlerbaum.

Für die Verletzung des Sicherheitsziels ergibt sich eine absolute Abweichung von  $-1,4 \cdot 10^{-9}$  und eine relative Abweichung bezogen auf die exakte Berechnung im Markov-Modell von 0,1 %. Die Berechnung im Markov-Modell ergibt demnach eine höhere Ausfallwahrscheinlichkeit. Die Abweichung

liegt eine Zehnerpotenz unter dem Grenzwert für zufällige Ausfälle gemäß ISO 26262. Vergleichbare Abweichungen ergeben sich auch für den Ausfall im Nominalbetrieb, welcher bezüglich der Wahrscheinlichkeit, der dominierende Betriebsmodus für eine Sicherheitszielverletzung ist (vgl. Abbildung 8.9b). Die Abweichungen in den Rückfallbetrieben liegen unter 0,4 %, bei absoluten Abweichungen von  $< 1 \cdot 10^{-10} \text{ h}^{-1}$ . Die weiteren Ereignisse zeigen relative Abweichungen von unter 1 %, beziehungsweise 1,6 % im Falle eines Fehlers auf dem Rückfallkanal (RK) im Rückfallbetrieb (RB). Die erkannten Fehler im Nominalbetrieb, die zu einem Rückfallbetrieb führen zeigen eine absolute Abweichung zwischen  $6 \cdot 10^{-8} \text{ h}^{-1}$  und  $8,3 \cdot 10^{-8} \text{ h}^{-1}$ . Die Auswirkung der Abweichung wird aber dadurch gemindert, dass die Fehler erst bei einem weiteren Ausfall im Rückfallbetrieb kritisch sind. Dies gilt auch für die weiteren Fehlerereignisse.

Die Abweichung der Berechnungen resultieren aus der Nichtberücksichtigung beziehungsweise Näherung der dynamischen Zusammenhänge im statischen Fehlerbaum und der Exklusivität von Ereignissen im Markov-Modell. Die bedingte Wahrscheinlichkeit im statischen Fehlerbaum wird durch ein logisches *Und* modelliert, die Reihenfolge wird dabei vernachlässigt. Damit stellt die Wahrscheinlichkeit das Produkt der Ausfallwahrscheinlichkeiten

$$Q_{FTA} = (1 - e^{-\lambda t})^2$$

zweier Elemente dar, sofern  $(1 - e^{-\lambda_1 t}) \approx (1 - e^{-\lambda_2 t})$  gilt. Im Markov-Modell wird die Abhängigkeit, als Sequenz unter Berücksichtigung der Exklusivität dargestellt. Daraus ergibt sich die Ausfallwahrscheinlichkeit

$$Q_{Markov} = \frac{\lambda_1(1 - e^{-\lambda_2 t}) - \lambda_2(e^{-\lambda_2 t} - e^{-(\lambda_1 + \lambda_2)t})}{\lambda_1 + \lambda_2},$$

wobei Element 1, mit der Ausfallrate  $\lambda_1$ , den primären Ausfall darstellt (vgl. [Eri16]). Die Abweichung wird dabei umso größer je länger die Zustandssequenz ist und wirkt sich daher auf Fehler früherer Betriebspha-

sen, beispielsweise auf Fehler während der Lebensdauer, stärker aus. Die Abweichungen durch die beschriebenen Unterschiede der Berechnungen kompensieren sich teilweise, was sich insbesondere in der Wahrscheinlichkeit des Ausfalls im Nominalbetrieb zeigt.

Darüber hinaus führen durch weitere Transitionen geminderte Ausgangswahrscheinlichkeiten im Markov-Modell zu Abweichungen. Im Fehlerbaum wird die Wahrscheinlichkeit des Basisereignisses, wie beschrieben, durch die Ausfallrate und das relevante Zeitintervall unabhängig von anderen Ereignissen bestimmt. In einer Markov-Kette mindern alle ausgehenden Transitionen eines Zustands die Ausgangswahrscheinlichkeit. Die Ereignisse sind daher nicht unabhängig. Die Vereinfachung führt zu höheren Ausfallwahrscheinlichkeiten im Fehlerbaum.

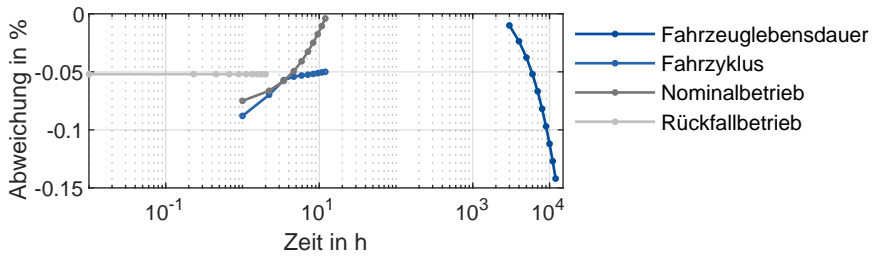
Die in [Abbildung 8.7](#) gezeigten Abweichungen der PMHF ergeben sich bei der beschriebenen Wahl der Parameter. Im Folgenden werden diese variiert, um eine allgemeine Aussage über die Validität der Fehlerbaummodellierung für ein Fail-Operational-Fahrzeugsystem zu treffen. [Abbildung 8.8](#) zeigt die Abweichung der Ausfallwahrscheinlichkeit zwischen Fehlerbaum und Markov-Modell bei Variation der Phasendauern, Fehlerraten und Diagnoseabdeckungen.

Die Variation der Parameter ist jeweils auf der horizontalen Achse, im Falle der Phasendauern und Fehlerraten logarithmisch, aufgetragen. Diese wurden, ausgehend von den definierten Parametern in einem plausiblen Bereich variiert. Vertikal ist die relative Abweichung in Prozent bezogen auf den Wert des Fehlerbaums aufgetragen.

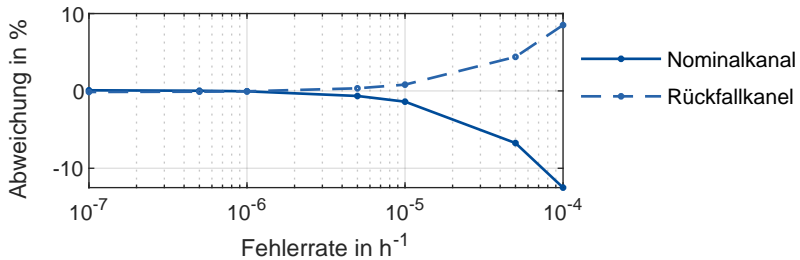
Bei der Variation der Phasendauer ([Abbildung 8.8a](#)) ergibt sich ein Abweichungen zwischen 0 und 0,1 %. Die Länge des Rückfallbetriebs zeigt keinen relevanten Einfluss auf die Abweichung, da die Wahrscheinlichkeit für den Rückfallbetrieb durch die Eintrittswahrscheinlichkeit eines Erstfehlers begrenzt ist. In den übrigen Phasen können latente Fehler respektive nicht diagnostizierte Fehler direkt zu einem Ausfall führen.

Die deutlichsten Abweichungen entstehen bei der Variation der Fehlerrate

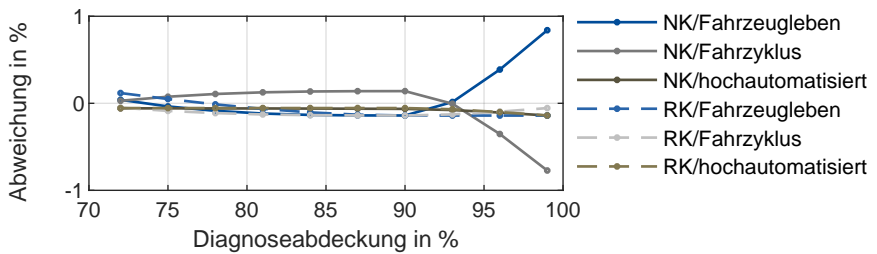




(a) Abweichungen bei der Variation der Phasendauern



(b) Abweichungen bei der Variation der Rohfehlerraten



(c) Abweichungen bei der Variation der Diagnoseabdeckungen

Abbildung 8.8: Abweichungen zwischen der Berechnung im Fehlerbaum und im Markov-Modell bei Parametervariation

in Abbildung 8.8b. Für Fehlerraten von  $10^{-4} \text{ h}^{-1}$  zeigt sich eine Abweichung von nahe 8 beziehungsweise 13 %. Die Ausfallrate im Nominalkanal führt dabei zu deutlicheren Auswirkungen, da solche Fehler unter Umständen direkt zu einer Verletzung des Sicherheitsziels führen. Die Variation der Rohfehlerraten umfasst dabei allerdings mehrere Zehnerpotenzen und ein deutlicher Anstieg ist erst bei Fehlerraten  $< 10^{-5} \text{ h}^{-1}$  zu sehen. Die Abweichung ist in den meisten Fällen akzeptabel, da die Abstufung der Risikobetrachtung in der ISO 26262 in Zehnerpotenzen erfolgt (vgl. Tabelle 8.1).

Die Variation der Diagnoseabdeckung in Abbildung 8.8c zeigt wiederum, insbesondere im Rückfallkanal, einen geringen Einfluss, auf die Abweichung der Berechnungsmethoden. Die Diagnoseabdeckung auf dem Nominalkanal für das Fahrzeugleben und den Fahrzyklus führt bei über 90 % Abdeckung zu Abweichungen nahe 1 %. Dies liegt an der Vernachlässigung der Sequenzen, die sich über mehrere Betriebsphasen entsprechend stärker auswirken.

Insgesamt zeigen sich nur geringe Abweichungen der Ausfallwahrscheinlichkeit zwischen den verschiedenen Berechnungsmethoden. Die Abweichungen bei der initialen Parameterauswahl liegt eine Zehnerpotenz unter dem Grenzwert der ISO 26262 für ASIL D Systeme (vgl. Tabelle 8.1). Die Fehlerbaummodellierung, die in Kapitel 8.1 dargestellt wurde, stellt daher eine akzeptable Näherung dar. Auch die Variationen der Parameter ändert dies nicht. Nur bei der Variation der Rohfehlerraten wurde eine Abweichung von 10 % festgestellt, was mitunter an den Variationsintervallen, die über übliche Werte sicherheitskritischer Systeme hinaus gehen, liegt. Statische Fehlerbäume liefern daher auch für Variationen in den relevanten Parameterbereichen eine hinreichende Näherung für die quantitative Bestimmung der Sicherheitszielverletzungen bei Fail-Operational-Systemen und können für den Sicherheitsnachweis gemäß ISO 26262 genutzt werden [Int18].

Das vorgestellte Fehlerbaummodell repräsentiert dabei eine spezifische Fail-Operational-Architektur, anhand derer die Gültigkeit der Modellierung gezeigt wurde. Die Exklusivität der Ereignisse wirkt sich vergleichbar auch bei anderen Konzepten aus. Bezüglich der bedingten Wahrscheinlichkeit

führt das untersuchte System aufgrund der Abhängigkeit kompletter Kanäle zu Abweichungen, die bei anderen Konzepten geringer sind. Eine Generalisierbarkeit der Aussage zur Anwendbarkeit von Fehlerbäumen für die Analyse von Fail-Operational-Systemen gemäß ISO 26262 ist daher gegeben.

Im Markov-Modell kann darüber hinaus eine zeitabhängige Wahrscheinlichkeit berücksichtigt werden. Durch die Berücksichtigung der Wahrscheinlichkeit einer Fahrerübernahme kann damit die Häufigkeit eines fehlerbedingten Eingreifens bestimmt werden. Im vorgestellten System ergeben sich  $10^{-5}$  Eingriffe pro Stunde bei einer Übernahmewahrscheinlichkeit von 80 % nach 10 s (vgl. [Gol16, ZWV+19]). Auf die Ausfallwahrscheinlichkeit hat die Fahrerübernahme nur einen geringen Einfluss (Abweichung von  $< 0,1\%$ ), da diese nur Ausfälle durch Zweitfehler im zeitlichen beschränkten Rückfallbetrieb verhindert.

### 8.3 Systemoptimierung auf Basis einer Sensitivitätsanalyse

Neben dem Sicherheitsnachweis kann durch das Fehlermodell (vgl. Abbildungen 8.2 und 8.3) eine Systemoptimierungen bezüglich der Ausfallwahrscheinlichkeit und damit einhergehend Ansätze zur Minimierung der Wahrscheinlichkeit einer Sicherheitszielverletzung identifiziert werden. Auf Basis des Fehlerbaums werden die Auswirkungen der Parameter Phasendauern, Fehlerraten und Diagnoseabdeckungen auf die Ausfallwahrscheinlichkeit des Systems untersucht. Durch die Auswirkungen auf die Wahrscheinlichkeit der Sicherheitszielverletzung werden die dominanten Parameter und Stellhebel für eine Systemoptimierung identifiziert. Für den Vergleich der Maßnahmen wird zusätzlich eine Metrik, die Birnbaum-Importanz, bestimmt.

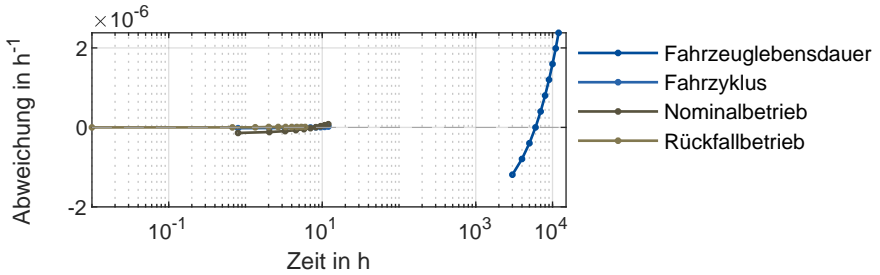
Die Parametervariation und deren Auswirkungen auf die berechnete Ausfallwahrscheinlichkeit im Fehlerbaum werden in Abbildung 8.9 gezeigt. Die Intervalle wurden entsprechend der Validierung in Kapitel 8.2 gewählt.

Dargestellt ist die Abweichung der Ausfallwahrscheinlichkeit pro Stunde (PMHF) bezogen auf die, mit den initialen Parametern (vgl. Kapitel 8.2) bestimmte, Ausfallwahrscheinlichkeit.

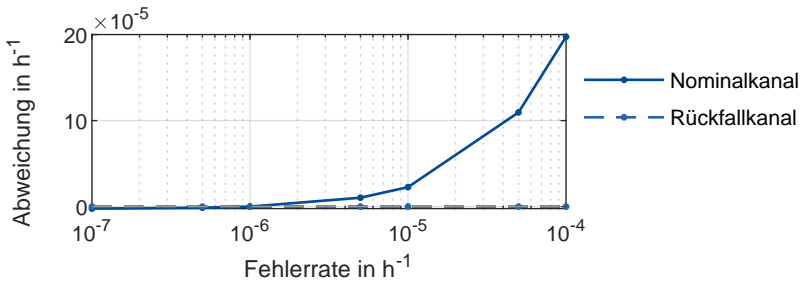
Abbildung 8.9a zeigt den Einfluss der Phasendauern auf die Ausfallwahrscheinlichkeit. Die Gradienten stellen dabei ein Maß für die Sensitivität der Ausfallwahrscheinlichkeit dar, negative Werte zeigen Optimierungspotential durch geringere Ausfallwahrscheinlichkeiten. Die Ausfallwahrscheinlichkeit weist bezüglich der Fahrzeuglebensdauer die größte Sensitivität bei Variationen der Phasendauern auf. Die Dauer des Rückfallbetriebes verändert die Ausfallwahrscheinlichkeit nur minimal, was auch im Hinblick auf höhere Automatisierungsstufen die Eignung der Systemarchitektur zeigt. Ursache sind dabei die Abhängigkeit vom Erstfehler und die Intervallgrenzen. Eine durchschnittlicher Fahrzyklus liegt unter einer Stunde [Ver15b], dieser wurde bis maximal 12 h untersucht. Nominal- und Rückfallbetrieb liegen analog innerhalb dieser Intervallgrenzen.

Bei der Variation der Fehlerraten in Abbildung 8.9b zeigt sich, dass die Fehlerrate des Nominalkanals schon zu einer deutlichen Abweichung der Ausfallwahrscheinlichkeit führt. Dies gilt für eine Reduzierung der Fehlerraten aber insbesondere für eine Verschlechterung bei hohen Fehlerraten. Dagegen weist die Fehlerrate des Rückfallbetriebs nur geringen Einfluss auf, da ein Fehler auf dem Rückfallkanal nur wirksam wird, wenn zuerst ein Fehler auf dem Nominalkanal detektiert wird. Nicht erkannte Fehler auf dem Nominalkanal dagegen führen direkt zu einem Ausfall, weswegen diese die Ausfallwahrscheinlichkeit dominieren.

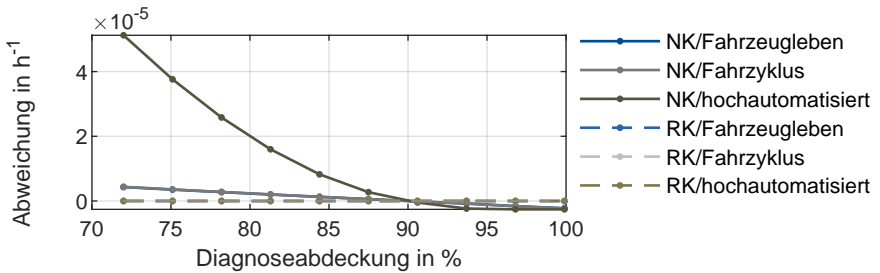
Die Sensitivitätsanalyse der verschiedenen Diagnoseabdeckungen ergibt einen, verglichen mit der Fehlerrate des Hauptkanals, geringeren Einfluss. Die Diagnosen auf dem Nominalkanal wirken sich deutlicher auf die Ausfallwahrscheinlichkeit aus als die auf dem Rückfallkanal. Die Sensitivität bezüglich der Diagnose im Nominalbetrieb ist am größten, da diese sich auch auf die latenten Fehler vorheriger Phasen auswirkt.



(a) Sensitivität der Ausfallwahrscheinlichkeit bei der Variation der Phasendauern



(b) Sensitivität der Ausfallwahrscheinlichkeit bei der Variation der Fehlerraten



(c) Sensitivität der Ausfallwahrscheinlichkeit bei der Variation der Diagnoseabdeckungen

Abbildung 8.9: Sensitivität der Ausfallwahrscheinlichkeit bei Parametervariationen

Eine Zusammenfassung und Priorisierung erfolgt anhand einer Importanz-Metrik. Die Birnbaum-Importanz, eine Metrik für quantitative Bewertung der Sensitivität, bietet die Möglichkeit die Parameter und damit die Ansätze für eine Systemoptimierung in Relation zu setzen. Diese ist definiert als die partielle Ableitung der Ausfallwahrscheinlichkeit über die Eintretenswahrscheinlichkeit eines Ereignisses  $E$ . [Eri16, VSD+02]

$$BI(E) = \frac{\partial Q_{Sys}}{\partial Q(E)} = Q(Sys, E_{max}) - Q(Sys, \neg E).$$

Nach Edler [ESH15] entspricht dies der Differenz der Ausfallwahrscheinlichkeit des Hauptereignisses bei maximaler und minimaler Eintretenswahrscheinlichkeit. Das Importanzmaß ist dabei auch für die Untersuchung einzelner Parameter geeignet. Es stellt ein Maß für die Optimierung der Ausfallwahrscheinlichkeit bei Optimierung der Parameter dar. Abbildung 8.10 zeigt die berechneten Importanzen für die Fehlerraten und Diagnoseabdeckungen.

Dabei wurden sowohl für die Ausfallraten als auch für die Diagnoseabdeckungen die Wahrscheinlichkeiten auf Eins beziehungsweise Null gesetzt und der Betrag der Differenz bestimmt, die quantitativen Werte dienen dabei der Priorisierung, lassen aber keine absolute Aussage über die Risikominderung zu. Zudem stellen diese aufgrund der Linearisierung eine Näherung dar. Die Bestimmung der Phasendauern ist nur innerhalb von Intervallen sinnvoll und stellt üblicherweise eine Konzeptentscheidung dar. Daher wird die Birnbaum-Importanz für die Phasendauern nicht bestimmt, sondern für Aussagen zur Sensitivität auf Abbildung 8.9a verwiesen. Die Optimierung des Hauptkanals stellt den wirksameren Ansatz zur Minimierung der Ausfallwahrscheinlichkeit als die Optimierung des Rückfallkanals dar, wie durch die Sortierung in Abbildung 8.10 gezeigt wird. Für beide Kanäle gilt, dass eine Reduzierung der Ausfallrate darüber hinaus mehr Wirkung zeigt als eine Verbesserung der Diagnoseabdeckung. Wie bereits in Abbildung 8.9c gezeigt, ist die Diagnose im Nominalbetrieb, die den Diagnosezyklus vor der

Aktivierung des Nominalbetriebs einschließt, wirkungsvoller bezüglich der Ausfallwahrscheinlichkeit als die Diagnose im Fahrzyklus und zur Lebenszeit.

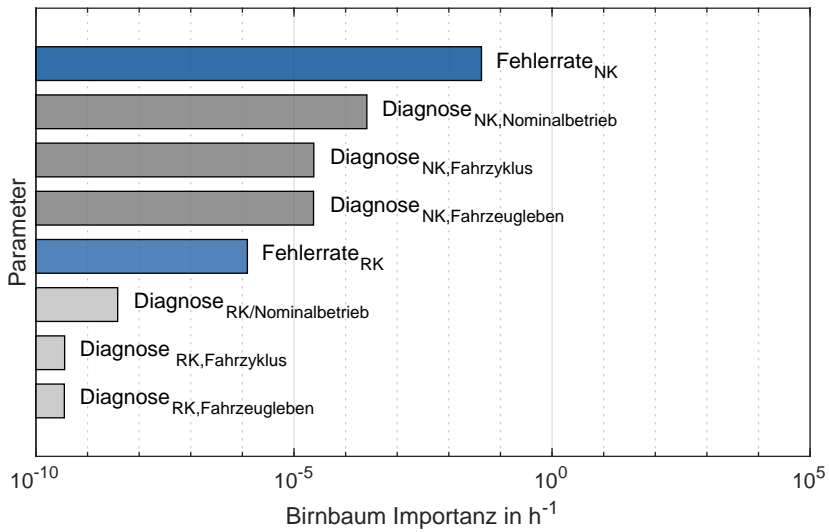


Abbildung 8.10: Birnbaum-Importanzen des der Parameter des Fail-Operational-Systems





# KAPITEL 9

## SCHLUSS

Der Schlussteil umfasst die Zusammenfassung der Ergebnisse und zeigt die relevanten Beiträge der Arbeit auf. Dazu werden die Inhalte der einzelnen Kapitel kurz beschrieben und die Ergebnisse im Kontext der Problemstellung erläutert. Darüber hinaus erfolgt in Kapitel 9.2 eine Diskussion der Grenzen sowie ein Ausblick auf zukünftige Forschungsrichtungen basierend auf den Erkenntnissen dieser Arbeit.

### 9.1 Zusammenfassung und Beiträge

Im Rahmen dieser Arbeit wurde eine hinreichende Argumentation der funktionalen Sicherheit für eine Fail-Operational-Fahrzeugführung im Kontext des hochautomatisierten Fahrens erbracht und der Bedarf an Sicherheitsanalysen auf Systemebene identifiziert. Ausgehend von der Sicherheitsargumentation wurden Vorgehen und Methoden für die identifizierten Analysen gemäß ISO 26262 abgeleitet.

Eine **hinreichende Sicherheitsargumentation für ein Fail-Operational System gemäß der ISO 26262**, welche in der Literatur bisher nur anhand einzelner Aspekte behandelt wurde, wurde in Kapitel 4 vorgestellt. Hierfür erfolgte eine Konsolidierung von Fail-Operational-Aspekten der Sicherheitsanforderungen gemäß ISO 26262 und Fehlermodellen zu Nachweiszielen in einer Goal-Structuring Notation. Die erstellte Sicherheitsargumentation definiert den hinreichenden Umfang der Sicherheitsanalysen gemäß ISO 26262. Dabei zeigt sich, dass auf Systemebene die Analyse abhängiger Fehler, die quantitative Analyse, die Verifikation des Sicherheits- und Umschaltkonzeptes und die Fehlertoleranzzeit berücksichtigt werden müssen. Die Argumentation zeigt außerdem auf, welche Beiträge Analysen auf Komponenten-Ebene liefern müssen.

Ein **Vorgehen zur Analyse gemeinsamer Ausfälle** wurde in Kapitel 5 präsentiert. Das definierte Vorgehen, ergänzt um ein entsprechendes Datenmodell, stellt eine Erweiterung existierender Ansätze für die Erfüllung der Anforderungen an die Analyse eines Fail-Operational-Systems im Industrieumfeld dar. Zur Identifikation der Anforderungen wurde eine Studie durchgeführt. Grenzen existierender Vorgehensweisen ergeben sich insbesondere durch den Unternehmenskontext, die verteilte Entwicklung und den Umfang der Analysen bei Fail-Operational-Systemen. Durch die Erweiterung des Vorgehens und Unterstützung anhand eines Datenmodells konnten die Anforderungen berücksichtigt werden, wie die Validierung anhand eines Anwendungsfalles gezeigt hat. Außerdem wurde die Anwendung quantitativer Methoden diskutiert. Dabei ergaben sich Limitierungen bezüglich der Übertragbarkeit auf die Automobilindustrie insbesondere durch kurze Entwicklungszyklen und den notwendigen Detaillierungsgrad der Fehleranalysen.

Die **Verifikation der Umschaltlogik** erfolgte in Kapitel 6 anhand eines Model-Checking Verfahrens, da manuelle Analysen dabei aufgrund der Anzahl der möglichen Zustände impraktikabel sind. Hierfür wurde ein Vorgehen präsentiert, das die Modellierung relevanter Aspekte, die formale

Spezifikation der Sicherheitsanforderungen, die Zuordnung dieser zu relevanten Fehlerfällen und die Validierung umfasst. Durch die Berücksichtigung der ISO 26262 bei der Definition des Umfangs und der Qualifikation des Werkzeugs wurde Konformität zur Industrienorm sichergestellt. Durch Beschränkung der Suchtiefe entsprechend der Fehlertoleranzzeit, ergab sich eine praktikable Rechenzeit wodurch der Nachweis der funktionalen Sicherheit vollständig erbracht und die Anwendbarkeit formaler Methoden für industrierelevante Problemstellungen gezeigt werden konnte.

Eine Erweiterung des Model-Checking für die Untersuchung der **Fehlerpropagation unter Berücksichtigung der Fehlertoleranzzeit** wurde in Kapitel 7 vorgestellt. Durch die Berücksichtigung der Fehlerlogik von Software-Komponenten und minimalen und maximalen Latenzen bei der Signalübertragung wurde das Fail-Operational-Verhalten im Verbund untersucht. Der Algorithmus basiert auf einer rekursiven Tiefensuche, bei der der Zustandsraum On-the-Fly konstruiert wird. Zudem wurden Optimierungen zur Reduktion des Speicherbedarfs und der Laufzeit implementiert. Durch eine Validierung, welche auch eine Qualifizierung des Werkzeugs beinhaltet, wurden die Vorgaben der ISO 26262 berücksichtigt. Eine Limitierung der Verifikationsmethode stellen lange Rechenzeiten dar. Durch die Ableitung von Designentscheidungen zur Erhöhung der Robustheit, konnte jedoch gezeigt werden, welche Anforderungen ein Fail-Operational-System erfüllen muss, um Fehlerpropagation zu verhindern und damit das fehlertolerante Verhalten sicherzustellen.

Ein **quantitativer Nachweis** eines ausreichend geringen Risikos einer Sicherheitszielverletzung für Fail-Operational-Systeme wird in Kapitel 8 präsentiert. Hierfür wurde die Modellierung von Fehlerbäumen unter Berücksichtigung der Betriebsmodi und des Umschaltkonzeptes gezeigt, wofür Modelle aus der Luft- und Raumfahrt als Grundlage dienten. Die Validität der Modellierung unter Berücksichtigung der Vereinfachungen im Fehlerbaum ließ sich durch Markov-Modelle nachweisen. Die Fehlerbaumanalyse kann daher auch als Nachweis für Fail-Operational-Systeme genutzt wer-

den, was in der Literatur bisher nicht hinreichend belegt wurde. Zusätzlich konnte durch eine Sensitivitätsanalyse der relevanten Parameter gezeigt werden, dass für die Ausfallwahrscheinlichkeit eines Fail-Operational-Systems insbesondere die Verfügbarkeit des Nominalkanals und dessen Diagnoseabdeckung relevant ist, eine Optimierung des Rückfallkanals dagegen einen geringere Reduktion der Ausfallwahrscheinlichkeit bewirkt.

## 9.2 Diskussion und Ausblick

Im Folgenden werden die Ergebnisse und Limitierungen, sowie alternative Lösungsansätze der Problemstellung diskutiert. Grenzen der einzelnen Analysen wurden bereits im Rahmen der einzelnen Kapitel ausgiebig diskutiert, der Fokus im Folgenden liegt auf allgemeinen Aspekten. Zudem ergeben sich auf Basis dieser Arbeit mögliche nächste Schritte, diese umfassen sowohl weiteren Forschungsbedarf als auch die Umsetzung der vorgestellten Ansätze und werden im Anschluss adressiert.

Die vorgestellte Sicherheitsargumentation adressiert die funktionale Sicherheit einer Fail-Operational-Fahrzeugführung, die Umsetzung einer Fahrvorgabe, nicht aber eines kompletten hochautomatisierten Fahrzeugsystems. Sicherheitsanalysen der Umfeldsensorik und ein damit einhergehendes Sicherheitskonzept werden in der vorliegenden Arbeit nicht berücksichtigt. Für den Nachweis der Sicherheit der Fahrzeugführung stellte dies keine Einschränkung dar, da die relevanten Fehlerfälle als Fehlerursachen der Algorithmen implizit berücksichtigt wurden. Ein Sicherheitsnachweis für eine hochautomatisierte Fahrfunktion erfordert jedoch darüber hinaus die Analyse der Umfeldsensorik sowie die entsprechende Ableitung von Sicherheitsmaßnahmen, was unter Umständen auch die Betrachtung stochastischer Algorithmen und Lernverfahren inkludiert.

Für die Herleitung der Sicherheitsargumentation und als Anwendungsfall der einzelnen Analysen wurde eine Fahrzeugarchitektur inklusive einem Umschaltkonzept genutzt. Das betrachtete System ist zweikanalig und

verfügt über eine dezentral verteilte Umschaltlogik zur Umsetzung des Fail-Operational-Verhaltens und einer möglichen Priorisierung von Aktoren. Zweikanalige Systeme mit dynamischer Redundanz werden dabei in der Literatur für die Anwendung in der Automobilindustrie empfohlen und stellen den Stand der Technik dar. Bezüglich der Umschaltlogik gibt es neben der Umschaltung zwischen diskreten Kanälen auf Systemebene auch komponentenweise Umschaltkonzepte. Das präsentierte Konzept kombiniert dabei beide Ansätze, weswegen mehrere Betriebsmodi definiert wurden. Die identifizierten Analysen sind für die funktionale Sicherheit jeglicher Fail-Operational-Konzept relevant. Die einzelnen vorgestellten Analysemethoden sind übertragbar, allerdings variiert der Aufwand und die Komplexität der Analysen mit dem Konzept. Im vorgestellten System stellt die Analyse der Umschaltlogik einen Schwerpunkt dar, diese ist im Falle einer Komponentenweisen Umschaltung unter Umständen weniger komplex. Dies gilt auch für die Analyse gemeinsamer Ausfälle, da keine kompletten Kanäle analysiert werden müssen. Dagegen ist der Nachweis des Fail-Operational-Verhaltens im Verbund inklusive der Analyse der Fehlerpropagation und auch der Absicherung auf der Zielhardware aufgrund möglicher Systemkonfigurationen aufwändiger. Dabei spielt auch das Diagnosekonzept, das unter Umständen eine Kommunikation der Signale erfordert eine Rolle. Je nach Konzept können dabei auch manuelle Analysen oder Simulationen einen möglichen Ansatz darstellen, welche für die behandelte Umschaltlogik impraktikabel sind.

Darüber hinaus unterliegen die einzelnen vorgestellten Methoden Limitierungen, welche im Rahmen der jeweiligen Kapitel diskutiert wurden. Dies betrifft unter anderem die Validität der Studie, weswegen potentielle Einschränkungen im Aufbau und der Durchführung berücksichtigt wurden, aber auch Vereinfachungen und den Aufwand im Rahmen der Modellierung für die formale Verifikation sowie die Rechenzeiten und systemspezifische Lösungen, die Einschränkungen bezüglich der Generalisierung darstellen.

Auf Basis dieser Arbeit ergeben sich weitere mögliche Schritte. Im Folgenden wird ein Ausblick auf weiteren Forschungsbedarf und die Umsetzungen gegeben.

Die Integration der Absicherung und Validierung für einen kompletten Safety-Case nach ISO 2626 erfordert eine Erweiterung der Sicherheitsaktivitäten. Für die Absicherung auf Fahrzeugebene ist hierfür auch die Integration der Umfeldsensorik und damit einhergehend eine szenarienbasierte Betrachtung notwendig. Auch stellt die Integration von Analysen der Datenverarbeitung durch stochastische und lernende Algorithmen nach ISO 26262 weiteren Bedarf dar, wobei auf Forschungsbeiträgen aus diesen Gebieten aufgebaut werden kann (vgl. [WRM+19]).

Die Arbeit hat die theoretische Basis für die Analysen geliefert und mögliche Umsetzungen aufgezeigt. Eine vollständige Integration der Analysen in die Serienentwicklung erfordert darüber hinaus weitere Schritte.

Bezüglich der Analyse gemeinsamer Ausfälle ist eine Einführung in verschiedenen relevanten Fachbereichen inklusive der Initialisierung einer entsprechenden Datenbank notwendig. Synchronisationen der Fehlerbeschreibungen und das Berücksichtigen von Rückmeldungen sind im Rahmen einer Etablierung erforderlich.

Auch die Integration der formalen Methoden in die entsprechende Entwicklungsumgebung und eine kontinuierliche Entwicklung stellt weiteren Forschungsbedarf dar. Dies erfordert neben der Anbindung an genutzte Werkzeuge, die Optimierung der Rechenzeiten sowie eine Qualifizierungsmaßnahmen von Personal und Software-Werkzeugen.

# LITERATURVERZEICHNIS

- [ADI+17] A. Albore, S. Dal Zilio, G. Infantes, C. Seguin, P. Virelizier. „A Model-Checking Approach to Analyse Temporal Failure Propagation with AltaRica“. In: *Model-based safety and assessment*. Hrsg. von M. Bozzano, Y. Papadopoulos. Bd. 10437. Lecture notes in computer science Programming and software engineering. Cham: Springer, 2017, S. 147–162. DOI: 10.1007/978-3-319-64119-5\_10 (Zitiert auf S. 76).
- [AHDR18] A. Aniculaesei, F. Howar, P. Denecke, A. Rausch. „Automated generation of requirements-based test cases for an adaptive cruise control system“. In: *2018 IEEE Workshop on Validation, Analysis and Evolution of Software Tests (VST)*. IEEE, 2018, S. 11–15. DOI: 10.1109/VST.2018.8327150 (Zitiert auf S. 71, 153).
- [AKHA20] S. Alcaide Portet, L. Kosmidis, C. Hernandez, J. Abella. „Software-Only Triple Diverse Redundancy on GPUs for Autonomous Driving Platforms“. In: *2020 50th Annual IEEE-IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S)*. IEEE, 2020, S. 82–88. DOI: 10.1109/DSN-S50200.2020.00045 (Zitiert auf S. 57).
- [AKM18] S. Altenburg, H.-P. Kienzler, A. auf der Maur. *Einführung von Automatisierungsfunktionen in der Pkw-Flotte: Auswirkungen auf Bestand und Sicherheit*. Hrsg. von Allgemeiner Deutschen Automobilclub e.V. 2018. URL: [https://www.adac.de/-/media/pdf/motorwelt/prognos\\_automatisierungsfunktionen.pdf](https://www.adac.de/-/media/pdf/motorwelt/prognos_automatisierungsfunktionen.pdf) (Zitiert auf S. 11).
- [AUT06] AUTOSAR Development Partnership. *Specification of RTE Software: Version 1.0.1*. 2006 (Zitiert auf S. 172, 190).

- [AVR19a] A. Aniculaesei, A. Vorwald, A. Rausch. „Automated Generation of Requirements-Based Test Cases for an Automotive Function using the SCADE Toolchain“. In: *ADAPTIVE 2019-The Eleventh International Conference on Adaptive and Self-Adaptive Systems and Applications*. Hrsg. von N. Abchiche-Mimouni. Wilmington, DE, USA: IARIA, 2019 (Zitiert auf S. 70, 71, 153).
- [AVR19b] A. Aniculaesei, A. Vorwald, A. Rausch. „Using the SCADE Toolchain to Generate Requirements-Based Test Cases for an Adaptive Cruise Control System“. In: *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. IEEE, 2019, S. 503–513. DOI: 10.1109/MODELS-C.2019.00079 (Zitiert auf S. 70, 71, 153).
- [AVZR21] A. Aniculaesei, A. Vorwald, M. Zhang, A. Rausch. „Architecture-based Hybrid Approach to Verify Safety-critical Automotive System Functions by Combining Data-driven and Formal Methods“. In: *2021 IEEE 18th International Conference on Software Architecture Companion (ICSA-C)*. IEEE, 2021, S. 1–10. DOI: 10.1109/ICSA-C52384.2021.00036 (Zitiert auf S. 69, 70).
- [BB19] J. M. Borky, T. H. Bradley. *Effective Model-Based Systems Engineering*. Cham: Springer International Publishing, 2019. DOI: 10.1007/978-3-319-95669-5 (Zitiert auf S. 45, 46).
- [BBC03] J. Barnat, L. Brim, J. Chaloupka. „Parallel breadth-first search LTL model-checking“. In: *Proceedings of the 18th IEEE International Conference on Automated Software Engineering (ASE’03)* (2003), S. 106–115. DOI: 10.1109/ASE.2003.1240299 (Zitiert auf S. 73).
- [BBC17] B. Bittner, M. Bozzano, A. Cimatti. „Timed Failure Propagation Analysis for Spacecraft Engineering: The ESA Solar Orbiter Case Study“. In: *Model-based safety and assessment*. Hrsg. von M. Bozzano, Y. Papadopoulos. Bd. 10437. Lecture notes in computer science Programming and software engineering. Cham: Springer, 2017, S. 255–271. DOI: 10.1007/978-3-319-64119-5\_17 (Zitiert auf S. 47).
- [BBG+05] S. Beyer, P. Bohm, M. Gerke, M. Hillebrand, T. I. der Rieden, S. Knapp, D. Leinenbach, W. J. Paul. „Towards the formal verification of lower system layers in automotive systems“. In: *2005 International Con-*



- ference on Computer Design*. IEEE Comput. Soc, 2005, S. 317–324. doi: 10.1109/ICCD.2005.110 (Zitiert auf S. 70).
- [BCC+09] A. Biere, A. Cimatti, E. M. Clarke, O. Strichman, Y. Zhu. „Bounded Model Checking“. In: *Handbook of satisfiability*. Hrsg. von A. Biere. Bd. 185. Frontiers in artificial intelligence and applications. Amsterdam und Washington, DC: IOS Press, 2009, S. 457–481 (Zitiert auf S. 50, 72, 73).
- [BCF+14] K. Beckers, I. Côté, T. Frese, D. Hatebur, M. Heisel. „Systematic Derivation of Functional Safety Requirements for Automotive Systems“. In: *Computer Safety, Reliability, and Security*. Hrsg. von A. Bondavalli, F. Di Giandomenico. Bd. 8666. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2014, S. 65–80. doi: 10.1007/978-3-319-10506-2\_5 (Zitiert auf S. 59).
- [BCL+11] M. Bozzano, A. Cimatti, O. Lisagor, C. Mattarei, S. Mover, M. Roveri, S. Tonetta. „Symbolic Model Checking and Safety Assessment of Altarica models“. In: *Electronic Communications of the EASST, Volume 46: Automated Verification of Critical Systems 2011*. Hrsg. von European Association of Software Science and Technology. European Association of Software Science and Technology, 2011. doi: 10.14279/tuj.eceasst.46.697 (Zitiert auf S. 76).
- [BCS07] H. Boudali, P. Crouzen, M. Stoelinga. „Dynamic Fault Tree Analysis Using Input/Output Interactive Markov Chains“. In: *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)*. 2007, S. 708–717. doi: 10.1109/DSN.2007.37 (Zitiert auf S. 79, 196).
- [BDL04] G. Behrmann, A. David, K. G. Larsen. „A Tutorial on Uppaal“. In: *Formal Methods for the Design of Real-Time Systems*. Hrsg. von D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, M. Bernardo, F. Corradini. Bd. 3185. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, S. 200–236. doi: 10.1007/978-3-540-30080-9\_7 (Zitiert auf S. 138).

- [BDP+17] J. Botham, G. Dhadyalla, A. Powell, P. Miller, O. Haas, D. McGeoch, A. Chakrapani Rao, C. O'Halloran, J. Kiec, A. Farooq, S. Poushpas, N. Tudor. „PICASSOS – Practical Applications of Automated Formal Methods to Safety Related Automotive Systems“. In: *SAE Technical Paper Series*. SAE Technical Paper Series. SAE International, 2017. doi: 10.4271/2017-01-0063 (Zitiert auf S. 70).
- [BEK17] G. Bahig, A. El-Kadi. „Formal Verification of Automotive Design in Compliance With ISO 26262 Design Verification Guidelines“. In: *IEEE Access (Volume 5)* (2017), S. 4505–4516. doi: 10.1109/ACCESS.2017.2683508 (Zitiert auf S. 70).
- [BFG+08] M. Broy, M. Feilkas, J. Grünbauer, A. Gruler, A. Harhurin, Judith Hartmann, B. Penzenstadler, B. Schaetz, D. Wild. *Umfassendes Architekturmodell fuer das Engineering eingebetteter Software-intensiver Systeme*. München, 2008 (Zitiert auf S. 78).
- [BH08] H. Boucheneb, R. Hadjidj. „Model Checking of Time Petri Nets“. In: *Petri Net*. Hrsg. von V. Kordic. London: IntechOpen, 2008. doi: 10.5772/5318 (Zitiert auf S. 46).
- [BK08] C. Baier, J.-P. Katoen. *Principles of model checking*. Cambridge, MA und London, England: The MIT Press, 2008 (Zitiert auf S. 48–50, 74, 75, 144).
- [BL04] B. Bertsche, G. Lechner. *Zuverlässigkeit im Fahrzeug- und Maschinenbau: Ermittlung von Bauteil- und System-Zuverlässigkeiten*. 3., überarbeitete und erweiterte Auflage. VDI-Buch. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg New York, 2004. doi: 10.1007/3-540-34996-0 (Zitiert auf S. 29).
- [BLP03] G. Behrmann, K. G. Larsen, R. Pelánek. „To Store or Not to Store“. In: *Computer Aided Verification*. Hrsg. von G. Goos, J. Hartmanis, J. van Leeuwen, W. A. Hunt, F. Somenzi. Bd. 2725. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, S. 433–445. doi: 10.1007/978-3-540-45069-6\_40 (Zitiert auf S. 74).

- [BPSW09] P. Braun, J. Philipps, B. Schätz, S. Wagner. „Model-Based Safety-Cases for Software-Intensive Systems“. In: *Electronic Notes in Theoretical Computer Science* 238.4 (2009), S. 71–77. DOI: 10.1016/j.entcs.2009.09.007 (Zitiert auf S. 60).
- [BRH+13] J. Birch, R. Rivett, I. Habli, B. Bradshaw, J. Botham, D. Higham, P. Jesty, H. Monkhouse, R. Palin. „Safety Cases and Their Role in ISO 26262 Functional Safety Assessment“. In: *Computer Safety, Reliability, and Security*. Hrsg. von F. Bitsch, J. Guiochet, M. Kaâniche. Bd. 8153. Lecture Notes in Computer Science / Programming and Software Engineering. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, S. 154–165. DOI: 10.1007/978-3-642-40793-2\_15 (Zitiert auf S. 59).
- [BS15] K. Becker, B. Schätz. „Deployment Calculation and Analysis for a Fault-Tolerant System Platform“. In: *11th Dagstuhl-Workshop on Model-Based Development of Embedded Systems (MBEES)* (2015) (Zitiert auf S. 57, 58).
- [BSK16] G. Biggs, T. Sakamoto, T. Kotoku. „A profile and tool for modelling safety information with design information in SysML“. In: *Software & Systems Modeling* 15.1 (2016), S. 147–178. DOI: 10.1007/s10270-014-0400-x (Zitiert auf S. 46).
- [Bal15] N. Balakrishnan. „An Overview of System Safety Assessment: Dependability in Medicine and Neurology“. In: *Dependability in Medicine and Neurology*. Hrsg. von N. Balakrishnan. Bd. 66. Cham: Springer International Publishing, 2015, S. 33–81. DOI: 10.1007/978-3-319-14968-4\_2 (Zitiert auf S. 61–63, 65, 66, 122).
- [Bel17] J.R. Belland. „Modeling common cause failures in diverse components with fault tree applications“. In: *Annual Reliability and Maintainability Symposium 2017 proceedings*. Piscataway, NJ: IEEE, 2017, S. 1–6. DOI: 10.1109/RAM.2017.7889659 (Zitiert auf S. 133).
- [Bie09] A. Biere, Hrsg. *Handbook of satisfiability*. Bd. v. 185. Frontiers in artificial intelligence and applications. Amsterdam und Washington, DC: IOS Press, 2009 (Zitiert auf S. 50).

- [Bry86] Bryant. „Graph-Based Algorithms for Boolean Function Manipulation“. In: *IEEE Transactions on Computers* C-35.8 (1986), S. 677–691. doi: 10.1109/TC.1986.1676819 (Zitiert auf S. 50).
- [CCG+02] A. Cimatti, E. M. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, A. Tacchella. „NuSMV 2: An OpenSource Tool for Symbolic Model Checking“. In: *Proceedings of the 14th International Conference on Computer Aided Verification. CAV '02*. Berlin, Heidelberg: Springer-Verlag, 2002, S. 359–364 (Zitiert auf S. 140).
- [CCJ+10] R. Cavada, A. Cimatti, C. Jochim, G. Keighren, E. Olivetti, M. Pistore, M. Roveri, A. Tchaltsev. *NuSMV 2.6 User Manual*. Hrsg. von FBK-irst. 2010 (Zitiert auf S. 51, 140, 141).
- [CGP+02] A. Cimatti, E. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, A. Tacchella. „Integrating BDD-Based and SAT-Based Symbolic Model Checking“. In: *Frontiers of Combining Systems*. Hrsg. von G. Goos, J. Hartmanis, J. van Leeuwen, A. Armando. Bd. 2309. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, S. 49–56. doi: 10.1007/3-540-45988-X\_5 (Zitiert auf S. 73).
- [CHVB18] E. M. Clarke, T. A. Henzinger, H. Veith, R. Bloem. *Handbook of Model Checking*. Cham: Springer International Publishing, 2018. doi: 10.1007/978-3-319-10575-8 (Zitiert auf S. 48–52, 74, 75).
- [CJWZ17] L. Chen, J. Jiao, Q. Wei, T. Zhao. „An improved formal failure analysis approach for safety-critical system based on MBSA“. In: *Engineering Failure Analysis* 82 (2017), S. 713–725. doi: 10.1016/j.engfailanal.2017.06.034 (Zitiert auf S. 77).
- [CVWY92] C. Courcoubetis, M. Vardi, P. Wolper, M. Yannakakis. „Memory-efficient algorithms for the verification of temporal properties“. In: *Formal Methods in System Design* 1.2-3 (1992), S. 275–288. doi: 10.1007/BF00121128 (Zitiert auf S. 73).
- [DBB93] J. B. Dugan, S. J. Bavuso, M. A. Boyd. „Fault trees and Markov models for reliability analysis of fault-tolerant digital systems“. In: *Reliability Engineering & System Safety* 39.3 (1993), S. 291–307. doi: 10.1016/0951-8320(93)90005-J (Zitiert auf S. 44, 79, 193, 200).

- [DTT+18] C. Dropmann, E. Thaden, M. Trapp, D. Uecker, R. Amarnath, L. A. da Silva, P. Munk, M. Schweizer, M. Jung, R. Adler. „A Model-Based Safety Analysis of Dependencies Across Abstraction Layers“. In: *International Conference on Computer Safety, Reliability, and Security*. Bd. 11088. 2018, S. 73–87. DOI: 10.1007/978-3-319-99130-6\_6 (Zitiert auf S. 66).
- [Deu01] Deutsches Institut für Normung e.V. *Funktionale Sicherheit sicherheitsbezogener elektrischer/elektronischer/programmierbarer elektronischer Systeme*. 2001-07-03 (Zitiert auf S. 22, 26, 61, 64, 131, 133).
- [Dor18] C. Dorrer. „Automated driving at BMW – Solutions for today and tomorrow“. In: *18. Internationales Stuttgarter Symposium*. Hrsg. von M. Bargende, H.-C. Reuss, J. Wiedemann. Wiesbaden: Springer Fachmedien Wiesbaden, 2018, S. 859–871 (Zitiert auf S. 11).
- [EGZ20] C. Ebner, K. Gorelik, A. Zimmermann. „Model-Based Dependability Analysis of Fail-Operational Electric Drivetrains“. In: *2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE, 2020, S. 256–263. DOI: 10.1109/AIM43001.2020.9158882 (Zitiert auf S. 56).
- [ELL04] S. Edelkamp, S. Leue, A. Lluch-Lafuente. „Directed explicit-state model checking in the validation of communication protocols“. In: *International Journal on Software Tools for Technology Transfer* 5.2-3 (2004), S. 247–267. DOI: 10.1007/s10009-002-0104-3 (Zitiert auf S. 73).
- [ES03] N. Eén, N. Sörensson. „An Extensible SAT-solver“. In: *International conference on theory and applications of satisfiability testing*. Bd. 2919. 2003, S. 502–518. DOI: 10.1007/978-3-540-24605-3\_37 (Zitiert auf S. 50).
- [ESH15] F. Edler, M. Soden, R. Hankammer. *Fehlerbaumanalyse in Theorie und Praxis: Grundlagen und Anwendung der Methode*. Berlin und Heidelberg: Springer Vieweg, 2015. DOI: 10.1007/978-3-662-48166-0 (Zitiert auf S. 40–43, 78, 193, 212).
- [Ehr01] J. Ehret. „Validation of Safety-Critical Distributed Real-Time Systems“. Dissertation. München: Technische Universität München, 20.01.2003 (Zitiert auf S. 77).

- [Eri16] C. A. Ericson II. *Hazard analysis techniques for system safety*. 2. ed. Hoboken, NJ: Wiley, 2016 (Zitiert auf S. 40–47, 61–63, 65, 66, 78, 123, 131, 193, 200, 205, 212).
- [FH94] P. Fenelon, B. Hebborn. „Applying HAZOP to software engineering models“. In: *Risk management and critical protective systems: proceedings of SARSS*. 1994, S. 11–116 (Zitiert auf S. 47).
- [FMS12] S. Friedenthal, A. Moore, R. Steiner. *A practical guide to SysML: The systems modeling language*. 2. Auflage. Waltham, MA: Morgan Kaufmann, 2012. DOI: 10.1016/B978-0-12-385206-9.10001-8 (Zitiert auf S. 46).
- [Fri21] M. J. Friese. „Modeling and Analysis of Automotive Cyber-physical Systems“. Dissertation. Kiel: Universität Kiel, 2021. DOI: 10.21941/kcss/2021/2 (Zitiert auf S. 36, 172, 173).
- [GZ17] W. Gong, X. Zhou. „A survey of SAT solver“. In: *Applied mathematics and computer science*. Hrsg. von K. Ntalianis. Conference collection. Melville, New York: AIP Publishing, 2017, S. 020059. DOI: 10.1063/1.4981999 (Zitiert auf S. 50).
- [Gol11] J. Goll. *Methoden und Architekturen der Softwaretechnik*. 1. Aufl. Studium. Wiesbaden: Vieweg+Teubner Verlag, 2011. DOI: 10.1007/978-3-8348-8164-9 (Zitiert auf S. 125, 167).
- [Gol16] C. Gold. „Modeling of Take-Over Performance in Highly Automated Vehicle Guidance“. Dissertation. München: Technische Universität München, 2016. URL: <https://mediatum.ub.tum.de/doc/1296132/document.pdf> (Zitiert auf S. 209).
- [HHHH16] S. Hauge, A. Hoem, P. Hokstad, Habrekke, Solfrid, Lundteigen, Mary Ann. „Common cause failures in safety-instrumented systems: Using field experience from the petroleum industry: Beta-factored and equipment specific checklists based on operational experience“. In: *Reliability Engineering & System Safety* 151 (2016), S. 34–45. DOI: 10.1016/j.ress.2015.09.018 (Zitiert auf S. 133).

- [HHK+10] R. Hayama, M. Higashi, S. Kawahara, S. Nakano, H. Kumamoto. „Fault-tolerant automobile steering based on diversity of steer-by-wire, braking and acceleration“. In: *Reliability Engineering & System Safety* 95.1 (2010), S. 10–17. DOI: 10.1016/j.ress.2009.07.003 (Zitiert auf S. 56, 57).
- [HIRK10] I. Habli, I. Ibarra, R. S. Rivett, T. Kelly. „Model-Based Assurance for Justifying Automotive Functional Safety“. In: *2010 SAE World Congress*. 2010. DOI: 10.4271/2010-01-0209 (Zitiert auf S. 59).
- [HKAS14] A. B. Hocking, J. Knight, M. A. Aiello, S. Shiraishi. „Arguing Software Compliance with ISO 26262“. In: *2014 IEEE International Symposium on Software Reliability Engineering Workshops*. IEEE, 2014, S. 226–231. DOI: 10.1109/ISSREW.2014.88 (Zitiert auf S. 60).
- [Har20] M. Hartwig. *Die fünf Stufen zum autonomen Fahren*. München, 2020. URL: <https://www.bmw.com/de/automotive-life/autonomes-fahren.html> (Zitiert auf S. 12).
- [Hil12] M. Hillenbrand. *Funktionale Sicherheit nach ISO 26262 in der Konzeptphase der Entwicklung von Elektrik/Elektronik Architekturen von Fahrzeugen: Zugl.: Karlsruhe, KIT, Dissertation, 2011*. Bd. 4. Steinbuch series on advances in information technology. Karlsruhe und Hannover: KIT Scientific Publishing and Technische Informationsbibliothek u. Universitätsbibliothek, 2012 (Zitiert auf S. 25, 32, 86).
- [Hof13] D. W. Hoffmann. *Software-Qualität*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. DOI: 10.1007/978-3-642-35700-8 (Zitiert auf S. 152, 153).
- [Hol08] G. J. Holzmann. *The Spin Model Checker: Primer and reference manual*. 4th print. Boston, Mass. und Munich: Addison-Wesley, 2008 (Zitiert auf S. 72–75, 138).
- [Int+06] International Electrotechnical Commission et al. „International standard IEC 61025 fault tree analysis“. In: *Geneva: International Electrotechnical Commission (2006)* (Zitiert auf S. 134).
- [Int11] International Organization for Standardization. *ISO 26262-Road vehicles-Functional Safety*. Genf, 2011 (Zitiert auf S. 27).

- [Int18] International Organization for Standardization. *ISO26262-2018-Road vehicles-Functional Safety*. Genf, 2018 (Zitiert auf S. 14, 22, 23, 25–33, 37–39, 54, 61, 62, 64–66, 78, 85, 86, 88, 90, 92, 99, 102, 112, 114, 115, 131, 134, 148, 154–156, 164, 189, 191, 192, 204, 208).
- [Ise16] R. Isermann. „Fehlertoleranz bei mechatronischen Systemen“. In: *Forschung im Ingenieurwesen* 80.1-2 (2016), S. 41–56. DOI: 10.1007/s10010-016-0200-2 (Zitiert auf S. 13, 28, 33, 34).
- [JMHWO5] A. Joshi, S. P. Miller, M. P. Heimdahl, M. Whalen. *Model-Based Safety Analysis*. Hrsg. von National Aeronautics and Space Administration. Langley, VA, 2005 (Zitiert auf S. 46).
- [JZW19] Q. Jiang, C. Zhu, S. Wang. „Qualitative analysis for state/event fault trees using formal model checking“. In: *Journal of Systems Engineering and Electronics* 30.5 (2019), S. 959. DOI: 10.21629/JSEE.2019.05.13 (Zitiert auf S. 76).
- [Joh87] B. D. Johnston. „A structured procedure for dependent failure analysis (DFA)“. In: *Reliability Engineering* 19.2 (1987), S. 125–136. DOI: 10.1016/0143-8174(87)90107-7 (Zitiert auf S. 61).
- [Jon12] H. Jones. „Common Cause Failures and Ultra Reliability“. In: *42nd International Conference on Environmental Systems* (2012), S. 127. DOI: 10.2514/6.2012-3602 (Zitiert auf S. 132, 133).
- [KKS+06] A. Kohn, M. Kabmeyer, R. Schneider, A. Roger, C. Stellwag, A. Herkersdorf. „Fail-operational in safety-related automotive multi-core systems“. In: *10th IEEE International Symposium on Industrial Embedded Systems (SIES)*. IEEE, 8.06.2015 - 10.06.2015, S. 1–4. DOI: 10.1109/SIES.2015.7185051 (Zitiert auf S. 34, 57, 58).
- [KL18] M. Kölbl, S. Leue. „Automated Functional Safety Analysis of Automated Driving Systems“. In: *Formal Methods for Industrial Critical Systems*. Hrsg. von F. Howar, J. Barnat. Bd. 11119. Cham: Springer International Publishing, 2018, S. 35–51. DOI: 10.1007/978-3-030-00244-2\_3 (Zitiert auf S. 38, 56, 67).



- [KLN+15] J. H. Kim, K. G. Larsen, B. Nielsen, M. Mikučionis, P. Olsen. „Formal Analysis and Testing of Real-Time Automotive Systems Using UPPAAL Tools“. In: *Formal Methods for Industrial Critical Systems*. Hrsg. von M. Núñez, M. Güdemann. Bd. 9128. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015, S. 47–61. DOI: 10.1007/978-3-319-19458-5\_4 (Zitiert auf S. 69).
- [KSA+18] B. Kaiser, D. Schneider, R. Adler, D. Domis, F. Möhrle, A. Berres, M. Zeller, K. Höfig, M. Rothfelder. „Advances in component fault trees“. In: *Safety and Reliability - Safe Societies in a Changing World*. Hrsg. von T. Kongsvik, J. Erik Vinnem, S. Haugen, C. van Gulijk, A. Barros. Taylor & Francis, 2018 (Zitiert auf S. 76).
- [KST+19] A. Kron, I. Schaffer, J. Tchai, K.-H. Meitinger, S. Schraufstetter. „Motion control solutions for automated driving systems at BMW“. In: *19th Proceedings of Stuttgarter International Symposium (2019)*, S. 1–13. DOI: 10.1007/978-3-658-25939-6\_1 (Zitiert auf S. 13, 34–36).
- [Kai06] B. Kaiser. „State/Event Fault Trees: A Safety and Reliability Analysis Technique for Software-Controlled Systems“. Dissertation. Kaiserslautern: Technischen Universität Kaiserslautern, 2006. DOI: 10.13140/2.1.4362.5601 (Zitiert auf S. 75, 76).
- [Kel98] T. Kelly. „Arguing Safety - A Systematic Approach to Safety Case Management“. Dissertation. York: University of York, 1998 (Zitiert auf S. 32, 33, 87, 99).
- [Kle09] S. Kleuker. *Formale Modelle der Softwareentwicklung: Model-Checking, Verifikation, Analyse und Simulation*. 1. Aufl. Datenbanken und Softwareentwicklung. Wiesbaden: Vieweg+Teubner Verlag / GWV Fachverlage GmbH Wiesbaden, 2009. DOI: 10.1007/978-3-8348-9595-0 (Zitiert auf S. 47, 52).
- [Koc18] I. Kocsis. „Design for Dependability Through Error Propagation Space Exploration“. In: *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. 2018, S. 172–178. DOI: 10.1109/DSN-W.2018.00059 (Zitiert auf S. 11, 12).

- [Kue18] M. Kuemmel. „Verfahren zur fehlerrobusten Regelung von hochautomatisierten Fahrzeugen“. Pat. DE102017218395A1. 2018 (Zitiert auf S. 36, 158).
- [Kug12] S. Kugele. „Model-Based Development of Software-intensive Automotive Systems“. Dissertation. München: Technische Universität München, 2012 (Zitiert auf S. 78).
- [LA04] R. A. La Band, J. D. Andrews. „Phased mission modelling using fault tree analysis“. In: *Proceedings of the Institution of Mechanical Engineers, Part E: Journal of Process Mechanical Engineering* 218.2 (2004), S. 83–91. DOI: 10.1243/095440804774134262 (Zitiert auf S. 80, 81, 195).
- [LDC09] M. Leeman, P. Degoul, P. Chaussis. „Independence and Non-interference: Two Cardinal Concepts to Develop EE Architectures Hosting Safety-Critical Systems“. In: *SAE Technical Paper*. 2009. DOI: 10.4271/2009-01-0739 (Zitiert auf S. 63).
- [LL14] X. Li, S. Li. „Graphical Modeling of System Failure Behavior and its Translating into Altarica“. In: *Procedia Engineering* 80 (2014), S. 581–591. DOI: 10.1016/j.proeng.2014.09.114 (Zitiert auf S. 76).
- [ME15] M. Moestl, R. Ernst. „Cross-Layer Dependency Analysis for Safety-Critical Systems Design“. In: *Architecture of computing systems - ARCS 2015*. Hrsg. von L. M. Pinho, W. Karl, A. Cohen, U. Brinkschulte. Lecture Notes in Computer Science. Cham: Springer, 2015 (Zitiert auf S. 63, 64, 124).
- [MNKC04] F. Mhenni, N. Nguyen, H. Kadima, J. Choley. „Safety analysis integration in a SysML-based complex system design process“. In: *2013 IEEE International Systems Conference (SysCon)*. IEEE, 15.04.2013 - 18.04.2013, S. 70–75. DOI: 10.1109/SysCon.2013.6549861 (Zitiert auf S. 46).
- [MRDM99] A. Mosleh, Rasmuson, D.M., F.M. Marshall. „Guidelines on Modeling Common-Cause Failures in Probabilistic Risk Assessment: NUREG/CR-5485“. In: *Idaho National Engineering and Environmental Laboratory, University of Maryland* (1999) (Zitiert auf S. 63, 64).

- [MSC20] A. Mehmed, W. Steiner, A. Causevic. *Formal Verification of an Approach for Systematic False Positive Mitigation in Safe Automated Driving Systems*. Hrsg. von Mälardalen Real-Time Research Centre. 2020 (Zitiert auf S. 36).
- [MSU+92] A. Misra, J. Sztipanovits, A. Underbrink, R. Carnes, B. Purves, Hrsg. *Diagnosability of Dynamical Systems*. Rosario, WA, 1992 (Zitiert auf S. 47).
- [May14] P. Mayring. *Qualitative content analysis: theoretical foundation, basic procedures and software solution*. Klagenfurt, 2014. URL: <https://www.ssoar.info/ssoar/handle/document/39517> (Zitiert auf S. 111, 112).
- [McM93] K. L. McMillan. „Symbolic Model Checking“. In: *Symbolic Model Checking*. Springer, 1993, S. 25–60. doi: 10.1007/978-1-4615-3190-6 (Zitiert auf S. 74).
- [NEA04] NEA Committee on the Safety of Nuclear Installations. *ICDE General Coding Guidelines - Technical Note: NEA/CSNI/R(2004)4*. 2004 (Zitiert auf S. 133).
- [NGL+18] M. Nyberg, D. Gurov, C. Lidström, A. Rasmusson, J. Westman. „Formal Verification in Automotive Industry: Enablers and Obstacles“. In: *Leveraging Applications of Formal Methods, Verification and Validation. Industrial Practice*. Hrsg. von T. Margaria, B. Steffen. Bd. 11247. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018, S. 139–158. doi: 10.1007/978-3-030-03427-6\_14 (Zitiert auf S. 68, 69).
- [NKF19] S. Niedermaier, F. Koetter, A. Freymann, S. Wagner. „On Observability and Monitoring of Distributed Systems – An Industry Interview Study“. In: *Service-Oriented Computing*. Hrsg. von S. Yangui, I. Bouassida Rodriguez, K. Drira, Z. Tari. Bd. 11895. Springer eBooks Computer Science. Cham: Springer, 2019, S. 36–52. doi: 10.1007/978-3-030-33702-5\_3 (Zitiert auf S. 111, 119).
- [NR20] D. Niedballa, H.-C. Reuss. „Concepts of functional safety in E/E-architectures of highly automated and autonomous vehicles“. In: *20. Internationales Stuttgarter Symposium*. Hrsg. von M. Bargende,

- H.-C. Reuss, A. Wagner. Proceedings. Wiesbaden: Springer Fachmedien Wiesbaden, 2020, S. 457–470. DOI: 10.1007/978-3-658-30995-4\_41 (Zitiert auf S. 36, 58).
- [NRW+18] J. Nellen, T. Rambow, M. T. B. Waez, E. Ábrahám, J.-P. Katoen. „Formal Verification of Automotive Simulink Controller Models: Empirical Technical Challenges, Evaluation and Recommendations“. In: *Formal Methods*. Hrsg. von K. Havelund, J. Peleska, B. Roscoe, E. de Vink. Bd. 10951. SpringerLink Bücher. Cham: Springer International Publishing, 2018, S. 382–398. DOI: 10.1007/978-3-319-95582-7\_23 (Zitiert auf S. 68–70).
- [Nen07] P. Nenninger. „Vernetzung verteilter sicherheitsrelevanter Systeme im Kraftfahrzeug“. Dissertation. Karlsruhe: Universität Karlsruhe, 2007. DOI: 10.5445/KSP/1000006495 (Zitiert auf S. 58).
- [Nyb13] M. Nyberg. „Failure propagation modeling for safety analysis using causal Bayesian networks“. In: *2013 Conference on Control and Fault-Tolerant Systems (SysTol 2013)*. Piscataway, NJ: IEEE, 2013, S. 91–97. DOI: 10.1109/SysTol.2013.6693936 (Zitiert auf S. 47).
- [OM16] A. O’Connor, A. Mosleh. „A general cause based methodology for analysis of common cause and dependent failures in system risk and reliability assessments“. In: *Reliability Engineering & System Safety* 145 (2016), S. 341–350. DOI: 10.1016/j.res.2015.06.007 (Zitiert auf S. 131).
- [O’R17] G. O’Regan. *Concise Guide to Formal Methods: Theory, Fundamentals and Industry Applications*. Undergraduate Topics in Computer Science. Cham: Springer International Publishing, 2017. DOI: 10.1007/978-3-319-64021-1 (Zitiert auf S. 47–49, 52).
- [Osz21] F. Oszwald. „Dynamische Rekonfigurationsmethodik für zuverlässige, echtzeitfähige Eingebettete Systeme in Automotive“. Dissertation. Karlsruhe: Karlsruher Institut für Technologie, 2021. DOI: 10.5445/KSP/1000132321 (Zitiert auf S. 56).
- [PH10] R. Palin, I. Habli. „Assurance of Automotive Safety – A Safety Case Approach“. In: *Computer Safety, Reliability, and Security*. Hrsg. von D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern,

- J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, E. Scoitsch. Bd. 6351. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, S. 82–96. DOI: 10.1007/978-3-642-15651-9\_7 (Zitiert auf S. 59, 60).
- [PL20] G. Prashanth Reddy, R. Leburu. „Matrix Approach to Perform Dependent Failure Analysis in Compliance with Functional Safety Standards“. In: *Proceedings of the Third International Conference on Computational Intelligence and Informatics*. Hrsg. von K. S. Raju, A. Govardhan, B. P. Rani, R. Sridevi, M. R. Murty. Bd. 1090. Advances in Intelligent Systems and Computing. Singapore: Springer Singapore, 2020, S. 123–131. DOI: 10.1007/978-981-15-1480-7\_10 (Zitiert auf S. 65).
- [PMSH01] Y. Papadopoulos, J. McDermid, R. Sasse, G. Heiner. „Analysis and synthesis of the behaviour of complex programmable electronic systems in conditions of failure“. In: *Reliability Engineering & System Safety* 71.3 (2001), S. 229–247 (Zitiert auf S. 45).
- [PRŠ08] R. Pelánek, V. Rosecký, J. Šeděnka. *Evaluation of State Caching and State Compression Techniques: Technical Report*. Brno, 2008 (Zitiert auf S. 75).
- [PS16] S. Pischinger, U. Seiffert, Hrsg. *Vieweg Handbuch Kraftfahrzeugtechnik*. 8., aktualisierte und erweiterte Auflage. ATZ / MTZ-Fachbuch. Wiesbaden: Springer Vieweg, 2016. DOI: 10.1007/978-3-658-09528-4 (Zitiert auf S. 21).
- [RGD07] W. Ridderhof, H.-G. Gross, H. Doerr. „Establishing Evidence for Safety Cases in Automotive Systems – A Case Study“. In: *International Conference on Computer Safety, Reliability, and Security*. Bd. 4680. 2007, S. 1–13. DOI: 10.1007/978-3-540-75101-4\_1 (Zitiert auf S. 59).
- [RH09] P. Runeson, M. Höst. „Guidelines for conducting and reporting case study research in software engineering“. In: *Empirical Software Engineering* 14.2 (2009), S. 131–164. DOI: 10.1007/s10664-008-9102-8 (Zitiert auf S. 119).

- [RL15] M. Roth, P. Liggesmeyer. „Sequential Logic for State/Event Fault Trees: A Methodology to Support the Failure Modeling of Cyber Physical Systems“. In: *Computer Safety, Reliability, and Security*. Hrsg. von F. Koornneef, C. van Gulijk. Bd. 9338. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015, S. 121–132. DOI: 10.1007/978-3-319-24249-1\_11 (Zitiert auf S. 76).
- [RPV+01] S. Reiter, M. Pressler, A. Viehl, O. Bringmann, W. Rosenstiel. „Reliability assessment of safety-relevant automotive systems in a model-based design flow“. In: *2013 18th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 22.01.2013 - 25.01.2013, S. 417–422. DOI: 10.1109/ASPDAC.2013.6509632 (Zitiert auf S. 77, 169).
- [RS15] E. Ruijters, M. Stoelinga. „Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools“. In: *Computer Science Review* 15-16 (2015), S. 29–62. DOI: 10.1016/j.cosrev.2015.03.001 (Zitiert auf S. 42, 78, 79).
- [Rei14] K. Reif. *Automobilelektronik: Eine Einführung für Ingenieure*. 5., überarb. Aufl. ATZ-MTZ-Fachbuch. Wiesbaden: Springer Fachmedien Wiesbaden und Springer Vieweg, 2014. DOI: 10.1007/978-3-658-05048-1 (Zitiert auf S. 77, 172, 173).
- [Ros16] H.-L. Ross. *Functional Safety for Road Vehicles: New Challenges and Solutions for E-mobility and Automated Driving*. Springer, 2016 (Zitiert auf S. 27, 29, 30, 32).
- [SAE96] SAE International Standard. *Aerospace recommended practice- Guidelines and Methods for conducting the safety assessment process on civil airborne systems and equipment: ARP 4761*. 1996 (Zitiert auf S. 64, 122).
- [SBRM06] T. Stolte, G. Bagschik, A. Reschka, M. Maurer. „Hazard analysis and risk assessment for an automated unmanned protective vehicle“. In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 11.06.2017 - 14.06.2017, S. 1848–1855. DOI: 10.1109/IVS.2017.7995974 (Zitiert auf S. 38).

- [SDWB17] P. Schleiss, C. Drabek, G. Weiss, B. Bauer. „Generic Management of Availability in Fail-Operational Automotive Systems“. In: *Computer safety, reliability, and security*. Hrsg. von S. Tonetta, E. Schoitsch, F. Bitsch. Bd. 10488. Lecture Notes in Computer Science. Cham: Springer, 2017, S. 179–194. DOI: 10.1007/978-3-319-66266-4\_12 (Zitiert auf S. 57, 58).
- [SH19] L. Schnieder, R. S. Hosse. *Leitfaden Safety of the Intended Functionality*. Wiesbaden: Springer Fachmedien Wiesbaden, 2019. DOI: 10.1007/978-3-658-25023-2 (Zitiert auf S. 21, 22).
- [ST12] T. Streichert, M. Traub. *Elektrik/Elektronik-Architekturen im Kraftfahrzeug: Modellierung und Bewertung von Echtzeitsystemen*. 1. Aufl. VDI-Buch. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. DOI: 10.1007/978-3-642-25478-9 (Zitiert auf S. 142, 173).
- [Sar19] B. Sari. *Fail-operational safety architecture for ADAS/AD systems and a model-driven approach for dependent failure analysis: Zugl.: Stuttgart, Universität Stuttgart, Dissertation, 2019*. Wissenschaftliche Reihe Fahrzeugtechnik Universität Stuttgart. Wiesbaden: Springer Vieweg, 2019. DOI: 10.1007/978-3-658-29422-9 (Zitiert auf S. 46, 55, 56, 58, 66).
- [Sau99] T. Sauerbier. *Theorie und Praxis von Simulationssystemen: Eine Einführung für Ingenieure und Informatiker*. Studium Technik. Wiesbaden: Vieweg+Teubner Verlag, 1999. DOI: 10.1007/978-3-322-90773-8 (Zitiert auf S. 152).
- [Sch09] S. J. Schilling. „Beitrag zur dynamischen Fehlerbaumanalyse ohne Modulbildung und zustandsbasierte Erweiterungen“. Dissertation. Wuppertal: Bergischen Universität Wuppertal, 2009 (Zitiert auf S. 79).
- [Sch16] A. Schnellbach. „Fail-operational automotive systems“. Dissertation. Graz: Technische Universität Graz, 2016 (Zitiert auf S. 13, 25, 27, 33, 34, 54–56, 61–66, 78, 82, 88, 123, 128, 166).
- [Sea99] C. B. Seaman. „Qualitative methods in empirical studies of software engineering“. In: *IEEE Transactions on Software Engineering* 25.4 (1999), S. 557–572. DOI: 10.1109/32.799955 (Zitiert auf S. 111, 112, 120).

- [Sin11] P. Sinha. „Architectural design and reliability analysis of a fail-operational brake-by-wire system from ISO 26262 perspectives“. In: *Reliability Engineering & System Safety* 96.10 (2011), S. 1349–1359. doi: 10.1016/j.ress.2011.03.013 (Zitiert auf S. 56).
- [TAR20] J. Toennemann, A. Aniculăesei, A. Rausch. „Asserting Functional Equivalence between C Code and SCADE Models in Code-to-Model Transformations“. In: *Proceedings of the 5th Brazilian Symposium on Systematic and Automated Software Testing*. Hrsg. von E. Cavalcante, F. Dantas, T. Batista. New York, NY, USA: ACM, 2020, S. 60–68. doi: 10.1145/3425174.3425213 (Zitiert auf S. 71).
- [TBT18] V. Todorov, F. Boulanger, S. Taha. „Formal verification of automotive embedded software“. In: *FORMALISE: 6th International Conference on Formal Methods in Software Engineering*. Gotheborg, Sweden, 2018, S. 84–87. doi: 10.1145/3193992.3194003 (Zitiert auf S. 68, 69).
- [TTS98] J. G. Torres-Toledano, L. E. Sucar. „Bayesian Networks for Reliability Analysis of Complex Systems“. In: *Progress in Artificial Intelligence - IBERAMIA 98*. Hrsg. von H. Coelho. Bd. 1484. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, S. 195–206. doi: 10.1007/3-540-49795-1\_17 (Zitiert auf S. 47).
- [The19] I. The MathWorks. *Simulink Design Verifier: User’s Guide*. 2019. URL: <https://www.mathworks.com/products/simulink-design-verifier.html> (Zitiert auf S. 138).
- [UDR11] A. Udupa, A. Desai, S. Rajamani. „Depth Bounded Explicit-State Model Checking“. In: *Model Checking Software*. Hrsg. von A. Groce, M. Musuvathi. Bd. 6823. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, S. 57–74. doi: 10.1007/978-3-642-22306-8\_5 (Zitiert auf S. 73).
- [VGRH81] W. Vesely, F. F. Goldberg, N. H. Roberts, D. F. Haasl. *Fault Tree Handbook*. Hrsg. von Office of Nuclear Regulatory Research. Washington, DC, 1981 (Zitiert auf S. 42).
- [VSD+02] W. Vesely, M. Stamatelatos, J. Dugan, J. Fragola, J. Minarick, J. Railsback. *Fault Tree Handbook with Aerospace Applications*. Washington, DC, 2002 (Zitiert auf S. 79, 80, 195, 212).



- [VWTB+98] T. Villa, H. Wong-Toi, A. Balluchi, J. Preussig, A. L. Sangiovanni-Vincentelli, Y. Watanabe. „Formal verification of an automotive engine controller in cutoff mode“. In: *Proceedings of the 37th IEEE Conference on Decision and Control (Cat. No.98CH36171)*. IEEE, 1998, S. 4271–4276. DOI: 10.1109/CDC.1998.761976 (Zitiert auf S. 69).
- [Vau01] J. K. Vaurio. „Fault tree analysis of phased mission systems with repairable and non-repairable components“. In: *Reliability Engineering & System Safety* 74.2 (2001), S. 169–180. DOI: 10.1016/S0951-8320(01)00075-8 (Zitiert auf S. 80, 195).
- [Ver15a] Verband der Automobilindustrie. *Automatisierung–von Fahrerassistenzsystemen zum automatisierten Fahren*. Berlin, 2015 (Zitiert auf S. 11, 12).
- [Ver15b] Verband der Automobilindustrie. *Situationskatalog: E-Parameter nach ISO 26262-3*. Berlin, 2015 (Zitiert auf S. 204, 210).
- [Ver17] Verband der Automobilindustrie. *FMEA: Fehler-Möglichkeiten-und-Einfluss-Analyse*. Berlin, 2017 (Zitiert auf S. 44, 45).
- [Vol20] P. Vollmer. „Systems-Engineering-Methoden für die Entwicklung hochautomatisierter Fahrfunktionen“. In: *ATZextra* 25.S3 (2020), S. 22–25. DOI: 10.1007/s35778-020-0128-x (Zitiert auf S. 137, 163).
- [WBE+ 15] T. Weilkens, A. Berres, D. Endler, A. Haarer, C. Lalitsch-Schneider, M. Krammer, H. Martin. „System Safety in SysML“. In: *Tag des Systems Engineering*. 2015, S. 343–354. DOI: 10.3139/9783446447288.034 (Zitiert auf S. 46).
- [WHLS15] H. Winner, S. Hakuli, F. Lotz, C. Singer, Hrsg. *Handbuch Fahrerassistenzsysteme: Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort*. 3., überarb. und erg. Aufl. ATZ/MTZ-Fachbuch. Wiesbaden: Springer Vieweg, 2015. DOI: 10.1007/978-3-658-05734-3 (Zitiert auf S. 11, 14, 22).
- [WKP09] M. Wehrle, S. Kupferschmid, A. Podelski. „Transition-Based Directed Model Checking“. In: *Tools and Algorithms for the Construction and Analysis of Systems*. Hrsg. von S. Kowalewski, A. Philippou. Bd. 5505. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer

Berlin Heidelberg, 2009, S. 186–200. DOI: 10.1007/978-3-642-00768-2\_19 (Zitiert auf S. 74, 75, 178).

- [WO20] J. Wolf, M. Oertel. „Sichere Software für automatisiertes Fahren“. In: *ATZelektronik* 15.7-8 (2020), S. 50–53. DOI: 10.1007/s35658-020-0220-4 (Zitiert auf S. 36).
- [WRH+12] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén. *Experimentation in software engineering*. Berlin und Heidelberg: Springer, 2012. DOI: 10.1007/978-3-642-29044-2 (Zitiert auf S. 119).
- [WRHM06] M. W. Whalen, A. Rajan, M. P. Heimdahl, S. P. Miller. „Coverage metrics for requirements-based testing“. In: *Proceedings of the 2006 international symposium on Software testing and analysis - ISSA'06*. Hrsg. von L. Pollock, M. Pezzè. New York, New York, USA: ACM Press, 2006, S. 25. DOI: 10.1145/1146238.1146242 (Zitiert auf S. 71).
- [WRM+19] M. Wood, P. Robbel, M. Maass, R. D. Tebbens, Meijsm Marc, M. Harb, J. Reach, K. Robinson, D. Wittmann, T. Srivastava, M. E. Bouzou-raa, S. Liu, Y. Wang, C. Knobel, D. Boymanns, M. Löhning, B. Dehlink, D. Kaule, R. Krüger, J. Frtunikj, F. Raisch, M. Gruber, J. Steck, J. Mejia-Hernandez, S. Syguda, P. Blüher, K. Klonecki, P. Schnarz, T. Wiltschko, S. Pukallus, K. Sedlaczek, N. Garbacik, D. Smerzea, D. Li, A. Timmons, M. Belotti, M. O'Brien, M. Schöllhorn, U. Dannebaum, J. West, A. Tatourian, B. Dornieden, P. Schnetter, P. Themann, T. Weidner, P. Schlicht. „Safety First for Automated Driving“. 2019. URL: <https://www.bmwgroup.com/en/company/bmw-group-news/artikel/Safety-First-for-Automated-Driving.html> (Zitiert auf S. 11, 220).
- [WSPK10] S. Wagner, B. Schatz, S. Puchner, P. Kock. „A Case Study on Safety Cases in the Automotive Domain: Modules, Patterns, and Models“. In: *2010 IEEE 21st International Symposium on Software Reliability Engineering*. 2010, S. 269–278. DOI: 10.1109/ISSRE.2010.31 (Zitiert auf S. 59).
- [Wan17] P. Wang. „Common Cause Analysis: Civil Aircraft Electrical Power System Safety Assessment“. In: *Civil aircraft electrical power system safety assessment*. Hrsg. von P. Wang. Kidlington, Oxford und Cambridge,

MA: Butterworth-Heinemann, 2017, S. 157–186. DOI: 10.1016/B978-0-08-100721-1.00006-6 (Zitiert auf S. 63, 64, 122).

- [Wer12] M. Werdich. *FMEA - Einführung und Moderation: Durch systematische Entwicklung zur übersichtlichen Risikominimierung (inkl. Methoden im Umfeld)*. 2. Auflage. Wiesbaden: Springer Vieweg, 2012. DOI: 10.1007/978-3-8348-2217-8 (Zitiert auf S. 44).
- [YD98] C. H. Yang, D. L. Dill. „Validation with guided search of the state space“. In: *Proceedings of the 35th annual conference on Design automation conference - DAC '98*. Hrsg. von B. R. Chawla, R. E. Bryant, J. M. Rabaey, M. J. Irwin. New York, New York, USA: ACM Press, 1998, S. 599–604. DOI: 10.1145/277044.277201 (Zitiert auf S. 73).
- [ZWV+19] B. Zhang, J. de Winter, S. Varotto, R. Happee, M. Martens. „Determinants of take-over time from automated driving: A meta-analysis of 129 studies“. In: *Transportation Research Part F: Traffic Psychology and Behaviour* 64 (2019), S. 285–307. DOI: 10.1016/j.trf.2019.04.020 (Zitiert auf S. 209).

## Publikationen im Rahmen der Arbeit

- [FSW21] J. Fritzscht, T. Schmid, S. Wagner. „Experiences from Large-Scale Model Checking: Verifying a Vehicle Control System with NuSMV“. In: *2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2021, S. 372–382. DOI: 10.1109/ICST49551.2021.00049.
- [SSF+21] T. Schmid, S. Schraufstetter, J. Fritzscht, D. Hellhake, G. Koelln, S. Wagner. *Formal Verification of a Fail-Operational Automotive Driving System*. 2021. URL: <https://arxiv.org/pdf/2101.07307>.
- [SSHK20] T. Schmid, S. Schraufstetter, T. Hagler, D. Krepis. „Quantitative Analyse von Fail-Operational Hardwarearchitekturen“. In: *safe.tech-Funktionale Sicherheit in der Bahntechnik, Automatisierung und Automobiltechnik*. 2020.
- [SSW18] T. Schmid, S. Schraufstetter, S. Wagner. „An Approach for Structuring a Highly Automated Driving Multiple Channel Vehicle System for Safety Analysis“. In: *2018 3rd International Conference on System Reliability and Safety (ICSRS)*. 2018, S. 362–367. DOI: 10.1109/ICSRS.2018.8688859.
- [SSWH19] T. Schmid, S. Schraufstetter, S. Wagner, D. Hellhake. „A Safety Argumentation for Fail-Operational Automotive Systems in Compliance with ISO 26262“. In: *2019 4th International Conference on System Reliability and Safety (ICSRS)*. IEEE, 2019, S. 484–493. DOI: 10.1109/ICSRS48664.2019.8987656.
- [Sch10] T. Schmid. „Safety Analysis for Highly Automated Driving“. In: *2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE, 15.10.2018 - 18.10.2018, S. 154–157. DOI: 10.1109/ISSREW.2018.000-7.

Alle URLs wurden zuletzt am 21.12.2021 geprüft.

# ABBILDUNGSVERZEICHNIS

1.1	Definition der Stufen des automatisierten Fahrens nach Aufgabe des Fahrers und des Systems gemäß VDA [Ver15a] . . .	12
1.2	Aufbau der Arbeit . . . . .	17
2.1	Aufbau und Umfang des Industriestandards ISO 26262 [Int18]	23
2.2	Fehlerbegriffe und Fehlerwirkkette entsprechend ISO 26262 [Int18] . . . . .	25
2.3	Fehlerarten nach ISO 26262 [Int18] . . . . .	26
2.4	Zusammenhang der Fehlerursache und Fehlerauswirkung bei abhängigen Fehlern [Int18] . . . . .	27
2.5	Fehlertoleranzzeit und Fehlerbehandlungszeit nach ISO 26262 [Int18] . . . . .	28
2.6	Phasen des Sicherheitslebenszyklus nach ISO 26262 [Int18] .	30
2.7	Schritte der sicherheitsgerichteten Produktentwicklung anhand des V-Models nach ISO 26262 am Beispiel der Softwareentwicklung [Int18] . . . . .	31
2.8	Symbole der Goal Structuring Notation nach Kelly [Kel98] . .	33
2.9	Fail-Operational-Fahrzeugführung nach Kron et al. [KST+19]	35
2.10	Umschaltlogik für eine Fail-Operational-Fahrzeugführung nach Kron et al. [KST+19] . . . . .	36

2.11	Fail-Operational-Betriebsstrategie nach ISO 26262 [Int18] . .	37
2.12	logische Gatter bei der Fehlerbaumanalyse nach Edler [ESH15]	41
2.13	Beispiel eines Fehlerbaumes nach Edler [ESH15] . . . . .	42
2.14	Beispiel eines Markov-Modells . . . . .	43
2.15	Vorgehen bei der Fehlermöglichkeits- und Einflussanalyse nach dem VDA [Ver17] . . . . .	44
2.16	Schritte des Model-Checkings nach Clarke [CHVB18] . . . . .	48
2.17	Suchstrategien in einem Zustandsbaum . . . . .	49
3.1	Abhängigkeitsmodell nach Schnellbach [Sch16] (vgl. [Int18])	65
3.2	Berücksichtigung verschiedener Betriebsphasen im Fehler- baum nach La Band und Andrews [LA04] . . . . .	81
4.1	Vorgehen für das Ableiten einer Sicherheitsargumentation . .	86
4.2	Sicherheitsrelevante Attribute der Sicherheitsanforderungen nach ISO 26262 [Int18] . . . . .	90
4.3	Zustandsdarstellung des Fail-Operational-Systemsverhaltens .	92
4.4	Aktivitätsdiagramm der Reaktion des Systems auf Komponen- tenebene im Fall eines Doppelfehlers . . . . .	94
4.5	Schematischer Fehlerbaum für das Hauptereignis <i>Kollision</i> <i>durch ein Verlassen der Trajektorie</i> . . . . .	95
4.6	Fehlerbaumlogik für einen Fehler bei der Umschaltung . . . .	98
4.7	Wichtigste Symbole der Sicherheitsargumentation auf Basis der Goal Structuring Notation . . . . .	100
4.8	Ziele im Rahmen der Sicherheitsargumentation . . . . .	101
4.9	Sicherheitsargumentation für ein Fail-Operational-System, dargestellt anhand einer Goal Structuring Notation (Teil 1) .	103
4.10	Sicherheitsargumentation für ein Fail-Operational-System, dargestellt anhand einer Goal Structuring Notation (Teil 2) .	104
5.1	Identifizierte Stakeholder der Common-Cause Analyse . . . .	113
5.2	Identifizierte Ziele der Analyse gemeinsamer Ausfälle . . . .	115

5.3	Identifizierte Anforderungen an das Vorgehen bei der Analyse gemeinsamer Ausfälle . . . . .	116
5.4	Identifizierte Anforderungen an die Dokumentation bei der Analyse gemeinsamer Ausfälle . . . . .	117
5.5	Erweitertes Vorgehen für die Analyse gemeinsamer Ausfälle für Fail-Operational-Systeme . . . . .	121
5.6	Datenmodell für die Dokumentation der Analyse gemeinsamer Ausfälle . . . . .	124
5.7	Fehlerbaum zur Identifikation von Unabhängigkeiten . . . . .	127
5.8	Beispielhafter Fehlerbaum für die quantitative Bestimmung gemeinsamer Ausfälle . . . . .	132
6.1	Definierte Fehlerzeitpunkte im Rahmen der Prüfung . . . . .	151
6.2	Vorgehen bei der Analyse und Qualifizierung von Software-Werkzeugen nach ISO 26262 [Int18] . . . . .	155
6.3	Verteilung der Rechenzeiten beim Prüfen von Einfachfehlern in Abhängigkeit des Zielbetriebs . . . . .	160
6.4	Verteilung der Rechenzeiten beim Prüfen von Zweifachfehlern in Abhängigkeit des Zielbetriebs . . . . .	161
7.1	Klassendiagramm des für die Fehlerpropagation relevanten Systemmodells . . . . .	167
7.2	Logik der modularen Fehlerbäume für Software-Komponenten	170
7.3	Blockdiagramm der vereinfachten Wirkkette als Validierungsbeispiel . . . . .	181
7.4	Rechenzeiten in Abhängigkeit der geprüften und bereits besuchten Zustände . . . . .	185
8.1	Relevante Betriebsphasen für die Ausfallwahrscheinlichkeit im hochautomatisierten Fahrmodus . . . . .	194
8.2	Aufbau eines Fehlerbaums unter Berücksichtigung der Betriebsphasen . . . . .	198

8.3	Detaillierung des Fehlerbaums für den Rückfallbetrieb auf dem Rückfallkanal . . . . .	199
8.4	Schematischer Aufbau des Markov-Modells . . . . .	201
8.5	Detaillierung der Markov-Modells anhand der Basisereignisse	202
8.6	Berechnung des Markov-Modells über die Betriebsphasen . .	203
8.7	Abweichungen zwischen der anhand Fehlerbaum und Markov-Modell berechneten Ausfallwahrscheinlichkeit . . . .	204
8.8	Abweichungen zwischen der Berechnung im Fehlerbaum und im Markov-Modell bei Parametervariation . . . . .	207
8.9	Sensitivität der Ausfallwahrscheinlichkeit bei Parametervariationen . . . . .	211
8.10	Birnbau-Importanzen des der Parameter des Fail-Operational-Systems . . . . .	213



# TABELLENVERZEICHNIS

2.1	Sicherheitsziele des Fail-Operational-Systems . . . . .	39
2.2	relevante Ausdrücke der linear temporalen Logik . . . . .	51
5.1	Struktur und Beispiel der Fallstudie Fahrzeugführung . . . . .	129
6.1	Aufbau der Matrix für die von Fehlerkombinationen und Betriebszustand . . . . .	150
6.2	Äquivalenzklassenkonzept für die Validierung der formalen Spezifikationen . . . . .	153
8.1	Grenzwerte für Sicherheitszielverletzungen durch zufällige Hardwarefehler nach ISO 26262 [Int18] . . . . .	192



# ABKÜRZUNGSVERZEICHNIS

AADL	Architecture Analysis and Design Language
ACC	Adaptive-Cruise Control
ADL	Architecture Description Language
ASIL	Automotive Safety Integrity Level
Autosar	Automotive Open System Architecture
BFS	Breadth-First Search
BMW	Bayerische Motorenwerke
BR	Bremsregelsystem
CCA	Common-Cause Analysis
CCI	Common-Cause Initiator
CPU	Central Processing Unit
CRC	Cyclic-Redundancy-Check
CTL	Computational Tree Logic
DFA	Dependent Failure Analysis
DFS	Depth-First Search

E/E	Elektrik und Elektronik
EAST-ADL	Electronics Architecture and Software Technology - Architecture Description Language
EPS	Electric-Power Steering
FF	Fahrzeugführung
FIT	Failures in Time
FMEA	Fehlermöglichkeits- und Einflussanalyse
FPTN	Failure Propagation and Transformation Notation
FS	Fahrstrategie
FTA	Fault-Free Analysis
GSN	Goal Structuring Notation
HAZOP	Hazard and Operability Analysis
Hip-Hops	Hierarchically Performed Hazard Origin and Propagation Studies
LTL	Linear Temporal Logic
NASA	National Aeronautics and Space Administration
NB	Nominalbetrieb
OEM	Original Equipment Manufacturer
PMHF	Probabilistic Metric of Random Hardware Faults
RB	Rückfallbetrieb
RTE	Runtime Environment
SAT	Satisfiability

SW	Software
SysML	System Modeling Language
TCL	Tool Confidence Level
TDL	Tool Detection Level
TFP	Timed Failure Propagation
TIL	Tool Impact Level
TPN	Timed Petri Net
UML	Unified Modeling Language
VDA	Verband der deutschen Automobilindustrie