

Qualitätsmanagement in der Automobilindustrie

Technische Risikoanalyse von softwarebasierten Funktionalitäten

Qualitätsmanagement in der Automobilindustrie

Technische Risikoanalyse von softwarebasierten Funktionalitäten

1. Auflage, November 2025
Online-Download-Dokument
Verband der Automobilindustrie e. V. (VDA)

ISSN 0943-9412

Copyright 2025 by

Verband der Automobilindustrie e. V. (VDA)
Qualitäts Management Center (QMC)
10117 Berlin, Behrenstraße 35

Online-Download-Dokument

Unverbindliche Empfehlung des VDA

Der Verband der Automobilindustrie (VDA) empfiehlt seinen Mitgliedern, den nachstehenden VDA-Band bei der Einführung und Aufrechterhaltung von QM-Systemen anzuwenden.

Haftungsausschluss

Dieser VDA-Band ist eine Empfehlung, die allen frei zur Anwendung steht. Wer sie anwendet, hat im konkreten Fall für die richtige Anwendung Sorge zu tragen.

Dieser VDA-Band berücksichtigt die zum Zeitpunkt der jeweiligen Ausgabe bekannten technischen Verfahrensweisen. Durch das Anwenden der VDA-Empfehlungen entzieht sich niemand der Verantwortung für sein eigenes Handeln. Alle handeln selbstverantwortlich.

Eine Haftung des VDA und der Personen, die an der Erstellung der VDA-Empfehlungen beteiligt sind, ist ausgeschlossen.

Wer bei der Anwendung dieser VDA-Empfehlung auf Unrichtigkeiten oder die Möglichkeit einer unrichtigen Auslegung stößt, wird gebeten, dies dem VDA umgehend mitzuteilen, damit etwaige Mängel beseitigt werden können.

Urheberrechtsschutz

Diese Schrift ist urheberrechtlich geschützt. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des VDA unzulässig und strafbar. Dies gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Verarbeitung in elektronischen Systemen.

Übersetzungen

Das deutsche Dokument ist das Original. Bei Auslegungsfragen in anderen Sprachversionen ist auf die deutsche Version als Original Bezug zu nehmen. Diese Schrift wird auch in anderen Sprachen erscheinen. Der jeweils aktuelle Stand ist bei VDA QMC zu erfragen.

Vorwort

Dieser Band ist das Ergebnis eines dynamischen Arbeitsprozesses – entstanden in kurzer Zeit, getragen von hoher fachlicher Konzentration und methodischer Klarheit. Die Geschwindigkeit, mit der Anforderungen, Erfahrungen und Lösungsansätze zusammengeführt wurden, spiegelt die Dringlichkeit wider, mit der softwarebezogene Risiken heute adressiert werden müssen.

Statt bestehende Verfahren lediglich zu erweitern, wurde bewusst ein neuer Zugang gewählt: strukturiert, praxisnah und anschlussfähig an etablierte Qualitätsprozesse. Die Inhalte dieses Bandes bieten Orientierung in einem komplexen Themenfeld – nicht als starres Regelwerk, sondern als methodischer Kompass für die Risikoanalyse softwarebasierter Funktionalitäten im automobilen Kontext.

Inhaltsverzeichnis

Inhaltsverzeichnis	5	
1	Einleitung – Motivation und Zielsetzung	8
2	Nutzen, Geltungsbereich und Zielgruppe	9
2.1	Nutzen	9
2.1.1	Ziele und Vorteile der Risikoanalysen	9
2.1.2	Betrachtete technische Risiken	9
2.2	Geltungsbereich	10
2.2.1	Ziele und Vorteile der Risikoanalysen	10
2.2.2	Besonderer Fokus: Fahrzeugvernetzung	12
2.3	Zielgruppe	13
3	Begriffe und Definitionen	15
3.1	Funktionsarchitektur	15
3.2	Hardware (HW)	15
3.3	Induktive versus deduktive Methoden	15
3.4	Risiko	15
3.5	Software (SW)	16
3.6	Softwarebasierte Funktionalität	16
3.7	V2X (Vehicle to Everything)	16
4	Grundprinzipien der Risikoanalyse	17
4.1.1	Vorbedingungen zur Durchführung einer Risikoanalyse	17
4.1.2	Systematischer Ansatz zur Identifizierung der Risiken	17
4.2	Systemzerlegung und Interaktion an den Schnittstellen	19
5	Leitfaden zur Methodenauswahl	20
5.1	Anleitung zur Auswahl von Risikoanalyse-Methoden	20
5.1.1	Phasendefinition und zeitlicher Ablauf	20
5.1.2	Klassifizierung und Akzeptanzkriterien für Risiken	23

5.1.3	Kriterien zur Definition, Abgrenzung und Priorisierung des Analyseumfangs im Produktlebenszyklus	23
5.1.4	Übersicht zur Methodenauswahl	26
5.2	Standardisierte Methodendarstellung	27
5.2.1	ATAM (Architecture Tradeoff Analysis Method)	27
5.2.2	CCA (Common Cause Analysis)	30
5.2.3	CPA (Criticality and Prioritization Analysis)	35
5.2.4	DRBFM (Design Review Based on Failure Mode)	39
5.2.5	ETA (Event Tree Analysis) – Ereignisbaumanalyse	44
5.2.6	FMEA (Fehlermöglichkeits- und -einflussanalyse)	48
5.2.7	FMEA-MSR (FMEA-Monitoring und Systemreaktion)	52
5.2.8	FMEDA (Failure Modes, Effects and Diagnostic Analysis)	56
5.2.9	FTA (Fault Tree Analysis) – Fehlerbaumanalyse	59
5.2.10	HARA (Hazard Analysis and Risk Assessment)	65
5.2.11	HAZOP (Hazard and Operability Study)	68
5.2.12	SOTIF (Safety Of The Intended Functionality – ISO21448)	71
5.2.13	STPA (System-Theoretic Process Analysis)	76
5.2.14	SWIFT (Structured What-If Technique)	79
5.2.15	TARA (Threat Analysis and Risk Assessment)	82
6	Risikoanalyse entlang der Integrationskette	87
6.1	Einleitung	87
6.2	Zielsetzung	87
6.3	Motivation und Nutzen	87
6.4	Herausforderungen	88
6.5	Übernahme von Risikoanalysen	88
6.6	Handlungsempfehlungen für die Praxis	91
7	Annex – Fallbeispiele	92
7.1	Praxisbeispiele für die Anwendung der Methoden	92
7.1.1	ATAM – Spurhalteassistent	92
7.1.2	ETA – OTA-Update	95
7.1.3	CPA – Verkehrszeichenerkennung	98
7.1.4	HAZOP – Türöffner-App	100

7.2	Methodenauswahl in einem Safety-relevanten Projekt	102
8	Annex – Anwendbarkeit der Methoden (Übersicht)	105

1 Einleitung – Motivation und Zielsetzung

Die zunehmende Digitalisierung und Funktionserweiterung moderner Straßenfahrzeuge führen zu einem signifikanten Anstieg softwarebasierter Komponenten. Softwarebasierte Funktionalitäten sind heute integraler Bestandteil sicherheitsrelevanter und qualitätskritischer Fahrzeugsysteme. Dennoch wird ihre Risikoanalyse bislang überwiegend mit Methoden durchgeführt, die primär für Hardware konzipiert wurden.

Insbesondere die FMEA als etablierter Standard im Qualitätsmanagement weist in ihrer aktuellen Form methodische Einschränkungen auf, wenn es um die Bewertung softwareinduzierter Risiken geht. Die Vorgaben der IATF begrenzen die methodische Offenheit zusätzlich. Gleichzeitig zeigt sich in der Praxis, dass bestehende Risikoanalyseprozesse häufig zu langwierig, aufwendig und wenig zielführend sind – z. B. mit der Folge, dass Risiken unzureichend erkannt werden, etwa aufgrund nicht identifizierter Einsatzbedingungen oder Schnittstellen.

Das Hauptziel einer technischen Risikoanalyse besteht darin, potenzielle Probleme frühzeitig zu identifizieren und zu vermeiden, bevor sie zu negativen Konsequenzen führen – etwa Fehler bei der Integration in einer späteren Projektphase, Projektverzögerungen oder Qualitätsmängel im Betrieb. Eine wirksame Risikoanalyse trägt somit wesentlich zur Stabilität und Effizienz von Entwicklungsprozessen bei.

Dieser Band adressiert die Notwendigkeit, Softwarerisiken gezielter, effizienter und kontextgerechter zu analysieren. Ziel ist es, die Komplexität zu reduzieren und eine praxisnahe Orientierung zu bieten: Welche Risikoanalysemethoden sind in welcher Entwicklungsphase sinnvoll? Wie lassen sich softwarebezogene Risiken frühzeitig identifizieren und bewerten?

Ein besonderer Fokus liegt auf der phasenweisen Anwendung geeigneter Methoden, um bereits ab der Konzeptphase einen präventiven Ansatz zu fördern. Die Risikoanalyse soll als unterstützendes Werkzeug verstanden werden – nicht als zusätzliche Belastung, sondern als strukturierte Hilfe zur Sicherstellung funktionaler Qualität.

2 Nutzen, Geltungsbereich und Zielgruppe

Dieser Band beschreibt eine systematische Methodik zur Identifizierung, Analyse, Bewertung und Behandlung von technischen Risiken softwarebasierter Funktionalitäten (vgl. Kapitel 3 Begriffe). Der Fokus liegt auf dem gesamten Produktlebenszyklus. Die hier vorgestellten Methoden dienen als Leitfaden zur Steigerung der Qualität, Zuverlässigkeit und Sicherheit von softwarebasierten Funktionalitäten in Fahrzeugen der Automobilindustrie und deren Ökosystem.

2.1 Nutzen

2.1.1 Ziele und Vorteile der Risikoanalysen

Die Anwendung der in diesem Band beschriebenen Methodik zielt darauf ab, potenzielle technische Risiken proaktiv zu identifizieren und zu beherrschen. Dadurch sollen negative Auswirkungen vermieden und die Produktreife erhöht werden. Die primären Ziele sind die Vermeidung von:

- Fehlern bei der Integration in einer späteren Projektphase und Projektverzögerungen
- Kundenbeanstandungen, Produktrückrufen und kostspieligen Fehlerbehebungsmaßnahmen
- Safety-Vorfällen oder Security-Verletzungen
- Verlust des Kundenvertrauens

2.1.2 Betrachtete technische Risiken

Der Umfang dieser Analyse deckt ein breites Spektrum an softwarebezogenen technischen Risiken ab, die sich auf das Fahrzeug, seine Benutzer:innen oder seine Umgebung auswirken könnten. Dies umfasst unter anderem:

- **funktionale Risiken:** Die Software führt ihre beabsichtigte Funktionalität nicht korrekt aus, was z. B. zu Fehlfunktionen, Leistungsminderung, eingeschränkter User-Experience und Imageschäden führt
- **Safety- & Security-Risiken:** Softwareschwachstellen oder -ausfälle, die zu einem unsicheren Zustand (Safety) führen oder von böswilligen Akteuren ausgenutzt werden könnten (Security)

- **Zuverlässigkeits- & Verfügbarkeitsrisiken:** Die Software fällt zeitweise aus oder ist während des Betriebs nicht verfügbar
- **Performancerisiken:** Die Software erfüllt die Anforderungen an Timing, Speicher oder Rechenleistung nicht, was die Reaktionsfähigkeit und Stabilität des Systems beeinträchtigt
- **Integrations- und Kommunikationsrisiken:** Fehler an den Schnittstellen zwischen verschiedenen Softwarekomponenten, Steuergeräten, Aktuatoren, Sensoren und/oder anderen Systemen sowie Risiken durch fehlerhafte Kommunikation zwischen vernetzten Systemen

2.2 Geltungsbereich

Dieser Band ergänzt bestehende Industriestandards und ersetzt diese nicht. Er stellt eine anwendungsorientierte Methodik bereit, um für spezifische softwarebasierte Funktionalitäten geeignete Risikoanalysemethoden auszuwählen und anzuwenden. Er bezieht sich auf Standards wie z. B. AIAG & VDA FMEA-Handbuch, VDA-Reifegradabsicherung, ISO 26262 (Funktionale Sicherheit), ISO/SAE 21434 (Cybersecurity) und Automotive SPICE®.

2.2.1 Ziele und Vorteile der Risikoanalysen

Im Zentrum der Analyse steht die softwarebasierte Funktionalität. Es wird bewertet, welche Funktionalität die Software erbringen soll (Soll-Verhalten) und welche potenziellen Abweichungen davon auftreten können (Fehlverhalten). Dies gilt für alle Softwareartefakte und -komponenten, die zur Realisierung einer Funktionalität beitragen, unter anderem:

- Apps auf dem Steuergerät
- Middleware
- Betriebssysteme
- Firmware
- Treiber
- Debug-Software Boot Loader (z. B. Flash-BL)
- Konfigurationsdaten
- Nachrichtenkatalog
- Base Application

- KI-Modelle
- Remote Application
(Backend, Handy-App)
- Cloud-Anwendungen
- Legacy-Software
- Free and Open-Source Software (FOSS)
- Commercial Off-The-Shelf Software (COTS)
- Modified Off-The-Shelf Software (MOTS)

2.2.2 Besonderer Fokus: Fahrzeugvernetzung

Fahrzeuge sind hochgradig vernetzte Systeme, die in einem komplexen Ökosystem agieren (s. **Fehler! Verweisquelle konnte nicht gefunden werden.**).

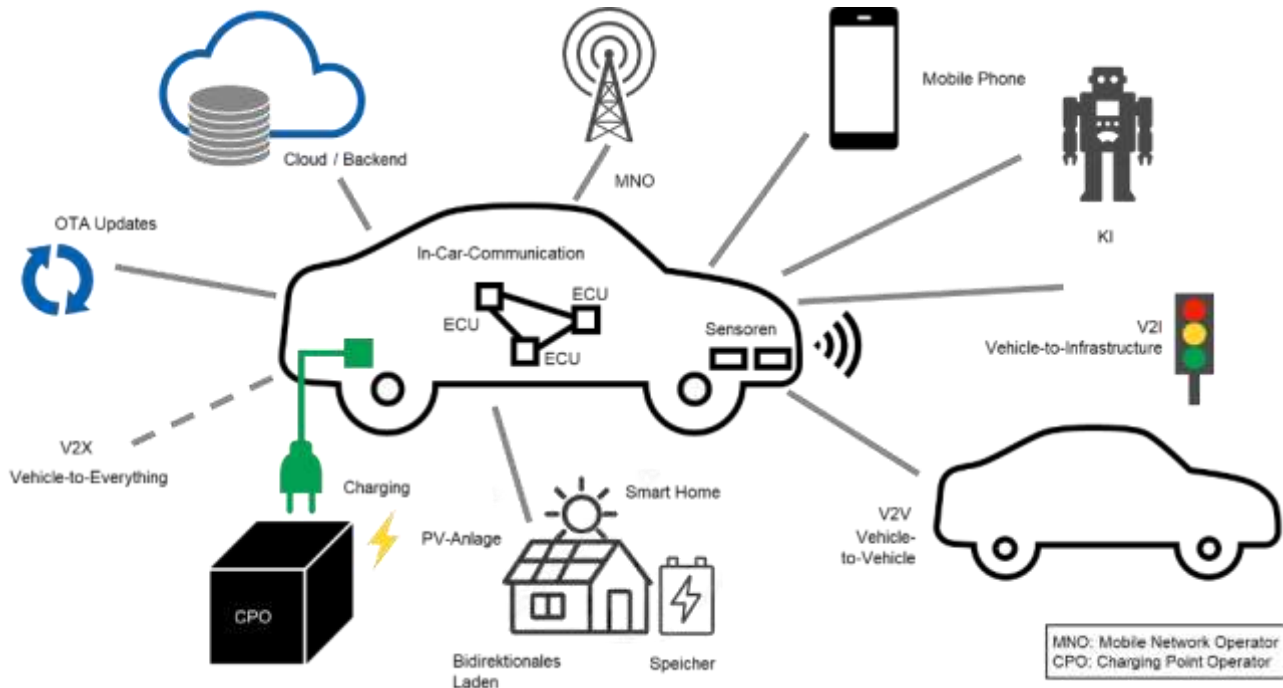


Abbildung 2-1: Fahrzeug-Ökosystem

Die daraus resultierenden Risiken sind ein Schwerpunkt dieses Bandes. Dies betrifft insbesondere:

- **In-Car-Communication:** Risiken in Bezug auf die Kommunikation einzelner Steuergeräte im Fahrzeug
- **V2X (Vehicle-to-Everything) Communication:** Risiken in Bezug auf Datenintegrität, Authentizität und Verfügbarkeit von externen Quellen
- **Cloud- und Backend-Schnittstellen:** Risiken im Zusammenhang mit der Interaktion mit Servern, dem Datenschutz und der Zuverlässigkeit von vernetzten Diensten

Eine zusätzliche Komplexitätsebene entsteht durch die global verteilte Lieferkette. Entwicklungspartner arbeiten parallel auf unterschiedlichen Integrationsstufen an der Software. Eine durchgängige Risikobeherrschung erfordert daher, dass die Ergebnisse der Risikoanalysen über Systemgrenzen und Unternehmensebenen hinweg synchronisiert und nachvollziehbar verknüpft werden können.

Außerhalb des Geltungsbereichs (Out of Scope): Dieser Band fokussiert auf Risiken, die ihren Ursprung in der Software oder dem Systemdesign softwarebasierter Funktionalitäten haben. Explizit nicht im Geltungsbereich (Out of Scope) liegen:

- Risiken, die ausschließlich auf mechanische und elektronische Fehler zurückzuführen sind
- Risiken im Zusammenhang mit Fertigungsprozessen
- rein kommerzielle Projektrisiken (z. B. Budgetüberschreitungen, Marktakzeptanz)
- organisatorische Risiken (z. B. Projektmanagement, Ressourcenplanung), sofern sie sich nicht direkt in einem technischen Produktrisiko manifestieren

2.3 Zielgruppe

Dieser Band richtet sich an alle Fach- und Führungskräfte, die an der Spezifikation, Entwicklung, Integration und Absicherung von Software im Automotive-Bereich beteiligt sind. Dazu zählen insbesondere:

- Softwareentwickler:innen und -architekt:innen
- System- und Sicherheitsingenieur:innen

- Qualitätssicherungs- und Testmanager:innen
- Projekt- und Produktmanager:innen
- technische Einkäufer und Lieferantenqualitätsingenieur:innen

3 Begriffe und Definitionen

3.1 Funktionsarchitektur

Funktionsarchitektur bezeichnet die strukturierte Zerlegung der übergeordneten System- und Softwarefunktionalitäten in einzelne Subsysteme und Komponenten. Dabei werden sowohl das statische Verhalten (z. B. Zustände und Schnittstellen) als auch das dynamische Verhalten (z. B. Abläufe und Interaktionen) definiert und modelliert.

3.2 Hardware (HW)

Hardware umfasst alle physischen Komponenten, die für die Funktion eines Systems erforderlich sind – darunter mechanische, elektronische, hydraulische und sensorische Elemente.

3.3 Induktive versus deduktive Methoden

Induktive Methoden analysieren mögliche Ursachen und deren Auswirkungen, während deduktive Methoden von einem Fehlerbild ausgehen und dessen Ursachen systematisch zurückverfolgen. Beide Ansätze sind Teil der Sicherheitsanalysen gemäß ISO 26262.¹

3.4 Risiko

Risiko bezeichnet die Kombination aus der Wahrscheinlichkeit des Auftretens und den Folgen eines bestimmten zukünftigen unerwünschten Ereignisses.²

¹ ISO 26262-9:2018. International Organization for Standardization (2018). *Road vehicles – Functional safety – Part 9: Automotive safety integrity level (ASIL)-oriented and safety-oriented analyses* (2nd ed., section 8).

² VDA-Online-Glossar, basierend auf: *ISO/IEC/IEEE 24765:2017, Systems and software engineering – Vocabulary*. (2017) (2nd ed.).

3.5 Software (SW)

Software umfasst ausführbare Programme, Verfahren sowie zugehörige Dokumentation und Daten, die sich auf den Betrieb eines rechnergestützten Systems beziehen.³

3.6 Softwarebasierte Funktionalität

Unter einer softwarebasierten Funktionalität wird eine spezifische Fähigkeit oder Verhaltensweise eines Systems verstanden, die durch Software realisiert oder unterstützt wird. Dies kann sowohl im Fahrzeug („In-Vehicle“) als auch in Kombination mit den vernetzten Systemen erfolgen (siehe **Fehler! Verweisquelle konnte nicht gefunden werden.**).

Die Aufgabe der softwarebasierten Funktionalität besteht darin, die vorgesehenen Funktionen und Abläufe mittels Software bereitzustellen, um definierte Anforderungen und Nutzungsziele innerhalb ihres Systemkontexts zu erfüllen.

Die softwarebasierte Funktionalität wird beschrieben durch die Definition von Input, Verarbeitung und Output unter Einhaltung von nicht-funktionalen Anforderungen innerhalb ihres Systemkontexts.

3.7 V2X (Vehicle to Everything)

V2X beschreibt die Vernetzung eines Fahrzeugs mit externen Partnern. Dazu zählen spezifische Ausprägungen wie

- V2I (Vehicle to Infrastructure) für die Kommunikation mit Verkehrsinfrastruktur und
- V2V (Vehicle to Vehicle) für die direkte Vernetzung zwischen Fahrzeugen

³ VDA-Online-Glossar, basierend auf: ISO/IEC/IEEE 24765:2017, *Systems and software engineering – Vocabulary*. (2017) (2nd ed.).

4 Grundprinzipien der Risikoanalyse

Die Grundprinzipien der Risikoanalyse sind zentrale Leitlinien, die sicherstellen, dass Risiken in technischen Systemen, insbesondere in der funktionalen Sicherheit (z. B. nach ISO 26262, ISO 31000, IEC 61508) und der Cybersecurity (z. B. nach ISO/SAE 21434, ISO/IEC 27001), systematisch und nachvollziehbar identifiziert, bewertet und durch geeignete Maßnahmen beherrscht werden. Die Vorgehensweise und Vorbedingungen der Risikoanalyse von Software-/softwarebasierter Funktionalität sind dabei analog zur Risikoanalyse von Hardware. Es ist nicht möglich, alle Risiken zu identifizieren, aber die kritischsten Themen müssen abgedeckt sein.

Für die systematische Identifizierung von Risiken sind verschiedene Vorbedingungen zur Durchführung einer Risikoanalyse zu beachten, die in dem folgenden Abschnitt beschrieben sind. Der darauffolgende Abschnitt behandelt den systematischen Ansatz zur Risikoidentifizierung.

4.1.1 Vorbedingungen zur Durchführung einer Risikoanalyse

Zu den Vorbedingungen einer Risikoanalyse zählen u. a.

- die Beschreibung der Softwarefunktionalitäten, welche z. B. aus einer Systemanalyse resultiert
- die Beschreibung von nicht-funktionalen Anforderungen, z. B. Sicherheitsrelevanz der Softwarefunktionalität, Performance, Zuverlässigkeit, Toleranzen, Signal-Qualität
- die Zuweisung der Softwarefunktionalitäten auf Elemente einer System-/Softwarearchitektur

4.1.2 Systematischer Ansatz zur Identifizierung der Risiken

Der systematische Ansatz ist in die folgenden vier Schritte gegliedert:

1. Systematisches Vorgehen

- Keine Ad-hoc-Analysen, sondern nachvollziehbarer Prozess
- Unabhängig von der jeweiligen Architektur-Ebene

2. Berücksichtigung des Systemkontexts

- Inhalt definieren

- Berücksichtigung des Systemkontextes (Risikoanalyse bezieht Systemgrenzen, Umgebungsbedingungen, Nutzerverhalten und Schnittstellen mit ein)
- Berücksichtigung von relevanten Risiken aus vorgelagerten Risikoanalysen (z. B. HARA, TARA, D-FMEA System)

3. Identifikation und Abschätzung potenzieller Risiken

- Auswahl der geeigneten Methode zur Risikoanalyse (s. Kapitel 5)

4. Definition und Umsetzung von geeigneten Maßnahmen

- Geeignete Maßnahmen (s. ISO 21434:2021 RQ-15-17) können sein:
 - Risikovermeidung
(z. B. durch die Rücknahme von Funktionalitäten)
 - Risikoreduzierung
(z. B. Implementierung von technischen Maßnahmen zur Risikoreduzierung)
 - Teilen des Risikos
(z. B. durch Versicherungsgesellschaften)
 - Akzeptanz des Risikos

Sollte eine Methode eine oder mehrere Anforderungen aus den oben genannten vier Schritten nicht erfüllen, ist eine definierte ergänzende Systematik zur Kompensation anzuwenden.

4.2 Systemzerlegung und Interaktion an den Schnittstellen

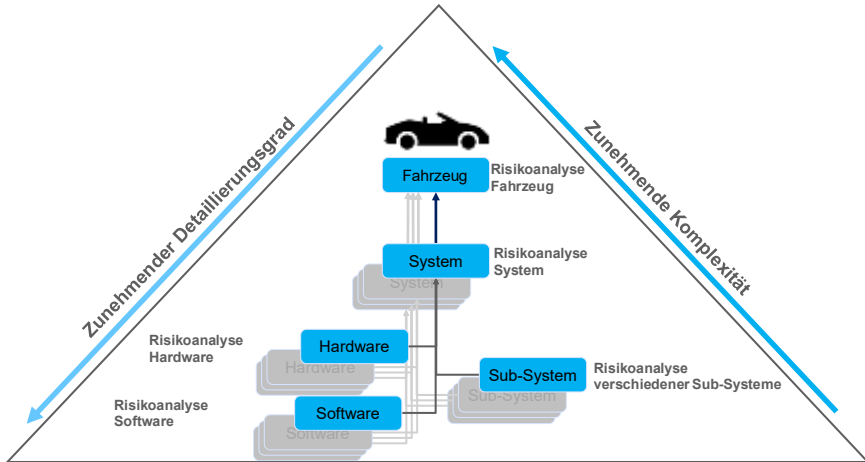


Abbildung 4-1: Prinzip-Darstellung von System und Subsystemen mit Schnittstellen innerhalb eines Fahrzeuges der Automobilindustrie

Die Abbildung veranschaulicht die Zerlegung eines Fahrzeuges der Automobilindustrie in Systeme und seine Subsysteme. Zwischen den Systemen und Subsystemen sind Schnittstellen für die gegenseitige Interaktion definiert. Risikoanalysen finden auf allen Ebenen statt.

Die Software ist Bestandteil eines (Sub-)Systems, welches eine Schnittstelle zur Hardware hat, auf deren Risikoanalyse in diesem Band eingegangen wird.

5 Leitfaden zur Methodenauswahl

5.1 Anleitung zur Auswahl von Risikoanalyse-Methoden

Dieses Kapitel bietet einen strukturierten Leitfaden zur Auswahl geeigneter Risikoanalysemethoden für Softwarefunktionen in softwarebasierten Fahrzeugsystemen. Die gezielte Auswahl passender Methoden ist von zentraler Bedeutung, um die Analyse effizient, relevant und nachvollziehbar zu gestalten. Sie ermöglicht eine gezielte Identifikation, Bewertung und Beherrschung von Risiken und trägt dazu bei, effiziente Analysepfade zu finden. Durch eine methodisch fundierte Herangehensweise wird sichergestellt, dass der Prozess der Risikoanalyse systematisch und anwendungsbezogen auf die spezifischen Anforderungen softwarebasierter Fahrzeugfunktionen erfolgt.

5.1.1 Phasendefinition und zeitlicher Ablauf

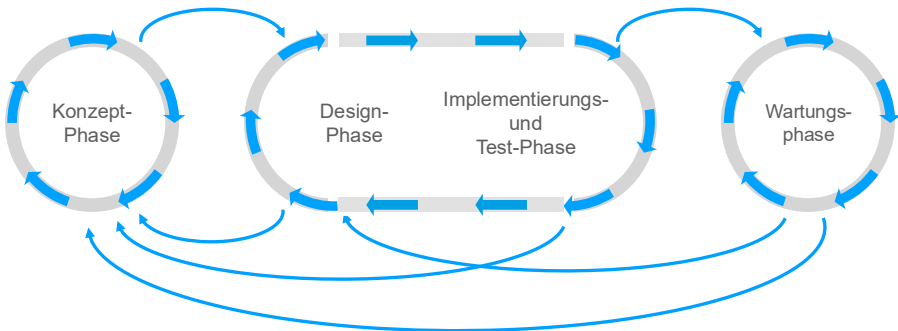


Abbildung 5-1: Phasenmodell

Die dargestellte Abbildung zeigt vier Hauptphasen: die Konzept-, Design-, Implementierungs- und Testphase sowie die Wartungsphase. In jeder dieser Phasen sollten spezifische Aktivitäten zur Risikoidentifizierung durchgeführt werden.

In der Konzeptphase erfolgt eine erste Risikoidentifikation, die sich auf grundlegende Funktionskonzepte stützt. Darauf folgt die Designphase, in der eine detaillierte Analyse basierend auf der Systemarchitektur durchgeführt wird.

Die Implementierungsphase beinhaltet Analyse und Bewertung von Risiken aus der konkreten Umsetzung. Weiters werden hier Risiken aus der Integration von mehreren Teilsystemen betrachtet.

In der Wartungsphase ist der Fokus der Risikoanalyse, dass die Integrität der laufenden Systeme nicht gefährdet wird.

Ein zentrales Element innerhalb dieses Prozesses ist die Iteration: Erkenntnisse aus späteren Phasen fließen kontinuierlich zurück in frühere Entwicklungsstufen, wodurch eine fortlaufende Verbesserung und Anpassung der Risikoanalyse ermöglicht wird. Diese iterative Vorgehensweise unterstützt eine robuste und adaptive Produktentwicklung über den gesamten Lebenszyklus hinweg.

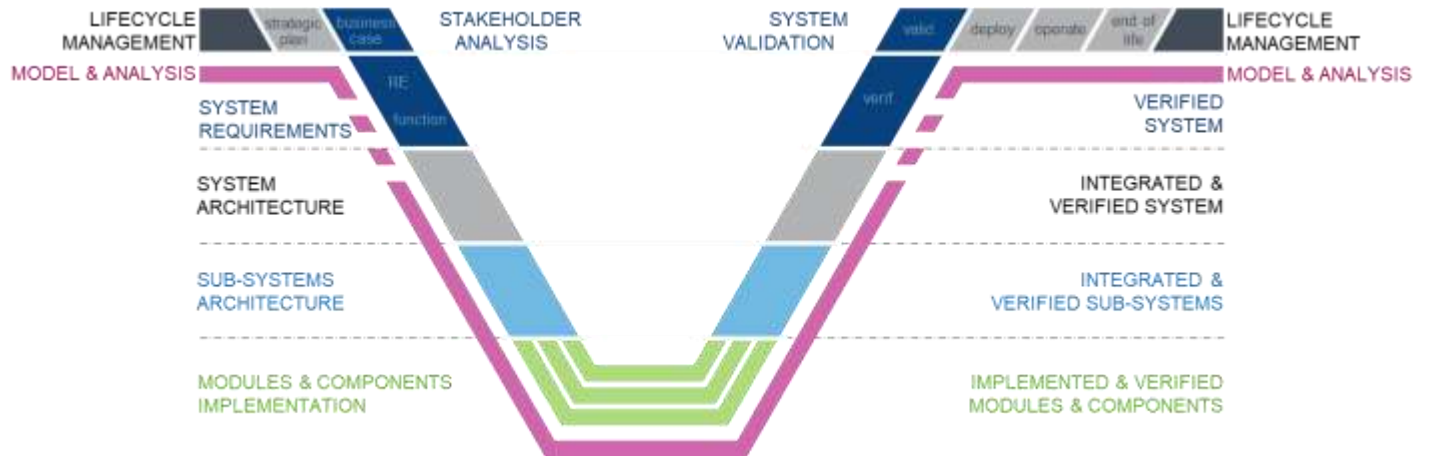


Abbildung 5-2: Übersicht System-Integrationskette

5.1.2 Klassifizierung und Akzeptanzkriterien für Risiken

Eine Klassifizierung, basierend auf der Analyse des Projektumfangs und den funktionalen sowie technischen Anforderungen, bestimmt maßgeblich die Auswahl und Anwendung von Risikoanalysemethoden. Sie fördert eine gezielte Konzentration auf sicherheits- und funktionskritische Systembereiche und optimiert die Ressourcenverwendung im Analyseprozess.

Projektspezifische Akzeptanzkriterien ergänzen definierte Schwellenwerte, indem sie festlegen, welche Risikolevel (z. B. Schweregrad und Auftretenswahrscheinlichkeit) im jeweiligen Anwendungskontext als akzeptabel gelten. Risiken, die diese Schwellenwerte überschreiten, erfordern entsprechende Maßnahmen zur Risikobeherrschung. Die Kriterien werden in enger Abstimmung mit den interessierten Parteien festgelegt und bilden die Grundlage für konsistente und nachvollziehbare Risikoentscheidungen.

5.1.3 Kriterien zur Definition, Abgrenzung und Priorisierung des Analyseumfangs im Produktlebenszyklus

Die Auswahl der geeigneten Risikoanalysemethoden ist maßgeblich vom aktuellen Stand der Produktentwicklung sowie von den spezifischen Eigenschaften des zu analysierenden Softwareanteils abhängig. Entlang des Produktlebenszyklus sind daher unterschiedliche Kriterien zu berücksichtigen, um den Analyseumfang zu definieren, abzugrenzen und zu priorisieren.

5.1.3.1 Konzeptphase

In der Konzeptphase liegt der Fokus auf der systemweiten Betrachtung sowie der Identifikation potenzieller Risikobereiche.

- **Systemarchitektur-Überblick:** Erste grobe Modellierung der Systemarchitektur inklusive Hauptkomponenten und deren Zusammenwirken (HW, SW, Kommunikation), um potenzielle Schwachstellen oder kritische Abhängigkeiten zu identifizieren
 - Systemkontext (Boundary-Analyse): Abgrenzung des zu analysierenden Systems einschließlich seiner Schnittstellen zu Software, Hardware, Umgebung und Nutzer:innen
- **Anforderungsanalyse:** Erfassung und Analyse funktionaler sowie nicht-funktionaler Anforderungen, einschließlich Input-/Output-Spezifikationen der Softwarefunktionalität z. B.:
 - regulatorische Anforderungen und Normen
 - Kundenanforderungen
 - Schnittstellenanforderungen

- Integration in der Lieferkette
- Safety
- (Cyber-)Security
- Datenschutzanforderungen
- Patentrechte und Lizenzen
- **Neuheitsgradbewertung:** Bewertung, ob es sich um eine Neuentwicklung, eine signifikante Änderung oder ein etabliertes Design handelt

5.1.3.2 Designphase

Diese Phase konzentriert sich auf die robuste Auslegung von Architekturen, Schnittstellen und Abhängigkeiten sowie Sicherheits- und Cyber-Security-Aspekten, auch bei Systemen mit maschinellem Lernen.

- **Detaillierte Architektur- und Modulbewertung:** Tiefgehende Analyse der einzelnen Softwarekomponenten und deren Funktionalität inklusive Kommunikation, Zustandsmanagement und Fehlerbehandlung
- **Schnittstellenanalyse:** Untersuchung softwareseitiger Abhängigkeiten und deren Interaktionen mit anderen Systemkomponenten (SW ↔ SW / SW ↔ HW)
- **Abhängigkeiten von Drittsystemen und Drittsoftware:** Prüfung der Risiken durch eingebundene Softwarebibliotheken, Middleware oder externe Services, die Sicherheits- und Stabilitätsanforderungen erfüllen müssen
- **Sicherheitsmechanismen und Absicherungsstrategien:** Evaluation implementierter Sicherheitsfunktionen wie Watchdogs, Redundanzen, Fehlererkennung und -management sowie deren Wirksamkeit im Risikokontext
- **Security:** Bewertung und Beherrschung der Cybersecurity-Risiken

5.1.3.3 Implementierungs- und Testphase

In dieser Phase liegt der Schwerpunkt auf der Realisierung und Prüfung von Anforderungen.

- **Implementierung, Integration, Verifizierung und Validierung von Software- und Hardwareanforderung:** Sicherstellung der kohärenten Umsetzung von Anforderungen sowohl auf Software- als

auch auf Hardwareebene, z. B. Absicherung von Steuergeräten und Buskommunikation

- **Validierung von Risikovermeidungsmaßnahmen:** Durchführung von Tests und Reviews zur Überprüfung, ob geplante und implementierte Maßnahmen die Risiken adäquat absichern

5.1.3.4 Wartungsphase

In dieser Phase liegt der Schwerpunkt auf der Überwachung und dem Management von Risiken im realen Betrieb.

- Berücksichtigung der realen Umgebungsbedingungen, unter denen die Softwarefunktion betrieben wird
 - Software-Update over the air (OTA)
 - V2X, V2I, V2V
- Prävention von Problemen, die sich aus der Änderung des Produkts im laufenden Betrieb ergeben könnten
- Überprüfung der technischen Risikoanalyse im Zusammenhang mit Feldproblemen
- Lessons Learned aus der Feldbeobachtung als Input für neue Risikoanalysen

5.1.4 Übersicht zur Methodenauswahl

Die nachfolgende Tabelle bietet eine Orientierung, welche Risikoanalysemethoden für entsprechende Phasen des Produktlebenszyklus besonders relevant sind. Sie dient als Entscheidungshilfe und betont, dass eine Kombination von Methoden oft den optimalen Ansatz darstellt.

Tabelle 5-1: Risikoanalysemethoden bezogen auf Phasen

Risikoanalysemethoden	Konzeptphase	Designphase	Implementierungs- und Testphase	Wartungsphase	Kapitel
ATAM – Architecture Tradeoff Analysis Method	++++	+++			5.2.1
CCA – Common Cause Analysis		+	+		5.2.2
CPA – Criticality and Priority Assessment	++++	+++	++	+	5.2.3
DRBFM – Design Review Based on Failure Mode		++++	++	+++	5.2.4
ETA – Event Tree Analysis		++		+++	5.2.5
FMEA – Failure Mode and Effects Analysis	+++	+++	++	+	5.2.6
FMEA-MSR – FMEA-Monitoring und Systemreaktion	+	+++	+	+	5.2.7
FMEDA – Failure Modes, Effects and Diagnostic Analysis		+	+	+	5.2.8
FTA – Fault Tree Analysis	+	+++	++	+++	5.2.9
HARA – Hazard Analysis and Risk Assessment	++++	+	+		5.2.10
HAZOP – Hazard and Operability Study	+	++++	+++		5.2.11
SOTIF – Safety Of The Intended Functionality ^{*)}	++++	+++	+	+	5.2.12
SWIFT – Structured What-If Technique	++	++++	+++		5.2.13
STPA – System-Theoretic Process Analysis	++++	++			5.2.14
TARA – Threat Analysis and Risk Assessment	++++	+++	++	+	5.2.15

^{*)} Beschränkt auf die Risikoanalysemethode aus dem Standard.

- **++++** (sehr hoch): Die Methode ist in dieser Phase sehr stark anwendbar und bietet einen überragenden Nutzen. Sie ist oft fundamental oder eine Best Practice
- **+++** (hoch): Die Methode ist in dieser Phase stark anwendbar und liefert signifikante Ergebnisse. Ihr Einsatz wird dringend empfohlen
- **++** (mittel): Die Methode ist in dieser Phase moderat anwendbar und kann nützliche Erkenntnisse liefern. Der Einsatz ist situationsabhängig sinnvoll
- **+** (niedrig): Die Methode ist in dieser Phase wenig anwendbar und bietet nur begrenzten Nutzen. Der Einsatz sollte sorgfältig geprüft werden
- (Leeres Feld): Die Methode ist in dieser Phase nicht oder kaum anwendbar

5.2 Standardisierte Methodendarstellung

5.2.1 ATAM (Architecture Tradeoff Analysis Method)

5.2.1.1 Ziel der Methode

Die systematische Bewertung der Auswirkungen von Architekturentscheidungen auf die Qualitätsattribute eines Softwaresystems dient der Identifikation von Risiken, Sensitivitätspunkten und möglichen Trade-offs zwischen konkurrierenden Qualitätszielen. Sie unterstützt fundierte Entscheidungen im Architekturprozess und schafft Transparenz hinsichtlich der relevanten Einflussfaktoren.

5.2.1.2 Grundsätzliches Vorgehen

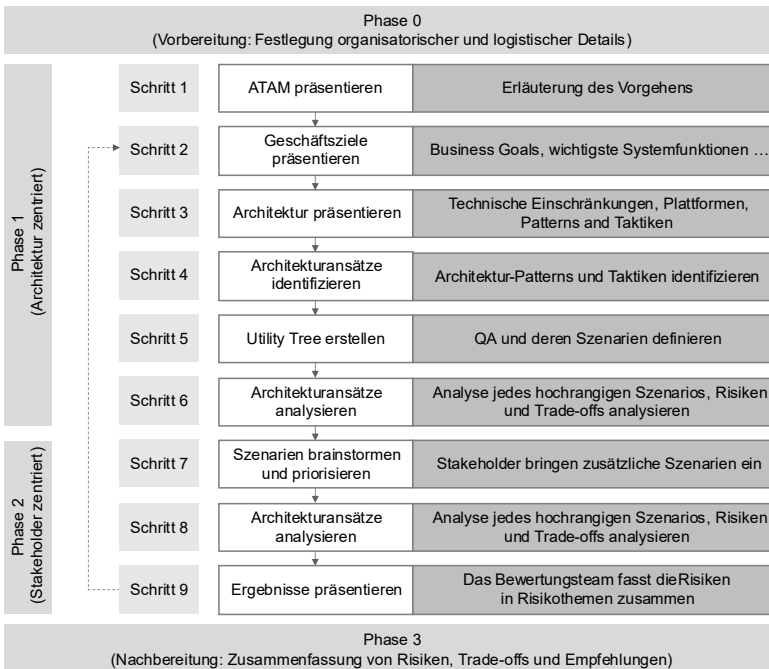


Abbildung 5-3: Phasen der ATAM-Methode

Die ATAM involviert drei Teams: das Evaluierungsteam, das die Methode moderiert, Workshops leitet und die Architektur bewertet; das Kernteam, bestehend aus Architekt:innen, Projektleiter:innen und weiteren Entscheidungsträger:innen, die fachliche und geschäftliche Einblicke liefern; sowie

die Architektur-Stakeholder, die beispielsweise aus Entwickler:innen, Tester:innen und Vertreter:innen aus dem Management bestehen und unterschiedliche Perspektiven und Anforderungen einbringen.

Die ATAM umfasst vier Phasen, die in Abbildung 5-3 dargestellt sind. In der Vorbereitung (Phase 0) werden organisatorische Rahmenbedingungen geklärt, die Teilnehmer:innen festgelegt und die Architekturunterlagen geprüft. Phasen 1 und 2 bilden den Kern der Methode. In Phase 1 trifft sich das Evaluierungsteam mit dem Kernteam, um die Methode vorzustellen, Geschäftsziele und Architektur zu präsentieren, zentrale Architekturansätze zu identifizieren und einen sog. Quality-Attribute Utility Tree (s. Fallbeispiel im Annex, Kapitel 7.1.1) zu erstellen, der Qualitätsanforderungen in priorisierten Szenarien strukturiert. Die Architektur wird dann anhand dieser Szenarien detailliert analysiert, wobei Risiken, Kompromisse und Stärken herausgearbeitet werden. In Phase 2 wird der Teilnehmerkreis durch weitere Stakeholder erweitert, die zusätzliche Qualitätsszenarien erarbeiten, priorisieren und gemeinsam mit dem Evaluierungsteam die Architektur vertieft bewerten. Dieses Vorgehen gewährleistet eine umfassende, szenarienbasierte Bewertung der Softwarearchitektur. Abschließend wird in Phase 3 ein Abschlussbericht erstellt, der Risiken, Entscheidungen, Kompromisse und Empfehlungen zusammenfasst.

5.2.1.3 Hinweise/Besonderheiten/Unterschiede beim Einsatz im Softwarekontext

ATAM eignet sich für komplexe Systeme mit hohen Qualitätsanforderungen. Der Fokus liegt auf der Analyse von Qualitätsattributen und ihren Abhängigkeiten anhand szenarienbasierter Bewertungen.

5.2.1.4 Voraussetzungen (Input)

Voraussetzung für die Durchführung der technischen Risikoanalyse ist eine dokumentierte Softwarearchitektur, die nachvollziehbare Festlegung von Architekturentscheidungen sowie die aktive Einbindung der Stakeholder.

5.2.1.5 Anwendungstiefe und Beendigungskriterien

Die Analyse erfolgt szenarienbasiert und orientiert sich an der Komplexität der Softwarearchitektur. Sie gilt als abgeschlossen, wenn die priorisierten Qualitätsattribute bewertet, zentrale Risiken und Trade-offs identifiziert sowie Konsens zwischen den Stakeholdern erreicht wurde.

5.2.1.6 Anforderungen an Aufwand, Team und Tools

Die Durchführung der ATAM-Methode ist mit hohem Aufwand verbunden, da mehrere Phasen mit Workshops, Szenarienentwicklung und Architekturbewertungen durchgeführt werden.

Für die Analyse wird ein interdisziplinäres Team mit spezifischem Wissen benötigt:

- **Evaluierungsteam:** methodische Expertise in Architekturbewertungen, Workshop-Moderation und Dokumentation
- **Kernteam:** bringt fundierte Kenntnisse in Softwarearchitektur, Projektmanagement sowie fachliche und geschäftliche Zusammenhänge ein und verfügt über Erfahrung in der Bewertung von Qualitätsattributen
- **Architektur-Stakeholder:** bringen unterschiedliche Perspektiven und Fachkenntnisse ein, z. B. technisches Wissen über die Implementierung, Kenntnisse zur Sicherstellung der Softwarequalität, Einblicke in Betrieb und Wartung sowie Verständnis für geschäftliche und organisatorische Anforderungen

Unterstützende Tools zur Dokumentation und Visualisierung sind hilfreich, aber nicht zwingend.

5.2.1.7 Nutzen im Softwarekontext (Vorteile)

ATAM ermöglicht die frühzeitige Erkennung architektonischer Risiken, unterstützt die Transparenz über Qualitätsanforderungen und fördert interdisziplinäre Kommunikation. Die Methode ist präventiv nutzbar und liefert eine nachvollziehbare Entscheidungsbasis.

5.2.1.8 Einschränkungen der Methode im Softwarekontext (Nachteile)

Keine quantitative Bewertung oder Maßnahmenableitung. Ergebnisse sind qualitativ und abhängig von der Expertise der Beteiligten. Die Methode ist zeitintensiv und setzt eine hohe Stakeholder-Beteiligung voraus.

5.2.1.9 Ergebnis (Output)

Dokumentierte Risiken, Sensitivitätspunkte, identifizierte Trade-offs sowie bewertete Qualitätsattribut-Szenarien sind zentrale Ergebnisse der Analyse. Sie bilden die Grundlage für die anschließende Architekturverfeinerung und für weiterführende technische Bewertungen.

5.2.1.10 Referenzen/Weiterführende Literatur

Kazman, R., Klein, M., & Clements, P. (2000). *ATAM: Method for architecture evaluation*. SEI Technical Report CMU/SEI-2000-TR-004.

Bass, L., Clements, P., & Kazman, R. (2022). *Software architecture in practice* (4th ed.). Addison-Wesley.

5.2.2 CCA (Common Cause Analysis)

5.2.2.1 Ziel der Methode

Die Methode dient der Identifizierung und Bewertung von Risiken, die durch gemeinsame Ursachen entstehen und mehrere ansonsten unabhängige Softwarefunktionen gleichzeitig beeinträchtigen können. Sie findet Anwendung in der funktionalen Sicherheit, der Cybersecurity und der Bewertung der Zuverlässigkeit des Fahrzeugs.

5.2.2.2 Grundsätzliches Vorgehen

Das grundsätzliche Vorgehen bei der Common Cause Analysis (CCA) im Softwarekontext umfasst mehrere Schritte, um systemische Schwachstellen aufzudecken.

1. **Beschreibung der Softwarefunktionalitäten und -systeme**

Zu Beginn werden die zu analysierenden Softwarefunktionalitäten und -systeme detailliert erfasst. Ziel ist es, deren Abhängigkeiten, Schnittstellen und Zusammenspiel zu verstehen. Dazu gehört die vollständige Erhebung aller Systemelemente und deren Verknüpfung in einer strukturierten Darstellung

2. **Systematische Ursachenanalyse**

Im nächsten Schritt wird gezielt nach gemeinsamen Ursachen gesucht, die mehrere, eigentlich unabhängige Softwarefunktionalitäten gleichzeitig beeinflussen können. Diese Analyse kann auf unterschiedlichen Architekturebenen erfolgen:

- E/E-Architektur
- Softwarearchitektur
- Entwicklungsprozess
- Tool-Ebene

3. **Durchspielen konkreter Ausfallszenarien**

Typische Szenarien werden simuliert, um potenzielle gemeinsame Ursachen zu identifizieren. Beispiele:

- Was passiert, wenn die zentrale Stromversorgung ausfällt?
- Was passiert, wenn ein Kommunikationsbus gestört ist?
- Was passiert, wenn ein gemeinsam genutzter Sensor falsche Daten liefert?

4. **Bewertung der Auswirkungen**

Die möglichen Folgen eines gemeinsamen Ausfalls werden hinsichtlich verschiedener Aspekte bewertet:

- funktionale Sicherheit (ASIL)
- Funktionalität
- Performance
- Komfort
- Cybersecurity

5. **Analyse vorhandener Schutzmechanismen**

Bestehende Redundanzen, Diversitäten, Überwachungs- und Trennungsmechanismen werden untersucht. Ziel ist es, deren Wirksamkeit gegenüber den identifizierten gemeinsamen Ursachen zu beurteilen.

6. **Ableitung von Maßnahmen**

Falls die Absicherung nicht ausreicht, werden konkrete Vorschläge zur Minderung oder Beseitigung der Risiken erarbeitet. Mögliche Maßnahmen sind:

- Designänderungen
- Implementierung zusätzlicher Diversität
- Verbesserung von Teststrategien
- Anpassung von Entwicklungsprozessen
- strengere Qualifizierung von COTS-Software

5.2.2.3 Hinweise/Besonderheiten/Unterschiede beim Einsatz im Softwarekontext

Softwarefunktionalitäten nutzen oft gemeinsame Ressourcen (z. B. Betriebssysteme, Bibliotheken, Hardwarekomponenten), Entwicklungsprozesse oder Kommunikationspfade. Dies kann zu systemischen Fehlern führen, die bei isolierter Betrachtung einzelner Funktionalitäten übersehen werden. Im Gegensatz zu Hardwarefehlern, die oft zufällig sind, können Softwarefehler durch gemeinsame Ursachen zu weitreichenden und gleichzeitig auftretenden Ausfällen führen. Die Methode fokussiert auf logische Abhängigkeiten und systemische Schwachstellen, die über reine physikalische Ausfälle hinausgehen. Der szenariobasierte Ansatz ist hierbei entscheidend, um die komplexen Interaktionen und potenziellen gemeinsamen Ausfallmodi in Softwaresystemen zu identifizieren.

5.2.2.4 Voraussetzungen (Input)

E/E-Architektur des Fahrzeugs: detaillierte Pläne der Elektronik/Elektrik-Architektur, Vernetzung (CAN, FlexRay, Ethernet), Steuergeräte (ECUs), Sensoren, Aktuatoren und deren Interaktionen.

Softwarearchitektur und -design: genaue Beschreibungen der Softwaremodule, Komponenten, Schnittstellen, Datenflüsse und Abhängigkeiten innerhalb und zwischen den ECUs.

Funktionale Spezifikationen: präzise Definitionen der einzelnen Softwarefunktionen (z. B. Bremssteuerung, Lenkunterstützung, Infotainment, Batteriemangement) und ihrer Leistungsmerkmale sowie der Sicherheitsziele.

Risikoanalysen (vorhandene): Ergebnisse aus bereits durchgeführten Analysen wie HARA, FMEA, FTA, TARA.

Sicherheits- und Cybersecurity-Konzepte: Informationen über implementierte Sicherheitsmechanismen (z. B. Redundanzen, Diversitäten, Überwachungsfunktionen, Fail-Operational-Strategien) und Cybersecurity-Maßnahmen (z. B. Verschlüsselung, Authentifizierung, Secure Boot).

Umgebungsbedingungen und Betriebsmodi: Kenntnisse über die verschiedenen Betriebsmodi des Fahrzeugs (z. B. Zündung ein/aus, Fahren, Parken), Umwelteinflüsse (Temperatur, Vibration, EMV) und deren Auswirkungen auf die Software.

Standardkomponenten und Bibliotheken: Details zur Verwendung von COTS-Software, Open-Source-Bibliotheken, AUTOSAR-Basissoftware oder

wiederverwendeten Softwaremodulen, die in verschiedenen Funktionen zum Einsatz kommen.

Hardwaredetails: Informationen über die zugrunde liegende Hardware (Prozessoren, Speicher, ASICs), da Hardwarefehler gemeinsame Ursachen für Softwarefehler sein können.

5.2.2.5 Anwendungstiefe und Beendigungskriterien

Fahrzeug-Gesamtsystemebene: Identifizierung kritischer Softwarefunktionen oder -systeme (z. B. Antrieb, Lenkung, Bremsen, ADAS), die durch gemeinsame Ursachen ausfallen könnten.

E/E-Architekturebene: Analyse gemeinsamer Hardwareressourcen (z. B. Stromversorgung, Kommunikationsbusse wie CAN/Ethernet, gemeinsame Sensoren, zentrale Recheneinheiten), deren Ausfall mehrere Softwarefunktionen beeinträchtigen könnte.

Softwarearchitekturebene: tiefere Betrachtung von gemeinsam genutzten Softwaremodulen, Bibliotheken, Betriebssystemdiensten, Middleware, Datenstrukturen oder Algorithmen, die von mehreren Funktionen verwendet werden.

Entwicklungsprozess- und Tool-Ebene: Untersuchung potenzieller gemeinsamer Ursachen, die aus dem Entwicklungsprozess selbst stammen (z. B. Fehler im Anforderungsmanagement, in den verwendeten Compilern, Modellierungs-Tools oder Codegeneratoren).

5.2.2.6 Anforderungen an Aufwand, Team und Tools

Eine gründliche CCA kann sehr zeit- und ressourcenintensiv sein, insbesondere bei großen und komplexen Fahrzeugsystemen. Die Identifizierung aller potenziellen gemeinsamen Ursachen in hochgradig vernetzten Softwaresystemen ist aufwendig und erfordert tiefes Systemverständnis. Ein interdisziplinäres Team ist essenziell. Es sollte tiefgehendes Wissen über die gesamte E/E-Architektur, Softwarearchitektur, das Systemdesign, die verwendeten Technologien und den Entwicklungsprozess umfassen. Dazu gehören Softwarearchitekt:innen, Entwickler:innen, Safety- und Security-Expert:innen sowie Domänenexpert:innen. Es gibt weniger spezialisierte Tools für CCA im Softwarebereich im Vergleich zu Hardware. Oft muss man auf generische Modellierungs- oder Analysewerkzeuge zurückgreifen.

Tools: Es gibt weniger spezialisierte Tools für CCA im Softwarebereich im Vergleich zu Hardware. Oft muss man auf generische Modellierungs- oder Analysewerkzeuge zurückgreifen.

5.2.2.7 Nutzen im Softwarekontext (Vorteile)

Identifizierung verdeckter Risiken: deckt Schwachstellen auf, die bei isolierter Betrachtung einzelner Funktionen oder durch andere Analysemethoden (die sich oft auf einzelne Fehler konzentrieren) übersehen werden könnten.

Erhöhung der Fahrzeugsicherheit: führt zu einem besseren Verständnis der Systemresilienz gegenüber systemischen Ausfällen, die zu gefährlichen Situationen führen könnten.

Optimierung von Redundanzen und Diversitäten: hilft zu beurteilen, ob implementierte Redundanzen oder Diversitäten tatsächlich wirksam sind oder ob sie durch eine gemeinsame Ursache umgangen werden können.

Verbesserung der E/E- und Softwarearchitektur: liefert wertvolle Erkenntnisse für die Gestaltung einer robusteren, sichereren und zuverlässigeren Fahrzeugarchitektur.

Effizientere Test- und Validierungsstrategien: leitet die Entwicklung von Testfällen an, die auf gemeinsame Ursachen abzielen, die sonst schwer zu testen wären (z. B. Injektion von Fehlern in gemeinsame Ressourcen).

Kostenreduktion langfristig: Durch die frühzeitige Erkennung und Behebung von Common-Cause-Fehlern werden teure Rückrufe, Feldaktionen oder Gewährleistungsfälle vermieden.

Erfüllung von Normen: unterstützt die Einhaltung relevanter Normen wie ISO 26262 (Funktionale Sicherheit), die Common Cause Failures explizit adressiert, und ISO 21434 (Cybersecurity), wo gemeinsame Schwachstellen ausgenutzt werden können.

5.2.2.8 Einschränkungen der Methode im Softwarekontext (Nachteile)

Hohe Komplexität: Moderne Fahrzeugsoftware ist komplex und stark vernetzt. Die Identifizierung aller potenziellen gemeinsamen Ursachen kann sehr aufwendig sein.

Abstraktionsgrad: Die Wahl des richtigen Abstraktionsgrades ist schwierig. Zu detailliert wird es unüberschaubar, zu grob werden kritische gemeinsame Ursachen übersehen.

Subjektivität: Die Identifizierung potenzieller gemeinsamer Ursachen kann eine gewisse Subjektivität beinhalten, wenn nicht alle Abhängigkeiten explizit dokumentiert sind.

Dynamisches Verhalten: Softwareverhalten ist oft dynamisch und zustandsabhängig. Statische Analysen können möglicherweise nicht alle dynamisch auftretenden gemeinsamen Ursachen erfassen.

5.2.2.9 Ergebnis (Output)

Die Analyse liefert eine strukturierte Übersicht über identifizierte Common Cause Failures (CCF), also Szenarien, in denen mehrere Softwarefunktionen oder -systeme durch eine gemeinsame Ursache gleichzeitig beeinträchtigt werden können. Dazu gehören detaillierte Beschreibungen der gemeinsamen Ursachen, betroffener Funktionen und Systeme sowie der potenziellen Auswirkungen auf Fahrzeugsicherheit (ASIL), Funktionalität, Performance, Komfort und Cybersecurity. Bestehende Schutzmaßnahmen werden hinsichtlich ihrer Wirksamkeit bewertet. Auf Basis dieser Bewertung werden konkrete Empfehlungen zur Risikominderung abgeleitet. Abschließend erfolgt eine qualitative oder quantitative Einschätzung des verbleibenden Restrisikos sowie eine vollständige Dokumentation des Analyseprozesses.

5.2.2.10 Referenzen/Weiterführende Literatur

International Organization for Standardization. (2018). *ISO 26262:2018 – Road vehicles – Functional safety*. ISO.

International Organization for Standardization, & SAE International. (2021). *ISO/SAE 21434:2021 – Road vehicles – Cybersecurity engineering*. ISO.

5.2.3 CPA (Criticality and Prioritization Analysis)

5.2.3.1 Ziel der Methode

Die Criticality and Prioritization Analysis (CPA) ist eine strukturierte Methode zur Bewertung von Softwarefunktionalitäten hinsichtlich ihrer Kritikalität im Gesamtsystem. Sie dient dazu, frühzeitig sicherheitsrelevante, robuste und cyberresiliente Funktionalitäten zu identifizieren und deren Absicherungsbedarf zu priorisieren. CPA wird insbesondere in der Konzept- und Architekturphase eingesetzt, um Risiken zu erkennen, Ressourcen gezielt zu planen und die Grundlage für weiterführende Analysen (z. B. FMEA, Threat Modeling) zu schaffen.

5.2.3.2 Grundsätzliches Vorgehen

1. Definition des Analyseumfangs

- Festlegung, welche Softwarefunktionen, Komponenten oder Kommunikationspfade betrachtet werden sollen
- Abgrenzung des Systems oder Subsystems (z. B. ADAS-Funktion, Steuergerät, Softwaremodul)

2. Identifikation der Funktionen und ihrer Systemrolle

- Beschreibung der betrachteten Softwarefunktionen und ihrer Aufgaben im Gesamtsystem
- Ermittlung der funktionalen Abhängigkeiten zu anderen Komponenten (z. B. Sensoren, Aktuatoren, Steuergeräte)

3. Bewertung der Kritikalität

- Einschätzung, wie kritisch eine Funktion im Hinblick auf folgende Aspekte ist:
 - funktionale Sicherheit: Führt ein Fehler zu einem sicherheitskritischen Zustand?
 - Betriebsrelevanz: Ist die Funktion für den normalen Fahrbetrieb essenziell?
 - Cybersecurity: Könnte ein Angriff auf diese Funktion sicherheitsrelevante Folgen haben?
 - Systemverfügbarkeit: Beeinflusst ein Ausfall die Gesamtverfügbarkeit des Fahrzeugs?

Typischerweise erfolgt die Bewertung anhand von Kriterien wie Schweregrad (Severity), Wahrscheinlichkeit (Probability) und Erkennbarkeit (Detectability) – ähnlich wie bei einer FMEA.

4. Priorisierung

- Funktionen werden nach ihrer Kritikalität in Klassen eingeteilt (z. B. hoch, mittel, niedrig)
- Diese Priorisierung dient als Grundlage für:
 - Sicherheitsanforderungen (ASIL-Einstufung)
 - Testtiefe und -strategie
 - Architekturentscheidungen (z. B. Redundanz, Partitionierung)
 - Ressourcenallokation im Projekt

5. Dokumentation und Nachverfolgbarkeit

- Die Ergebnisse der CPA werden dokumentiert, z. B. in einer Tabelle oder Matrix
- Rückverfolgbarkeit zu Anforderungen, Sicherheitszielen und Systemarchitektur wird sichergestellt
- Die Analyse sollte regelmäßig aktualisiert werden – insbesondere bei Änderungen im Design oder bei neuen Erkenntnissen aus Tests oder Reviews

Ein Fallbeispiel zur Anwendung der CPA-Methode nach NIST IR 8179 findet sich im Annex, Kapitel 7.1.3.

5.2.3.3 Hinweise/Besonderheiten/Unterschiede beim Einsatz im Softwarekontext

Im Softwarekontext unterstützt die Critical Path Analysis (CPA) die Bewertung von Softwarefunktionalitäten hinsichtlich ihrer Systemrelevanz, Robustheit und Cybersecurity. Sie wird in frühen Entwicklungsphasen eingesetzt, um Risiken abzuschätzen und Prioritäten zu setzen. Darüber hinaus unterstützt die Methode sowohl das Safety- als auch das Security-Engineering sowie die Testplanung.

5.2.3.4 Voraussetzungen (Input)

Für die Durchführung einer Criticality and Prioritization Analysis (CPA) sind eine klare System- und Funktionsabgrenzung, verfügbare Architektur- und Anforderungsinformationen, definierte Bewertungskriterien, ein interdisziplinäres Team sowie geeignete Werkzeuge und Templates erforderlich.

5.2.3.5 Anwendungstiefe und Beendigungskriterien

Die Critical Path Analysis (CPA) wird auf System- und Funktionalitätsebene durchgeführt. Dabei erfolgt die Betrachtung im Nutzungskontext, unabhängig von konkreten Implementierungsdetails, um kritische Abhängigkeiten frühzeitig zu erkennen und systemisch bewerten zu können.

5.2.3.6 Anforderungen an Aufwand, Team und Tools

Der Aufwand für die Durchführung der Critical Path Analysis (CPA) ist als mittel einzustufen, da eine strukturierte Bewertung erfolgt, ohne dass eine tiefgehende technische Analyse erforderlich ist. Methodenexpertise im engeren Sinne ist nicht notwendig – ein grundlegendes Verständnis für das

Systemverhalten und die sicherheitsrelevanten Anforderungen genügt. Für eine fundierte Durchführung wird ein interdisziplinäres Team empfohlen, das Expertise aus den Bereichen Software, Architektur, Safety und Security einbringt.

5.2.3.7 Nutzen im Softwarekontext (Vorteile)

Die Critical Path Analysis (CPA) ermöglicht die frühzeitige Identifikation kritischer Softwarefunktionalitäten und unterstützt eine ganzheitliche Betrachtung von Sicherheit, Robustheit und Cybersecurity. Sie liefert wertvolle Beiträge zur Architekturentscheidung und zur Entwicklung von Absicherungsstrategien. Darüber hinaus lässt sich die Methode effektiv mit anderen Verfahren wie der Fehlerbaum-Analyse (FTA), Szenarienanalysen oder Threat Modeling kombinieren.

5.2.3.8 Einschränkungen der Methode im Softwarekontext (Nachteile)

Im Softwarekontext bietet die Critical Path Analysis (CPA) keine detaillierte Fehleranalyse wie etwa die FMEA. Sie erlaubt keine quantitative Risikoabschätzung und leitet Maßnahmen nicht automatisch ab. Zudem bleiben technische Wechselwirkungen zwischen Systemelementen unberücksichtigt, was die Aussagekraft in komplexen Architekturen einschränken kann.

5.2.3.9 Ergebnis (Output)

Die Ergebnisse einer Criticality and Prioritization Analysis (CPA) in der automatisierten Softwareentwicklung bestehen typischerweise aus einer bewerteten und priorisierten Liste von Softwarefunktionen oder -komponenten, die nach ihrer Kritikalität im Hinblick auf Sicherheit, Betriebsrelevanz, Verfügbarkeit und ggf. Cybersecurity geordnet sind.

Die Ergebnisse der CPA dienen als Grundlage für viele nachgelagerte Aktivitäten, z. B.:

Bereich	Nutzung der CPA-Ergebnisse
Funktionale Sicherheit (ISO 26262)	Ableitung oder Verfeinerung von ASIL-Einstufungen, Sicherheitsanforderungen
Teststrategie	Festlegung von Testtiefe, Testprioritäten und Absicherungsmaßnahmen
Softwarearchitektur	Entscheidungen zu Modularisierung, Partitionierung, Redundanz

Cybersecurity (ISO/SAE 21434)	Identifikation von sicherheitskritischen Angriffszielen
Projektplanung	Ressourcen- und Zeitplanung basierend auf Kritikalität
Change-Impact-Analyse	Bewertung der Auswirkungen bei Änderungen an hochkritischen Funktionen

5.2.3.10 Referenzen/Weiterführende Literatur

Paulsen, C., Boyens, J., Bartol, N., & Winkler, K. (2018). *Criticality analysis process model – prioritizing systems and components*. National Institute of Standards and Technology. <https://doi.org/10.6028/NIST.IR.8179>

5.2.4 DRBFM (Design Review Based on Failure Mode)

5.2.4.1 Ziel der Methode

DRBFM (Design Review Based on Failure Mode) verfolgt das Ziel, potenzielle Design- oder Architekturschwachstellen in technischen Systemen frühzeitig zu erkennen, bevor größere Implementierungsaufwendungen entstehen. Dabei liegt der Fokus auf Änderungsentscheidungen wie neuen Schnittstellen, Migrationspfaden, API-Verträgen, Architektur- oder Deployment-Änderungen. Die Methode ermöglicht eine transparente Abbildung der Entwicklungsaufgaben aus diesen Änderungen, indem Ursachen, potenzielle Auswirkungen und Detektionsmechanismen formuliert sowie geeignete Gegenmaßnahmen definiert werden. Ein wesentliches Ziel ist zudem die bereichsübergreifende Kommunikation zwischen Entwicklung, Architektur, Sicherheit, QA und Betrieb, um eine gemeinsame Risikobewertung und verantwortliche Zuweisung sicherzustellen.

5.2.4.2 Grundsätzliches Vorgehen

Das Vorgehen ist detailliert beschrieben in SAE J2886 (04/2023). Die grundsätzliche Durchführung umfasst die folgenden 5 Schritte:

1. Festlegung des Betrachtungsrahmens

Im ersten Schritt wird der Betrachtungsrahmen festgelegt. Der Fokus kann sich auf einzelne Bauteile, Verbindungsstellen, Soft- oder Hardwarekomponenten sowie System- oder Subsystemfunktionen richten. Dieser Fokus ergibt sich beispielsweise aus Änderungen gegenüber einem existierenden Design oder aus der Übernahme von Teilen aus Neuentwicklungen

2. **Erfassung der Funktionen innerhalb des Betrachtungsrahmens**
 Innerhalb des festgelegten Rahmens werden alle Funktionen erfasst, die die Bauteile und ihre Schnittstellen ausüben; diese Funktionen möglichst systemisch zugeordnet. Funktionen dienen als Träger von Anforderungen und bilden den Rahmen für den Design-Entwurf

3. **Design-Entwurf und Verhaltensanalyse**
 Der Design-Entwurf wird nun genau daraufhin untersucht, wie sich das Verhalten während aller Phasen des Produktlebenszyklus – Montage, Test, Lagerung, Transport, Einbau beim Kunden, Betrieb und Service – gestaltet. Idealerweise kommen hierbei visualisierte Verhaltensmodelle zum Einsatz

4. **Identifikation von Schwächen und Risiken**
 In diesem Schritt werden mögliche Schwächen, Fehler und Risiken frühzeitig aufgespürt, um sie später abzubauen. Dafür werden die Grundursachen (Root Causes) von Fehlern gesucht, also Parameter im Design oder Prozess, die Überschreitungen der Funktionsgrenzen auslösen. Diese Parameter liefern oft Hinweise darauf, wie sich der Fehler bereits im Vorfeld vermeiden lässt

5. **Festlegung der Maßnahmen zur Abstellung potenzieller Fehler**
 Da Maßnahmen im Fokus eines Design-Reviews stehen, ist dieser finale Schritt besonders wichtig. Im Ergebnis liegt ein Maßnahmenplan vor, der die identifizierten Risiken und Schwächen adäquat behandelt

5.2.4.3 Hinweise/Besonderheiten/Unterschiede beim Einsatz im Softwarekontext

Im Softwarekontext sollten DRBFM-Workshops primär bei signifikanten Design-Entscheidungen oder Änderungen erfolgen, etwa bei Architekturwechseln, der Einführung neuer Schnittstellen, Migrationspfaden, Performance-Verbesserungen oder sicherheitsrelevanten Maßnahmen. Anstelle physischer Failure Modes werden Software-Failure-Modes verwendet, etwa Verletzungen des API-Vertrags, Dateninkonsistenzen, Race Conditions, Fehlkonfigurationen von Deployments oder Probleme im Datenfluss. Der Ansatz ist modular und beginnt beispielsweise mit einem kleinen, risikohaften Change-Set und wird iterativ auf weitere Änderungen ausgeweitet. Die Verwendung eines softwarespezifischen Templates, das Felder für Change,

Failure Mode, Causes, Effects, Detection, Mitigation, Risikobewertung, Verantwortliche und Fristen enthält, unterstützt die Nutzer:innen bei der Durchführung. Um den Nutzen zu maximieren, kann die DRBFM fest in den Softwarelebenszyklus integriert werden, sodass Ergebnisse mit Requirements-Reviews, Architektur-Reviews, API-Vertragsprüfungen, Sicherheits-Reviews und automatisierten Tests verknüpft werden.

Die Zuverlässigkeits- und Sicherheitsziele müssen dem Softwareprodukt angepasst sein (z. B. Fehlerraten, Ausfallzeiten, Sicherheitslücken, Benutzerakzeptanz).

Statt physischer Bauteile werden Software-Designentscheidungen, Schnittstellen, Architekturen, Algorithmen und Konfigurationsparameter geprüft.

5.2.4.4 Voraussetzungen (Input)

Für eine effektive DRBFM in der Softwareumgebung braucht es eine klare (Change-)Beschreibung, in der „Was“, „Warum“ und „Wie“ erläutert werden. Dazu gehören Architektur- oder Design-Diagramme wie Komponentendiagramme, Sequenz- oder Paketdiagramme sowie Data-Flow-Diagramme, um die Auswirkungen der Änderung nachvollziehen zu können. Außerdem sind Schnittstellen- und Vertragsdokumente erforderlich, beispielsweise API-Verträge, Datenmodelle, Contract-Tests oder Schema-Änderungen. Risikokriterien helfen bei der Priorisierung der behandelten Punkte, etwa Sicherheitsrelevanz, Kundennutzen oder Risiken, die durch Änderungen entstehen könnten. Ein Detektionsplan mit geplanten Metriken, Logs, Tests, Canary-Strategien oder Static/Dynamic Analysis ist hilfreich, ebenso wie klare Verantwortlichkeiten: Wer ist Change-Owner, wer verantwortet Architektur, Security, QA und Betrieb? Falls vorhanden, können Prototypen oder Pilotergebnisse als Referenz dienen, um die praktische Wirkung besser einschätzen zu können.

5.2.4.5 Anwendungstiefe und Beendigungskriterien

Die Anwendungstiefe hängt vom Risiko der Softwarelösung oder -änderung ab: Eine tiefe Anwendung umfasst alle Failure Modes, Causes, Effects, Detection-Mechanismen und Mitigationsmaßnahmen. Eine mittlere Tiefe konzentriert sich auf eine klare Liste von Failure Modes und deren Detektion sowie Gegenmaßnahmen. Eine oberflächliche Anwendung eignet sich für weniger risikoreiche Lösungen oder Änderungen und fokussiert sich auf die wesentlichen Risiken und Gegenmaßnahmen. Die Beendigungskriterien beinhalten, dass alle identifizierten Failure Modes bewertet sind, inklusive Ur-

sachen, Auswirkungen, Detektoren, Gegenmaßnahmen, Verantwortlichkeiten und Fristen. Die Risikobewertung sollte klar priorisiert sein und Detektionsmechanismen in Form von Tests, Monitoring oder Canary-Releases etabliert sein. Alle Ergebnisse sollten in relevanten Dokumenten verlinkt oder in Build-/Release-Dokumentationen aufgenommen werden. Offene Fragen oder Abhängigkeiten müssen eskaliert oder in die nächsten Review-Termine eingeplant werden.

5.2.4.6 Anforderungen an Aufwand, Team und Tools

Der benötigte Aufwand hängt von der Tiefe der DRBFM-Session ab: Niedrig bedeutet eine Sitzung von etwa einer Stunde mit 1–2 Change-Requests und minimaler Dokumentation. Mittel bezeichnet eine Session von 90 bis 150 Minuten mit 2–4 Changes, strukturierter Dokumentation und einer oder zwei Review-Runden. Hoch erfordert mehrere Sessions, mindestens 5 Changes, eine umfangreiche Dokumentation und Integration in Release-Pläne sowie ggf. zusätzliche Security- oder Datenschutz-Reviews. Das Team-Spektrum umfasst typischerweise Change-Owner aus Entwicklung oder Architektur, Backend- oder Frontend-Architekt:innen, QA/Test-Lead, Security-Owner bei sicherheitsrelevanten Themen, DevOps/Betrieb sowie gegebenenfalls Product-Owner und Compliance-Vertreter:innen. Als Tools werden genutzt: zentrale Dokumentation in Wikis, Diagramm-Tools wie Visio, Lucidchart oder diagrams.net, Ticketsysteme wie Jira, Azure DevOps oder GitHub Issues, Tests und Monitoring-Tools (Contract-Tests, Integrationstests, Logging/Observability-Plattformen) sowie Templates speziell für Software-DRBFM.

5.2.4.7 Nutzen im Softwarekontext (Vorteile)

Der Hauptnutzen von DRBFM im Softwarekontext besteht darin, Risiken früh zu erkennen und gezielt Gegenmaßnahmen zu definieren, bevor eine Implementierung teuer wird oder live geht. Dadurch steigen die Qualität und Zuverlässigkeit des Systems, weil bewusst auf potenzielle Probleme reagiert wird. Durch klare Verantwortlichkeiten und Fristen wird der Abstimmungsaufwand reduziert und die Zusammenarbeit über Entwicklung, Architektur, Sicherheit, QA und Betrieb hinweg verbessert. DRBFM unterstützt eine bessere Release-Planung, da Risiken in die Planung aufgenommen werden können; Canary-Deployments oder Feature Flags lassen sich gezielter einsetzen. Zudem entsteht eine nachvollziehbare Entscheidungsdocumentation, die die Gründe für Designentscheidungen transparent macht und eine klare Audit-Historie liefert.

Sie ist besonders technik- und praxisorientiert, da sie die Risikobewertung dann durchführt, wenn wichtige und relevante Änderungen zur Entscheidung anstehen und die darauffolgende Realisierung kurzfristig Feedback zur Entscheidung geben kann.

5.2.4.8 Einschränkungen der Methode im Softwarekontext (Nachteile)

DRBFM kann insbesondere bei größeren Changes mit vielen Failure Modes und umfangreicher Dokumentation zu einem signifikanten Review-Aufwand führen. Die Einschätzungen zu Ursachen und Auswirkungen basieren stark auf Erfahrungen des Teams, was zu Subjektivität führen kann und das Risiko einer Über- oder Unterabbildung birgt. Die Risikobewertung resultiert ebenfalls aus der Erfahrung des Teams und basiert auf den potenziellen Auswirkungen der identifizierten Failure Modes mit einer hohen/mittleren/niedrigen Risikoskala; eine Bewertung des Auftretens wird für die Priorisierung von Risiken in der Regel nicht berücksichtigt: Das Team erarbeitet alle identifizierten Risiken, bis eine Entscheidungsgrundlage erreicht ist.

Langfristige oder Architektur-Risiken, die jenseits einzelner Changes liegen, finden weniger Berücksichtigung. Ohne konsequente Nachverfolgung der definierten Maßnahmen besteht die Gefahr, dass der Nutzen verlorengeht. Nicht alle Organisationen benötigen eine vollständige DRBFM-Runde; je nach Kontext muss die Methode angepasst werden.

5.2.4.9 Ergebnis (Output)

Das zentrale Ergebnis einer DRBFM-Sitzung im Softwarekontext ist ein umfassendes Change-Register, das die Change-Requests mit Beschreibung enthält, gefolgt von den identifizierten Failure Modes pro Change, den vermuteten Causes, den potenziellen Effects sowie Detektionsmechanismen und passenden Mitigationsmaßnahmen. Eine Risikobewertung pro Failure Mode ergänzt den Output, ebenso wie klare Verantwortlichkeiten und Deadlines. Verknüpfungen zu Architektur-Dokumenten, API-Verträgen, Test-Suiten und Release-Plänen ermöglichen eine lückenlose Nachverfolgung. Als weiterer Output dient eine Freigabe- oder Gate-Ebene, die angibt, ob der Change den nächsten Schritt im Release-Prozess passieren darf, z. B. nach Code- oder Security-Reviews. Optional können Vorlagen mit konkreten Feldern wie Change/Design-Entscheidung, Potenzielle Failure Modes, Causes, Effects, Detection, Mitigation, Risikoskala, Owner/Deadline sowie Notes / Offene Fragen genutzt werden, um die Dokumentation zu standardisieren.

5.2.4.10 Referenzen/Weiterführende Literatur

SAE International (2023). *Design Review Based on Failure Modes (DRBFM)*, SAE Standard J2886_202304, Reaffirmed April 2023, Issued March 2013. https://doi.org/10.4271/J2886_202304

5.2.5 ETA (Event Tree Analysis) – Ereignisbaumanalyse

5.2.5.1 Ziel der Methode

Die Ereignisbaumanalyse (Event Tree Analysis, ETA) dient der systematischen Untersuchung und Darstellung der logischen Verknüpfungen von Komponenten- und Teilsystemausfällen, um die Auswirkungen möglicher unerwünschter Ereignisse sowie deren funktionale Zusammenhänge sichtbar zu machen. Im Gegensatz zur Fehlerbaumanalyse verfolgt die ETA ein Bottom-up-Vorgehen, bei dem von einem initialen Ereignis ausgegangen wird und die möglichen Folgeereignisse sowie deren Verzweigungen analysiert werden. Ziel ist es, verschiedene Entwicklungspfade eines Systems nach dem Eintreten eines bestimmten Ereignisses zu erfassen und deren Konsequenzen zu bewerten.

5.2.5.2 Grundsätzliches Vorgehen

1. Definition des auslösenden Ereignisses (Initiating Event)

- Beispiel: „Softwaremodul stürzt ab“ oder „Sensor liefert fehlerhafte Daten“

2. Identifikation relevanter Sicherheitsfunktionen oder Barrieren

- Welche Schutzmaßnahmen oder Reaktionen sind vorgesehen? (z. B. Fehlererkennung, Rückfallebene, Neustartmechanismus)

3. Aufbau des Ereignisbaums

- Für jede Barriere wird ein Pfad mit zwei möglichen Ausgängen modelliert: **Erfolg** (Barriere funktioniert) oder **Versagen** (Barriere versagt)
- Daraus ergeben sich verschiedene **Ereignispfade**, die jeweils zu einem bestimmten Endzustand führen

4. Beschreibung der Endzustände (Outcomes)

- Jeder Pfad endet in einem Ergebnis, z. B. „System stabilisiert sich“, „Teilfunktion fällt aus“, „Gesamtsystemausfall“

5. Quantitative Bewertung

- Jedem Pfad kann eine Wahrscheinlichkeit zugewiesen werden, basierend auf den Erfolgs-/Versagenswahrscheinlichkeiten der Barrieren
- Daraus ergibt sich die **Wahrscheinlichkeit jedes Endzustands**

6. Interpretation und Maßnahmen

- Identifikation von Pfaden mit hohem Risiko
- Ableitung von Verbesserungsmaßnahmen (z. B. zusätzliche Barrieren, Erhöhung der Zuverlässigkeit)

5.2.5.3 Hinweise/Besonderheiten/Unterschiede beim Einsatz im Softwarekontext

Im Softwarekontext sollte die Ereignisbaumanalyse gezielt auf die Identifikation des kritischen Pfads ausgerichtet sein. Ausgangspunkt ist ein initiales Ereignis – beispielsweise eine Verletzung funktionaler Anforderungen oder ein unerwünschter Systemzustand – von dem aus mögliche Folgeereignisse und deren Verzweigungen analysiert werden. Entscheidend für die Anwendung ist die Ermittlung geeigneter Basis-Events, also die Frage, welche konkreten Ereignisse oder Zustände im Softwaresystem auftreten könnten und wie sich diese logisch weiterentwickeln können. Die Methode eignet sich besonders zur Darstellung von Reaktionsketten und zur Analyse von Systemverhalten unter verschiedenen Bedingungen, etwa bei fehlerhaften Eingaben, Kommunikationsabbrüchen oder nicht erfüllten Anforderungen.

5.2.5.4 Voraussetzungen (Input)

Für die Anwendung der Ereignisbaumanalyse im Softwarekontext sind bestimmte Voraussetzungen erforderlich. Es muss eine Softwarefunktionalität vorliegen, die analysiert werden soll, sowie eine Funktionsarchitektur – auch als funktionelle Architektur oder Komposition bezeichnet –, die die logischen Zusammenhänge innerhalb des Systems abbildet. Darüber hinaus müssen unerwünschte Ereignisse einzelner Komponenten oder Funktionalitäten bereits identifiziert sein, um als Ausgangspunkt für die Analyse dienen zu können. Nur wenn diese Elemente klar beschrieben sind, kann die ETA sinnvoll eingesetzt werden, um mögliche Entwicklungspfade und deren Auswirkungen systematisch zu untersuchen.

5.2.5.5 Anwendungstiefe und Beendigungskriterien

Die Anwendungstiefe der Ereignisbaumanalyse richtet sich nach der Identifikation einer geeigneten Start-Ebene, die als Ausgangspunkt für die Analyse dient. Die Tiefe der Analyse ist nicht grundsätzlich eingeschränkt, sondern hängt maßgeblich von der Komplexität des betrachteten Systems und dem Umfang der zu untersuchenden Ereignispfade ab. Je detaillierter die Verzweigungen und Folgeereignisse betrachtet werden, desto höher ist der Aufwand für die Modellierung und Auswertung.

Eine sinnvolle Begrenzung der Analyse kann durch logische Kapselung erfolgen, insbesondere dann, wenn die Untersuchung der dynamischen und statischen Architektur abgeschlossen ist. In solchen Fällen bietet sich eine Modularisierung oder Segmentierung der Ereignispfade an, um die Übersichtlichkeit zu wahren und die Analyse effizient zu gestalten.

5.2.5.6 Anforderungen an Aufwand, Team und Tools

Die Anwendung der Ereignisbaumanalyse im Softwarekontext ist mit einem mittleren bis hohen Aufwand verbunden. Zwar ist die erforderliche Methodenexpertise vergleichsweise gering, da die Grundprinzipien der ETA leicht verständlich und strukturiert anwendbar sind, dennoch erfordert die Analyse eine sorgfältige Modellierung der Ereignispfade und eine klare Definition der Ausgangs- und Folgeereignisse.

Für eine fundierte Durchführung ist in der Regel ein interdisziplinäres Team notwendig, das sowohl die funktionale Architektur als auch die Systemverhalten im Fehlerfall versteht. Die Zusammenarbeit von Softwareentwickler:innen, Systemarchitekt:innen und Safety-Expert:innen trägt wesentlich zur Qualität und Aussagekraft der Analyse bei. Der Einsatz unterstützender Tools zur Visualisierung und Dokumentation kann den Analyseprozess erleichtern, ist aber nicht zwingend erforderlich.

5.2.5.7 Nutzen im Softwarekontext (Vorteile)

Im Softwarekontext bietet die Ereignisbaumanalyse mehrere spezifische Vorteile. Sie ist gut geeignet, Fehler in gekapselten Komponenten zu analysieren und deren Auswirkungen systematisch zu verfolgen. Die Methode erlaubt eine strukturierte Darstellung möglicher Reaktionspfade nach dem Eintreten eines initialen Ereignisses und unterstützt so die Nachvollziehbarkeit komplexer Systemverhalten. Aufgrund ihrer starken Fokussierung kann die ETA gezielt zur Untersuchung kritischer Abläufe eingesetzt werden und

eignet sich sowohl für einen präventiven Einsatz – zur frühzeitigen Identifikation potenzieller Risiken – als auch für eine reaktive Analyse bereits eingetretener Fehlfunktionen. Besonders wirkungsvoll ist die Methode in Kombination mit der Fehlerbaumanalyse (FTA), da sich durch die Verbindung beider Ansätze sowohl Ursachen als auch Folgen unerwünschter Ereignisse umfassend darstellen lassen.

5.2.5.8 Einschränkungen der Methode im Softwarekontext (Nachteile)

Trotz ihrer Stärken weist die Ereignisbaumanalyse im Softwarekontext mehrere methodische Einschränkungen auf. Die Betrachtung erfolgt nicht ganzheitlich, sondern fokussiert auf einzelne Ereignispfade und isolierte Risiken. Es werden keine Maßnahmen zur Risikominderung (Risk Mitigation) definiert, ebenso fehlt eine systematische Ableitung von Handlungsoptionen oder eine Entscheidungshilfe zum Umgang mit identifizierten Risiken. Auch eine Bewertung der Risiken im Sinne einer Priorisierung oder Gewichtung ist nicht Bestandteil der Methode. Die ETA liefert somit primär eine strukturierte Darstellung möglicher Systemreaktionen auf ein initiales Ereignis, ohne weiterführende Schritte im Risikomanagement zu integrieren.

5.2.5.9 Ergebnis (Output)

Das Ergebnis der Ereignisbaumanalyse besteht in einer strukturierten Darstellung möglicher Systemreaktionen auf ein initiales Ereignis sowie der damit verbundenen Verzweigungen und Folgeereignisse. Im Softwarekontext können daraus bewertete Risiken abgeleitet werden, wobei die Bewertung in der Regel qualitativ erfolgt. Anders als im Hardwarebereich stehen für Software keine belastbaren Ausfallwahrscheinlichkeiten zur Verfügung, sodass eine quantitative Risikoabschätzung meist nicht möglich ist. Die ETA liefert dennoch wertvolle Erkenntnisse über potenzielle Fehlerfolgen und unterstützt die systematische Analyse von Reaktionspfaden innerhalb komplexer Softwarearchitekturen.

5.2.5.10 Referenzen/Weiterführende Literatur

International Electrotechnical Commission. (2010). *IEC 62502:2010, Analysis techniques for dependability – Event Tree Analysis (ETA)*.

5.2.6 FMEA (Fehlermöglichkeits- und -einflussanalyse)

5.2.6.1 Ziel der Methode

Die Fehlermöglichkeits- und -einflussanalyse (FMEA) ist eine etablierte, systematische Methode zur frühzeitigen Identifikation und Bewertung potenzieller Fehlerquellen im Produktentstehungsprozess. Ziel ist es, Schwachstellen in der technischen Funktionalität bereits in frühen Entwicklungsphasen zu erkennen und deren mögliche Auswirkungen auf das Gesamtsystem zu analysieren. Im Rahmen einer methodischen Risikoanalyse werden potenzielle Fehler, deren Ursachen und Folgen strukturiert erfasst und bewertet. Auf dieser Grundlage lassen sich gezielt Maßnahmen ableiten, die der Fehlervermeidung oder -minderung dienen. Die Anwendung der FMEA trägt somit wesentlich dazu bei, Risiken proaktiv zu steuern und die Zuverlässigkeit softwarebasierter Funktionen nachhaltig zu erhöhen.

5.2.6.2 Grundsätzliches Vorgehen

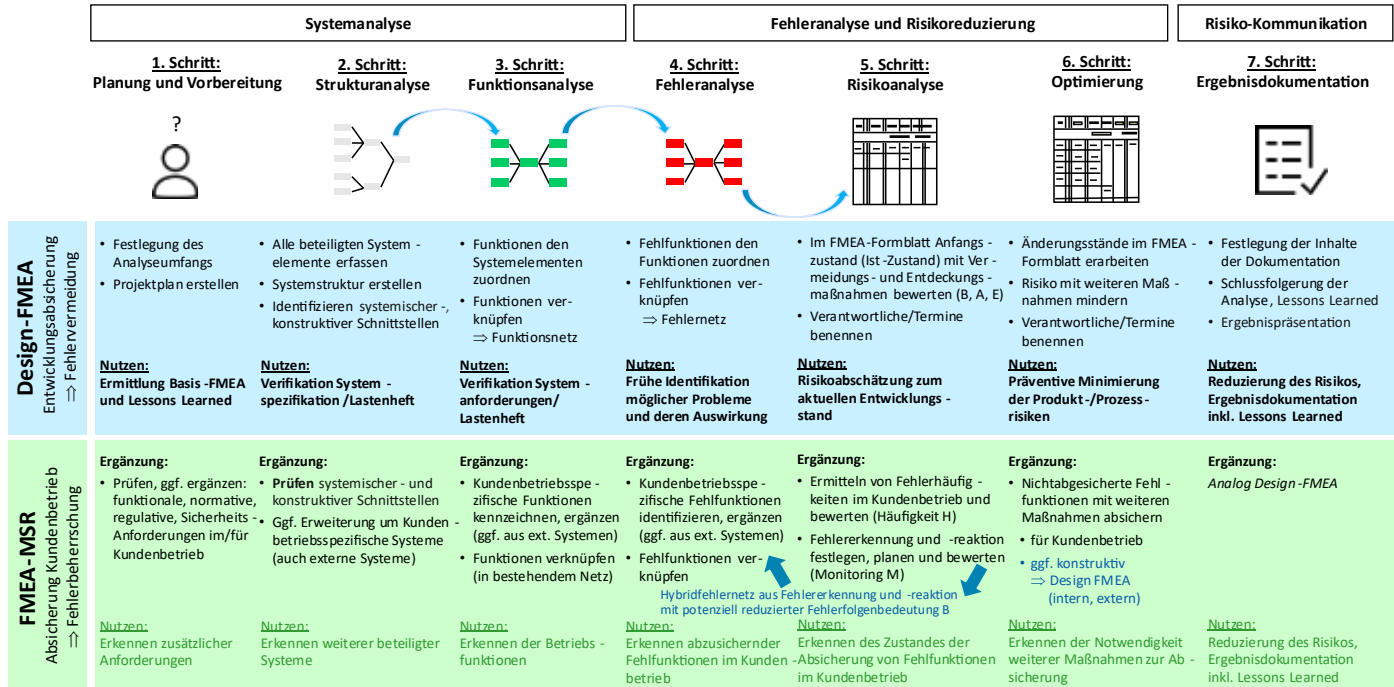


Abbildung 5-4: Grundlegender Ablauf Design-FMEA und FMEA-MSR

Die Durchführung einer Design-FMEA erfolgt in sieben methodischen Schritten. Zunächst wird in der Planungs- und Vorbereitungsphase der Analyseumfang festgelegt, ein Projektplan erstellt und die Dokumentationsinhalte definiert. Anschließend werden in der Strukturanalyse die für den Analyseumfang festgelegten Systemelemente erfasst und in einer Systemstruktur abgebildet. Dabei werden auch systemische und konstruktive Schnittstellen identifiziert. In der Funktionsanalyse erfolgt die Zuordnung der Funktionen zu den Systemelementen sowie die Verknüpfung dieser Funktionen in einem Funktionsnetz. Darauf aufbauend werden in der Fehleranalyse potenzielle Fehlfunktionen identifiziert, den Funktionen zugeordnet und in einem Fehlernetz miteinander verknüpft. Die Risikoanalyse bewertet die identifizierten Risiken anhand der Kriterien Bedeutung (B), Auftreten (A) und Entdeckung (E) im Ist-Zustand und dokumentiert diese im FMEA-Formblatt. In der Optimierungsphase werden Maßnahmen zur Risikominderung abgeleitet sowie Verantwortlichkeiten und Termine festgelegt. Nach Umsetzung der Maßnahmen werden potenzielles Auftreten und Entdeckung erneut bewertet. Abschließend werden die Ergebnisse der Analyse zusammengefasst und dokumentiert.

5.2.6.3 Hinweise/Besonderheiten/Unterschiede beim Einsatz im Softwarekontext

Bei der Anwendung der FMEA auf softwarebasierte Funktionalitäten liegt der Fokus nicht auf einzelnen Code-Zeilen oder physischen Ausfällen, sondern auf den funktionalen Abläufen innerhalb der Software sowie auf logischen Fehlern. Ziel ist es, systematisch potenzielle Schwachstellen in der Softwarelogik zu identifizieren – etwa fehlerhafte Zustandsübergänge, unvollständige Anforderungen, unvollständige Schnittstellendefinitionen oder unzureichende Fehlerbehandlung – und deren Auswirkungen auf die Gesamtfunktionalität zu bewerten. Die Analyse erfolgt auf einer abstrakten Ebene, die sich an der funktionalen Architektur und den vorgesehenen Betriebszuständen orientiert. So lassen sich Risiken frühzeitig erkennen und gezielte Maßnahmen zur Verbesserung der Robustheit und Zuverlässigkeit softwarebasierter Systeme ableiten.

5.2.6.4 Voraussetzungen (Input)

Für eine wirksame Anwendung der FMEA im Bereich softwarebasierter Funktionalitäten müssen zentrale Systeminformationen vorliegen. Dazu zählen die Beschreibung der Softwarefunktionalitäten sowie die System- und Softwarearchitektur mitsamt ihren Schnittstellen, Datenflüssen und Betriebszuständen. Ebenso ist die Kenntnis der funktionellen Architektur –

auch als Funktionsarchitektur oder Komposition bezeichnet – erforderlich. Diese Informationen bilden die Grundlage für eine strukturierte Analyse potenzieller Fehlerursachen und -folgen. Analog zur Vorgehensweise bei der Fehlerbaumanalyse (FTA) ermöglicht die klare Definition dieser Eingangsgrößen eine zielgerichtete und nachvollziehbare Risikoanalyse im Softwarekontext.

5.2.6.5 Anwendungstiefe und Beendigungskriterien

Die Anwendung der FMEA im Softwarekontext konzentriert sich auf die Funktionen der Software, die zur Umsetzung bzw. Realisierung der jeweiligen Systemanforderungen beitragen. Im Mittelpunkt stehen dabei die funktionalen Abläufe und deren Beitrag zur Gesamtfunktionalität des Systems – nicht die technische Implementierung auf Code-Ebene. Die Analyse erfolgt so lange, bis alle relevanten Funktionen hinsichtlich potenzieller Fehlerursachen und -folgen bewertet und geeignete Maßnahmen zur Risikominimierung definiert wurden. Als Beendigungskriterium gilt, dass für alle betrachteten Funktionen eine nachvollziehbare Risikoabschätzung vorliegt und keine offenen, unbehandelten Risiken mehr bestehen, die die Erfüllung der Systemanforderungen gefährden könnten.

5.2.6.6 Anforderungen an Aufwand, Team und Tools

Die Durchführung einer FMEA im Softwarekontext ist mit einem hohen methodischen und organisatorischen Aufwand verbunden. Aufgrund der Komplexität softwarebasierter Funktionalitäten und der Vielzahl möglicher Fehlerursachen erfordert die Methode ein hohes Maß an Expertise in der Risikoanalyse sowie fundierte Kenntnisse der Softwarearchitektur und funktionalen Zusammenhänge. Die Analyse sollte durch ein interdisziplinäres Team erfolgen, das sowohl Systemverständnis als auch Softwareentwicklungs- und Qualitätssicherungskompetenz vereint. Zur Unterstützung der strukturierten Durchführung und Dokumentation sind geeignete Tools erforderlich, die die Modellierung funktionaler Abläufe, die Bewertung von Risiken und die Nachverfolgung von Maßnahmen ermöglichen.

5.2.6.7 Nutzen im Softwarekontext (Vorteile)

Die Anwendung der FMEA auf softwarebasierte Funktionalitäten bietet eine strukturierte und nachvollziehbare Unterstützung bei der Entwicklung von Strategien zur Fehlervermeidung, -erkennung und -begrenzung. Durch die systematische Analyse funktionaler Abläufe und potenzieller Fehlerursachen lassen sich Risiken frühzeitig identifizieren und gezielt adressieren.

Dies stärkt nicht nur die Robustheit der Software, sondern fördert auch ein tieferes Verständnis für kritische Zusammenhänge innerhalb der Systemarchitektur. Die Methode trägt somit wesentlich zur Qualitätssicherung und zur Erhöhung der funktionalen Sicherheit softwareintensiver Systeme bei.

5.2.6.8 Einschränkungen der Methode im Softwarekontext (Nachteile)

Die FMEA bietet keine Garantie für die Vollständigkeit der Risikoanalyse. Sie basiert auf den zum Zeitpunkt der Durchführung bekannten Systemen, Abläufen und Funktionen und kann daher nur innerhalb dieses definierten Rahmens Aussagen treffen. Risiken, die aus nicht erkannten, nicht spezifizierten oder nicht modellierten Zuständen resultieren, bleiben außerhalb des Betrachtungshorizonts. Insbesondere im Softwarekontext, in dem komplexe Zustandsübergänge und emergente Verhaltensweisen auftreten können, ist diese Einschränkung von besonderer Bedeutung. Die Aussagekraft der Analyse hängt somit wesentlich von der Qualität und Vollständigkeit der Eingangsinformationen ab.

5.2.6.9 Ergebnis (Output)

Das Ergebnis der FMEA im Softwarekontext besteht in der systematischen Identifikation kritischer und nicht abgesicherter Zustände innerhalb der betrachteten Funktionen und Abläufe. Durch die strukturierte Bewertung potenzieller Fehlerursachen und -folgen werden gezielt Schwachstellen sichtbar, die ein Risiko für die zuverlässige Umsetzung der Systemanforderungen darstellen können. Die Analyse liefert konkrete Absicherungsmaßnahmen, die zur Fehlervermeidung, -erkennung oder -begrenzung beitragen und somit die funktionale Sicherheit und Robustheit softwarebasierter Systeme stärken.

5.2.6.10 Referenzen/Weiterführende Literatur

AIAG, & VDA. (2019). *FMEA-Handbuch. Design-FMEA, Prozess-FMEA, FMEA-MSR (Monitoring und Systemreaktion)*

5.2.7 FMEA-MSR (FMEA-Monitoring und Systemreaktion)

5.2.7.1 Ziel der Methode

Die FMEA-MSR (Monitoring und Systemreaktion) erweitert die klassische FMEA um die Betrachtung von Risiken im realen Kundenbetrieb. Ziel der

Methode ist es, kritische Betriebszustände zu identifizieren, die während der Nutzung des Produkts auftreten können und zu sicherheitsrelevanten, funktionalen oder regulatorischen Risiken führen. Durch die systematische Analyse dieser Zustände und der vorhandenen Überwachungs- und Reaktionsmechanismen wird überprüft, ob potenzielle Risiken ausreichend abgesichert sind. Die FMEA-MSR trägt somit zur Nachweisführung bei, dass das System auch unter realen Betriebsbedingungen robust und sicher funktioniert.

5.2.7.2 Grundsätzliches Vorgehen

Die FMEA-MSR folgt denselben sieben methodischen Schritten wie die Design-FMEA (siehe Abbildung 5-4), wird jedoch um spezifische Aspekte, die im Kundenbetrieb wichtig sind, erweitert. In der Planungs- und Vorbereitungsphase werden zusätzlich kundenspezifische Anforderungen und Betriebsbedingungen berücksichtigt. Die Strukturanalyse wird um externe und kundenbetriebene Systeme sowie Schnittstellen ergänzt. In der Funktionsanalyse werden Betriebsfunktionen identifiziert und in bestehende Funktionsnetze integriert. Die Fehleranalyse umfasst die Identifikation kundenspezifischer Fehlfunktionen, die mit bestehenden Fehlernetzen verknüpft werden. Die Risikoanalyse wird durch die Bewertung der Fehlerhäufigkeit im Kundenbetrieb (H) sowie der Monitoring-Fähigkeit (M) ergänzt. Dabei werden Konzepte zur Fehlererkennung und -reaktion geplant und bewertet. In der Optimierungsphase werden nicht sichere Fehlfunktionen durch zusätzliche Maßnahmen gezielt abgesichert, insbesondere im Hinblick auf das Auftreten im Kundenbetrieb. Die Ergebnisdokumentation enthält die erweiterten Analyseergebnisse, Monitoring-Konzepte und Lessons Learned zur Absicherung des Betriebszustands.

5.2.7.3 Hinweise/Besonderheiten/Unterschiede beim Einsatz im Softwarekontext

Im Softwarekontext richtet sich die FMEA-MSR auf Fehlfunktionen, die im realen Kundenbetrieb auftreten können und sicherheitsrelevante, regulatorische oder funktionale Risiken nach sich ziehen. Im Fokus stehen dabei nicht nur die Softwarefunktionen selbst, sondern insbesondere deren Verhalten unter Betriebsbedingungen, wie etwa fehlerhafte Zustandsübergänge, unzureichende Plausibilitätsprüfungen oder fehlende Reaktionsmechanismen. Die Methode unterstützt die systematische Bewertung, ob kritische Zustände erkannt und durch geeignete Monitoring- und Reaktionsstrategien abgesichert sind – etwa durch Warnungen, Notfallstrategien oder Rückfallebenen.

5.2.7.4 Voraussetzungen (Input)

Für eine fundierte Durchführung der FMEA-MSR im Softwarekontext sind umfassende Informationen zu sicherheitsrelevanten, regulatorischen und funktionalen Anforderungen erforderlich. Dazu zählen insbesondere die Sicherheitsziele und deren ASIL-Einstufungen, normative und gesetzliche Vorgaben sowie detaillierte System- und Funktionsbeschreibungen. Ebenso notwendig sind Schnittstellenbeschreibungen zu angrenzenden Systemen, die relevanten Betriebszustände, bekannte Fehlermöglichkeiten aus vorgelegten Analysen (z. B. D-FMEA oder FTA) sowie Fehlernetze. Soweit verfügbar, fließen auch Informationen zu Diagnosefähigkeiten und bestehenden Reaktionsstrategien ein. Ergänzt wird die Analyse durch qualitative, attributive und – sofern vorhanden – quantitative Nachweise zur Absicherung der betrachteten Risiken im Kundenbetrieb.

5.2.7.5 Anwendungstiefe und Beendigungskriterien

Die FMEA-MSR wird typischerweise auf System- und Subsystemebene durchgeführt, da hier die relevanten Betriebszustände und deren potenzielle Risiken im Kundenbetrieb am besten erfasst werden können. Für ein vertieftes Fehlerverständnis kann es jedoch hilfreich sein, ergänzend auch die Komponenten- oder Bauteilebene zu betrachten – insbesondere, wenn dort kritische Zustände entstehen oder überwacht werden. Die Analyse gilt als abgeschlossen, wenn alle relevanten Betriebszustände hinsichtlich ihrer Überwachbarkeit und Reaktionsfähigkeit bewertet wurden und keine offenen Risiken verbleiben, die sicherheitsrelevante, regulatorische oder funktionale Anforderungen gefährden.

5.2.7.6 Anforderungen an Aufwand, Team und Tools

Die Durchführung der FMEA-MSR ist mit einem hohen Aufwand verbunden und erfordert ausgeprägte Methodenkompetenz. Aufgrund der Komplexität der Analyse – insbesondere im Hinblick auf die Bewertung von Betriebszuständen, Diagnosefähigkeiten und Reaktionsstrategien – ist ein interdisziplinäres Team notwendig. Dieses sollte Expertise in den Bereichen funktionale Sicherheit, Softwareentwicklung, Systemarchitektur und Qualitätsmanagement vereinen. Zur strukturierten Durchführung und Dokumentation sind geeignete Tools erforderlich, die die Modellierung von Betriebszuständen, die Risikoanalyse sowie die Ableitung und Nachverfolgung von Absicherungsmaßnahmen unterstützen.

5.2.7.7 Nutzen im Softwarekontext (Vorteile)

Die FMEA-MSR bietet im Softwarekontext eine strukturierte Unterstützung bei der Entwicklung und Bewertung von Strategien zur Fehlervermeidung, -erkennung und -begrenzung im realen Kundenbetrieb. Durch die gezielte Analyse kritischer Betriebszustände und deren Absicherung trägt die Methode dazu bei, sicherheitsrelevante und regulatorische Anforderungen auch unter Betriebsbedingungen zuverlässig zu erfüllen. Sie fördert ein tieferes Verständnis für die Wirksamkeit von Diagnose- und Reaktionsmechanismen und unterstützt die kontinuierliche Verbesserung der funktionalen Sicherheit softwareintensiver Systeme.

5.2.7.8 Einschränkungen der Methode im Softwarekontext (Nachteile)

Die FMEA-MSR ermöglicht keine Aussage über die Vollständigkeit der Risikoanalyse. Sie basiert auf bekannten und definierten Betriebszuständen und kann daher Risiken, die aus nicht erkannten oder nicht spezifizierten Zuständen entstehen, nicht erfassen. Zudem hat die Methode keinen Einfluss auf die Auftretenswahrscheinlichkeit von Fehlfunktionen – sie bewertet ausschließlich deren Auswirkungen und die Wirksamkeit vorhandener Überwachungs- und Reaktionsstrategien. Die Aussagekraft der Analyse hängt somit stark von der Qualität und Vollständigkeit der Eingangsinformationen sowie der Systembeschreibung ab.

5.2.7.9 Ergebnis (Output)

Das Ergebnis der FMEA-MSR besteht im Nachweis der Qualität der Absicherung sicherheitsrelevanter, regulatorischer und normativer Anforderungen im Kundenbetrieb. Durch die strukturierte Bewertung kritischer Betriebszustände und der vorhandenen Überwachungs- und Reaktionsmechanismen wird sichtbar, ob potenzielle Risiken angemessen erkannt und beherrscht werden. Die Analyse liefert somit eine fundierte Grundlage für die Bewertung der funktionalen Sicherheit und der Robustheit softwarebasierter Systeme unter realen Einsatzbedingungen.

5.2.7.10 Referenzen/Weiterführende Literatur

AIAG, & VDA. (2019). *FMEA-Handbuch. Design-FMEA, Prozess-FMEA, FMEA-MSR (Monitoring und Systemreaktion)*

5.2.8 FMEDA (Failure Modes, Effects and Diagnostic Analysis)

5.2.8.1 Ziel der Methode

Die FMEDA (Failure Modes, Effects and Diagnostic Analysis) dient der quantitativen Bewertung der funktionalen Sicherheit technischer Systeme gemäß ISO 26262. Ziel ist es, potenzielle Ausfallarten systematisch zu identifizieren, deren Auswirkungen auf die Sicherheitsziele zu analysieren und die Wirksamkeit vorhandener Diagnosemechanismen zu bewerten. Auf dieser Basis werden sicherheitsrelevante Kennzahlen wie der SPFM (Single Point Fault Metric) und der LFM (Latent Fault Metric) berechnet, die für die Sicherheitsfreigabe und die Einhaltung regulatorischer Anforderungen erforderlich sind. Die Methode findet vor allem Anwendung in elektronischen Systemarchitekturen, während ihr Einsatz für rein softwarebezogene Umfänge nur eingeschränkt möglich ist.

5.2.8.2 Grundsätzliches Vorgehen

Das Vorgehen umfasst folgende Schritte:

- 1. Identifikation der Komponenten (Hardware- oder Softwareelemente)**
Grundlage der Analyse ist eine vollständige Systembeschreibung, in der alle relevanten Komponenten – sowohl hardware- als auch softwareseitig – erfasst werden
- 2. Ermittlung der möglichen Ausfallarten (Failure Modes)**
Für jede Komponente werden potenzielle Ausfallarten systematisch identifiziert, z. B. Kurzschluss, Speicherfehler oder Kommunikationsverlust
- 3. Analyse der Auswirkungen auf die Sicherheitsziele**
Die Auswirkungen der Ausfälle auf die definierten Sicherheitsziele werden bewertet, insbesondere im Hinblick auf die Verletzung funktionaler Anforderungen oder die Gefährdung von Personen
- 4. Zuordnung von Diagnosemechanismen und deren Abdeckung**
Für jede Ausfallart wird geprüft, ob ein Diagnosemechanismus existiert und wie zuverlässig dieser den Fehler erkennt. Die Diagnoseabdeckung (Coverage) ist ein zentraler Parameter für die spätere Bewertung
- 5. Berechnung von Sicherheitskennzahlen (z. B. SPFM, LFM)**
Auf Basis der Ausfallraten und Diagnoseabdeckungen werden

quantitative Kennzahlen berechnet, wie der Single Point Fault Metric (SPFM) und der Latent Fault Metric (LFM), die für die Sicherheitsfreigabe erforderlich sind

6. Dokumentation der Ergebnisse für den Sicherheitsnachweis

Die Analyseergebnisse werden strukturiert dokumentiert und dienen als Nachweis gegenüber Kunden, Auditoren oder Zertifizierungsstellen. Sie sind integraler Bestandteil des Sicherheitskonzepts

5.2.8.3 Hinweise/Besonderheiten/Unterschiede beim Einsatz im Softwarekontext

Im Unterschied zur Hardwareanalyse liegt der Fokus bei der Anwendung der FMEDA auf Software nicht auf physikalischen Ausfällen, sondern auf systematischen Fehlern – etwa fehlerhaften Algorithmen, Datenkorruption oder unzureichender Fehlerbehandlung. Die Diagnosemechanismen sind in diesem Kontext meist logisch aufgebaut, beispielsweise durch Plausibilitätsprüfungen, Watchdog-Funktionen oder Redundanzprüfungen. Eine besondere Herausforderung besteht in der Quantifizierung von Ausfallwahrscheinlichkeiten, da diese bei Software nicht direkt messbar sind. Häufig erfolgt daher eine qualitative Bewertung oder es werden Annahmen auf Basis von Erfahrungswerten getroffen. Die FMEDA im Softwarebereich erfordert eine enge Abstimmung mit den Entwicklungs- und Qualitätssicherungsteams, um die Analyse realitätsnah und belastbar zu gestalten.

5.2.8.4 Voraussetzungen (Input)

Für eine fundierte Durchführung der FMEDA sind mehrere Eingangsinformationen erforderlich. Dazu zählen zunächst die Systemarchitektur sowie die definierten Sicherheitsanforderungen, die den Rahmen für die Analyse setzen. Eine vollständige Liste der Softwarekomponenten und deren Funktionen bildet die Grundlage für die Identifikation potenzieller Fehlermodi. Ergänzend wird ein Fehlermodi-Katalog benötigt, der häufig aus vorangegangenen FMEA-Aktivitäten stammt. Die verfügbaren Diagnosemechanismen und deren Wirksamkeit müssen ebenfalls bekannt sein, um die Abdeckung bewerten zu können. Während bei Hardware zuverlässige Ausfallraten verwendet werden können, stützt sich die Analyse bei Software meist auf Fehlerannahmen oder Erfahrungswerte, da verlässliche statistische Daten oft fehlen.

5.2.8.5 Anwendungstiefe und Beendigungskriterien

Die FMEDA wird typischerweise bis auf Elementebene durchgeführt, das heißt für einzelne Hardwarekomponenten wie integrierte Schaltkreise (ICs) oder für spezifische Softwaremodule. Diese Detailtiefe ermöglicht eine präzise Bewertung der Ausfallarten und der Wirksamkeit von Diagnosemechanismen. Die Analyse gilt als abgeschlossen, wenn alle relevanten Fehlermodi identifiziert, deren Auswirkungen bewertet, geeignete Diagnosemaßnahmen zugeordnet und die erforderlichen Sicherheitskennzahlen – wie SPFM und LFM – berechnet wurden. Die Ergebnisse fließen in den Sicherheitsnachweis ein und bilden eine wichtige Grundlage für die Freigabe sicherheitskritischer Systeme.

5.2.8.6 Anforderungen an Aufwand, Team und Tools

Die Durchführung einer FMEDA erfordert ein interdisziplinäres Team, bestehend aus Safety-Expert:innen, Software- und Hardwareentwickler:innen sowie gegebenenfalls Systemarchitekt:innen. Die Analyse erfolgt häufig mithilfe von Tabellenkalkulationen oder spezialisierten FMEDA-Tools, die die strukturierte Erfassung und Bewertung von Fehlermodi, Diagnoseabdeckungen und Sicherheitskennzahlen unterstützen. Der Aufwand ist in der Regel hoch, da eine detaillierte Betrachtung auf Komponentenebene notwendig ist und sowohl funktionale als auch diagnostische Aspekte berücksichtigt werden müssen. Besonders in komplexen Systemen mit hohem Sicherheitsanspruch ist eine sorgfältige Planung und Abstimmung erforderlich.

5.2.8.7 Nutzen im Softwarekontext (Vorteile)

Auch wenn die FMEDA ursprünglich für Hardwarekomponenten entwickelt wurde, bietet sie im Softwarekontext wertvolle Vorteile. Sie liefert quantitative Sicherheitsnachweise, die zur Erfüllung der Anforderungen gemäß ISO 26262 beitragen. Darüber hinaus unterstützt sie die Validierung und Bewertung von logischen Diagnosemechanismen, wie etwa Plausibilitätsprüfungen oder Überwachungsfunktionen. Durch die strukturierte Analyse wird das Verständnis für kritische Fehlerpfade innerhalb der Softwarearchitektur deutlich erhöht, was wiederum die gezielte Ableitung von Sicherheitsmaßnahmen erleichtert.

5.2.8.8 Einschränkungen der Methode im Softwarekontext (Nachteile)

Die Anwendung der FMEDA auf Software bringt spezifische Herausforderungen mit sich. Im Gegensatz zu elektronischen Komponenten liegen für

Softwareelemente keine standardisierten Ausfallwahrscheinlichkeiten vor, was die quantitative Bewertung erschwert. Häufig müssen Annahmen auf Basis von Erfahrungswerten getroffen werden, was die Aussagekraft der Analyse begrenzen kann. Bei komplexer Software steigt zudem der Aufwand erheblich, da eine detaillierte Betrachtung aller relevanten Module und Fehlerpfade erforderlich ist. Ohne klare Zieldefinition besteht die Gefahr einer Überdokumentation, die zwar umfangreich ist, aber keinen echten Mehrwert für die Sicherheitsbewertung liefert.

5.2.8.9 Ergebnis (Output)

Das zentrale Ergebnis einer FMEDA ist eine strukturierte Tabelle, in der die identifizierten Fehlermodi, die zugeordneten Diagnosemechanismen sowie deren Abdeckungsgrade dokumentiert sind. Auf Basis dieser Daten werden die sicherheitsrelevanten Kennzahlen berechnet, darunter z. B. der SPFM (Single Point Fault Metric), der LFM (Latent Fault Metric) und der PMHF (Probabilistic Metric for Hardware Failures). Diese Kennzahlen dienen als quantitativer Nachweis für die Sicherheitskonformität gemäß ISO 26262 und sind essenziell für die Freigabe sicherheitskritischer Funktionen und Systeme.

5.2.8.10 Referenzen/Weiterführende Literatur

International Organization for Standardization. (2018). *ISO 26262: Road Vehicles – Functional Safety*. ISO.

SAE International. *SAE J2980:2018 – Considerations for ISO 26262 ASIL Hazard Classification*.

Ross, H.-L. (2016). *Functional Safety for Road Vehicles – New Challenges and Solutions for E-Mobility and Automated Driving*. Springer.

5.2.9 FTA (Fault Tree Analysis) – Fehlerbaumanalyse

5.2.9.1 Ziel der Methode

Die Fehlerbaumanalyse (Fault Tree Analysis, FTA) dient der systematischen Untersuchung und Darstellung fehlfunktionaler logischer Verknüpfungen von Komponenten- und Teilsystemausfällen. Ziel ist es, die Auswirkungen möglicher unerwünschter Ereignisse sowie deren funktionale Zusammenhänge transparent zu machen. Die Methode folgt einem Top-down-Vorgehen: Ausgehend von einem definierten Top-Event – beispielsweise einem sicherheitskritischen Systemausfall – werden die zugrunde liegenden

Fehlerursachen in einer baumartigen Struktur abgeleitet. Die logischen Verknüpfungen der Fehlfunktionen erfolgen dabei auf Basis boolescher Logik, etwa durch UND- oder ODER-Verknüpfungen.

Die Analyse kann sowohl qualitativ als auch quantitativ erfolgen:

- In der qualitativen Analyse steht die Identifikation kritischer Fehlerpfade und Schwachstellen im Vordergrund
- Die quantitative Analyse erlaubt darüber hinaus die Bewertung der Eintrittswahrscheinlichkeit des Softwarefehlers, sofern entsprechende Daten (z. B. Fehlerraten, Testabdeckungen) vorliegen

5.2.9.2 Grundsätzliches Vorgehen

Das Vorgehen umfasst folgende Schritte:

1. Definition des Top-Ereignisses

- Das Top-Ereignis ist das unerwünschte Verhalten auf Systemebene, das durch einen Softwarefehler verursacht werden kann
- Beispiel: *„Falsche Aktivierung der Notbremsfunktion“* oder *„Nicht-Erkennen eines Hindernisses durch das ADAS⁴“*

2. Systemverständnis aufbauen

- Sammeln von Informationen über das System (z. B. Schaltpläne, Funktionsbeschreibungen, Expertenwissen)
- Analyse der betroffenen Softwarefunktion im Kontext des Gesamtsystems:
 - Architekturdiagramme, Funktionsbeschreibungen, Software- und Hardwareschnittstellen
 - Austausch mit Fachexpert:innen (z. B. Funktionsentwickler:innen, Softwarearchitekt:innen, Safety Engineers)
 - Berücksichtigung von Anforderungen aus Normen wie ISO 26262

⁴ ADAS steht für Advanced Driver Assistance Systems, also „fortgeschrittene Fahrerassistenzsysteme“. Diese Systeme unterstützen die Fahrzeugführenden durch automatisierte Funktionen wie Spurhalteassistent, Notbremsassistent oder adaptive Geschwindigkeitsregelung. Sie gelten als Vorstufe zum automatisierten und autonomen Fahren und sind ein zentraler Bestandteil moderner Fahrzeugsicherheit und Komforttechnologie.

3. Erstellung des Fehlerbaums

- Zerlegung des Top-Ereignisses in mögliche Ursachen mithilfe logischer Verknüpfungen:
 - UND-Gatter: Mehrere Bedingungen müssen gleichzeitig erfüllt sein
 - ODER-Gatter: Eine von mehreren Bedingungen reicht aus
- Typische Ursachen im Softwarekontext:
 - fehlerhafte Anforderungen oder Spezifikationen
 - Logikfehler in Algorithmen
 - Kommunikationsfehler zwischen Softwaremodulen
 - fehlerhafte Sensorwerte oder unzureichende Plausibilisierung
 - Timing-Probleme oder Race Conditions

4. Fortsetzung der Zerlegung bis zu den Basiselementen

- Die Analyse wird so lange verfeinert, bis die Basiselemente erreicht sind:
 - einzelne Softwaremodule, Funktionen, Schnittstellen oder externe Einflüsse
 - Auch menschliche Fehler (z. B. bei der Implementierung oder Konfiguration) können Basiselemente sein

5. Durchführung der Analyse

- Qualitativ: Identifikation von Fehlerpfaden, kritischen Komponenten und logischen Abhängigkeiten
- Quantitativ: Bewertung der Eintrittswahrscheinlichkeit des Top-Ereignisses, sofern Daten wie Fehlerraten, Testabdeckungen oder Diagnosewirksamkeit vorliegen

6. Bewertung und Interpretation

- Analyse der Minimalen Schnittmengen (Minimal Cut Sets) zur Identifikation von:
 - kritischen Fehlerkombinationen
 - Einzelfehlerursachen mit hoher Relevanz
 - Schwachstellen im Softwaredesign oder in der Fehlerbehandlung

7. Dokumentation und Review

- Vollständige Dokumentation des Fehlerbaums, der Annahmen und der Ergebnisse
- Durchführung von Reviews mit interdisziplinären Teams (Software, Safety, Test, Architektur)
- Sicherstellung der Rückverfolgbarkeit zu Anforderungen und Sicherheitszielen
- Aktualisierung des Fehlerbaums bei Änderungen an Software, Architektur oder Anforderungen

5.2.9.3 Hinweise/Besonderheiten/Unterschiede beim Einsatz im Softwarekontext

Im Softwarekontext ist der Einsatz der Fehlerbaumanalyse (FTA) mit bestimmten Einschränkungen möglich und sollte gezielt auf die Identifikation des kritischen Pfads ausgerichtet sein. Ausgangspunkt ist ein definiertes kritisches Top-Event, von dem aus die relevanten logischen Verknüpfungen analysiert werden. Dabei steht die Ermittlung geeigneter Basis-Events im Vordergrund – also die Frage, welche konkreten Fehlfunktionen oder unerwünschten Zustände im Softwaresystem auftreten könnten. Da softwarebedingte Fehler häufig nicht auf physikalische Ausfälle zurückzuführen sind, sondern auf logische oder systematische Ursachen, ist eine präzise Definition der Basis-Events essenziell.

Die Anwendung der booleschen Logik bleibt auch im Softwarekontext erhalten. Der quantitative Teil der FTA, etwa die Berechnung von Eintrittswahrscheinlichkeiten, ist jedoch im Softwarebereich in der Regel nicht anwendbar, da belastbare statistische Daten zu Softwarefehlern meist fehlen. Die Methode eignet sich daher primär zur qualitativen Analyse, insbesondere zur strukturierten Identifikation von Fehlerpfaden und zur Ableitung von Maßnahmen zur Fehlervermeidung oder -begrenzung.

5.2.9.4 Voraussetzungen (Input)

Für die Anwendung der Fehlerbaumanalyse im Softwarekontext sind bestimmte Voraussetzungen erforderlich. Zunächst muss eine Softwarefunktionalität vorliegen, die analysiert werden soll, sowie ein unerwünschtes Ereignis, das typischerweise durch die Verletzung definierter Goals beschrieben wird. Darüber hinaus ist eine Funktionsarchitektur – auch als funktio-

nelle Architektur oder Komposition bezeichnet – notwendig, um die logischen Zusammenhänge und Abhängigkeiten innerhalb des Systems nachvollziehbar darzustellen.

Die Methode sollte präventiv eingesetzt werden, das heißt bereits in frühen Entwicklungsphasen, um potenzielle Risiken systematisch zu identifizieren. Dabei erfolgt die Analyse durch die Negierung von Funktionalitäten und Anforderungen oder eine mögliche Verletzung eines Sicherheitsziels, um mögliche unerwünschte Ereignisse sichtbar zu machen. Bei der präventiven Verwendung der Methode steht somit nicht die retrospektive Untersuchung eines eingetretenen Fehlers im Vordergrund, sondern die vorausschauende Identifikation kritischer Pfade und Schwachstellen im Systemdesign.

5.2.9.5 Anwendungstiefe und Beendigungskriterien

Die Anwendungstiefe der Fehlerbaumanalyse ist grundsätzlich nicht eingeschränkt und richtet sich nach der Komplexität des betrachteten Systems sowie der Relevanz des analysierten Top-Events. Je nach Zielsetzung kann die Analyse bis auf tiefere funktionale Ebenen fortgeführt werden, um auch indirekte oder systemübergreifende Zusammenhänge zu erfassen. Die Entscheidung über die Tiefe der Analyse erfolgt dabei situationsabhängig und orientiert sich an der Bedeutung und Kritikalität des Top-Events im Gesamtsystem.

Beendigungskriterien ergeben sich typischerweise dann, wenn alle relevanten Basis-Events identifiziert und die logischen Verknüpfungen vollständig nachvollzogen wurden – oder wenn die weitere Detaillierung keinen zusätzlichen Erkenntnisgewinn mehr bietet. Im Softwarekontext kann die Analyse beendet werden, sobald die kritischen Pfade ausreichend beschrieben und potenzielle Schwachstellen in der funktionalen Architektur erkannt sind.

5.2.9.6 Anforderungen an Aufwand, Team und Tools

Die Durchführung der Fehlerbaumanalyse im Softwarekontext ist mit einem sehr hohen Aufwand verbunden – unabhängig von der konkreten Systemkomplexität. Auch wenn die quantitative Methodenexpertise aufgrund der eingeschränkten Anwendbarkeit statistischer Verfahren gering ausfällt, erfordert die qualitative Analyse ein tiefes Verständnis der funktionalen Zusammenhänge und eine ausgeprägte methodische Kompetenz. Die Erstellung und Pflege der Baumstruktur, insbesondere bei komplexen Softwarearchitekturen, ist ressourcenintensiv und verlangt eine sorgfältige und strukturierte Vorgehensweise.

Ein dediziertes Team ist für die Anwendung der Methode nicht zwingend erforderlich, jedoch kann die Einbindung von Fachpersonen aus Softwarearchitektur, Safety oder Systementwicklung die Qualität der Analyse deutlich erhöhen. Der Einsatz unterstützender Tools zur Modellierung und Visualisierung ist hilfreich, aber nicht zwingend notwendig. Entscheidend ist die Fähigkeit, die logischen Verknüpfungen und funktionalen Abhängigkeiten präzise zu erfassen und darzustellen.

5.2.9.7 Nutzen im Softwarekontext (Vorteile)

Im Softwarekontext bietet die Fehlerbaumanalyse (FTA) mehrere spezifische Vorteile. Die Methode ist stark fokussiert und eignet sich besonders für einen gezielten Deep Dive in funktionale Zusammenhänge und Fehlerursachen. Durch die strukturierte Top-down-Analyse lassen sich kritische Pfade systematisch identifizieren und nachvollziehen, was insbesondere bei komplexen Softwarearchitekturen von hohem Wert ist. Die FTA kann sowohl präventiv – zur frühzeitigen Erkennung potenzieller Risiken – als auch reaktiv – zur Analyse bereits eingetretener Fehlfunktionen – eingesetzt werden. Ihre Stärke liegt in der Fähigkeit, gezielt in die Tiefe zu gehen und dabei logische Verknüpfungen transparent darzustellen.

5.2.9.8 Einschränkungen der Methode im Softwarekontext (Nachteile)

Trotz ihrer Stärken weist die Fehlerbaumanalyse im Softwarekontext auch deutliche methodische Einschränkungen auf. Die Betrachtung erfolgt nicht ganzheitlich, sondern fokussiert auf einzelne Fehlerpfade und isolierte Risiken. Es werden keine Maßnahmen zur Risikominderung (Risk Mitigation) definiert, ebenso fehlt eine systematische Ableitung von Handlungsoptionen oder eine Entscheidungshilfe zum Umgang mit identifizierten Risiken. Auch eine Bewertung der Risiken im Sinne einer Priorisierung oder Gewichtung ist nicht Bestandteil der Methode.

Die FTA liefert somit primär eine strukturierte Darstellung von Fehlerursachen und deren logischen Verknüpfungen, ohne jedoch weiterführende Schritte im Risikomanagement zu integrieren. Eine Ausnahme bildet die quantitative Anwendung der FTA, bei der – sofern belastbare Daten vorliegen – eine Aussage zur Ausfallwahrscheinlichkeit des Top-Events getroffen werden kann. In solchen Fällen ist eine genaue Abschätzung von Risiken möglich, was insbesondere bei sicherheitskritischen Systemen von Bedeutung sein kann.

5.2.9.9 Ergebnis (Output)

Das Ergebnis der Fehlerbaumanalyse besteht in einer strukturierten Darstellung der logischen Zusammenhänge zwischen Fehlfunktionen und deren Ursachen, ausgehend von einem definierten Top-Event. Im Hardwarekontext können daraus bewertete Risiken abgeleitet werden, da dort häufig belastbare Daten zur Berechnung von Ausfallwahrscheinlichkeiten vorliegen. Im Softwarekontext hingegen ist eine solche quantitative Bewertung nicht möglich, da es keine statistisch fundierten Ausfallwahrscheinlichkeiten für Softwarefunktionen gibt. Die Analyse liefert daher primär qualitative Erkenntnisse über kritische Pfade und potenzielle Schwachstellen, die als Grundlage für weitere sicherheitsrelevante Bewertungen und Maßnahmen dienen können.

5.2.9.10 Referenzen/Weiterführende Literatur

VDA. (2020). Risikoanalysen. In *Sicherung der Qualität in der Prozesslandschaft* (VDA Vol. 4, 3rd ed.).

5.2.10 HARA (Hazard Analysis and Risk Assessment)

5.2.10.1 Ziel der Methode

Ziel der Hazard Analysis and Risk Assessment (HARA) ist es, potenzielle Gefährdungen zu identifizieren, die durch Fehlfunktionen softwarebasierter Funktionalitäten entstehen können. Im Anschluss erfolgt eine Bewertung des jeweiligen Risikos, um daraus geeignete Sicherheitsziele abzuleiten, die das Risiko auf ein akzeptables Maß reduzieren. Abschließend wird der Automotive Safety Integrity Level (ASIL) bestimmt, der die erforderliche Sicherheitsstufe für die Umsetzung der Sicherheitsziele angibt.

5.2.10.2 Grundsätzliches Vorgehen

Das Vorgehen umfasst folgende Schritte:

1. Definition und Beschreibung des betrachteten Systems oder Subsystems (Item Definition) mit Angabe der relevanten Funktionen, Betriebszustände und Systemgrenzen
2. Identifikation von Gefährdungen: Analyse möglicher Fehlfunktionen des Systems, die zu Gefährdungen im Fahrzeugbetrieb führen können. Dabei werden das Umfeld, die Fahrer:innen und andere Verkehrsteilnehmer:innen berücksichtigt

3. Bewertung der Gefährdungen: Für jede Gefährdung werden die Parameter Schweregrad (S), Exposition (E) und Beherrschbarkeit (C) bestimmt
4. Risikoeinstufung: Auf Basis der Kombination von S, E und C erfolgt die Ermittlung des Automotive Safety Integrity Level (ASIL)
5. Formulierung von Sicherheitszielen: Für Gefährdungen mit nicht akzeptablem Risiko werden funktionale Sicherheitsziele abgeleitet, die das Risiko auf ein akzeptables Niveau reduzieren sollen
6. Dokumentation und Freigabe: Ergebnisse werden dokumentiert und freigegeben
7. Iterative Aktualisierung: Bei Änderungen im Systemkonzept oder bei neu identifizierten Gefahren wird die HARA angepasst, um sicherzustellen, dass alle Sicherheitsziele aktuell und vollständig sind

5.2.10.3 Hinweise/Besonderheiten/Unterschiede beim Einsatz im Softwarekontext

Die HARA stellt eine High-Level-Analyse dar, die auf dem funktionalen Verhalten des Fahrzeugs basiert. Eine detaillierte System- oder Softwarearchitektur ist für ihre Durchführung nicht erforderlich. Dadurch ist die Anwendung der HARA als Risikoanalyse für softwarebasierte Funktionalitäten nur eingeschränkt möglich.

5.2.10.4 Voraussetzungen (Input)

Für die Durchführung der HARA muss die Item Definition vorliegen. Sie beschreibt die betrachtete Funktionalität auf Fahrzeugebene und bildet die Grundlage für die Identifikation potenzieller Gefährdungen.

5.2.10.5 Anwendungstiefe und Beendigungskriterien

Für jede identifizierte Gefährdung wird ein funktionales Sicherheitsziel formuliert und der zugehörige ASIL bestimmt. Die HARA gilt als abgeschlossen, wenn alle bekannten Gefährdungen bewertet und durch entsprechende Sicherheitsziele abgedeckt sind. Werden im weiteren Entwicklungsverlauf zusätzliche Gefährdungen erkannt, müssen diese durch ergänzende Sicherheitsziele berücksichtigt werden.

5.2.10.6 Anforderungen an Aufwand, Team und Tools

Die HARA wird in der Praxis häufig in Tabellenform dokumentiert. In Einzelfällen kommen auch spezialisierte Analyse-Tools zum Einsatz. Im Vergleich zu anderen Risikoanalysen ist der Aufwand für die Durchführung der HARA als gering einzustufen. Die Verantwortung für die Erstellung liegt typischerweise bei den Safety-Verantwortlichen. Die System-, Hardware- und Softwareentwicklung unterstützen durch fachlichen Input.

5.2.10.7 Nutzen im Softwarekontext (Vorteile)

Die HARA unterstützt die frühzeitige Identifikation potenzieller Gefährdungen, deren Auswirkungen auf System- und Softwarefunktionen in späteren Entwicklungsphasen gezielt berücksichtigt werden können. Sie schafft eine nachvollziehbare Verbindung zwischen den identifizierten Gefährdungen, den daraus abgeleiteten Sicherheitszielen und den entsprechenden Software-Sicherheitsanforderungen. Durch die Risikoanalyse auf Systemebene lassen sich sicherheitskritische Aspekte bereits in einem frühen Stadium der Softwarearchitektur erkennen und bewerten. Im Rahmen des Sicherheitsnachweises (Safety Case) liefert die HARA zudem wichtige Eingangsdaten und Begründungen für sicherheitsrelevante Designentscheidungen im Softwarekontext.

5.2.10.8 Einschränkungen der Methode im Softwarekontext (Nachteile)

Im Softwarekontext zeigt die HARA bestimmte methodische Grenzen. Risiken, die nicht sicherheitsrelevant sind, werden durch die HARA nicht erfasst und können mit dieser Methode auch nicht sinnvoll bewertet werden. Zudem liegt die Verantwortung für die Durchführung der HARA – abhängig vom jeweiligen Entwicklungsgegenstand – häufig beim Auftraggeber, was die methodische Tiefe und den Einfluss auf die Softwareentwicklung zusätzlich einschränken kann.

5.2.10.9 Ergebnis (Output)

Als Ergebnis der HARA wird für jede identifizierte Gefährdung ein funktionales Sicherheitsziel formuliert. Für jedes dieser Sicherheitsziele wird ein ASIL (Automotive Safety Integrity Level) bestimmt, der die erforderliche Sicherheitsstufe angibt. Dabei handelt es sich bei Sicherheitszielen nicht um konkrete technische Lösungen, sondern um Anforderungen an das Systemverhalten – beispielsweise: „Das System muss eine Überladung der Batterie verhindern.“

5.2.10.10 Referenzen/Weiterführende Literatur

International Organization for Standardization. (2018). *ISO 26262:2018 – Road vehicles – Functional safety* (2nd ed.). ISO.

5.2.11 HAZOP (Hazard and Operability Study)

5.2.11.1 Ziel der Methode

Das Ziel der HAZOP-Methode besteht in der präventiven Analyse potenzieller Gefahren und Betriebsrisiken, die durch Fehler oder Fehlfunktionen – insbesondere in Softwarefunktionen – entstehen können. Dabei wird systematisch geprüft, ob geeignete Kontrollen und Schutzmechanismen vorhanden sind, um die möglichen Konsequenzen solcher Abweichungen zu verhindern oder zumindest zu minimieren.

5.2.11.2 Grundsätzliches Vorgehen

Die Durchführung erfolgt in vier aufeinanderfolgenden Schritten:

1. Definition

Zu Beginn wird die Studie initiiert, der Untersuchungsumfang und das Ziel festgelegt sowie Rollen und Verantwortlichkeiten im Team definiert

2. Vorbereitung

Die Studie wird geplant, relevante Daten und Dokumentationen gesammelt und die für die Analyse benötigten Leitwörter sowie mögliche Abweichungen bestimmt

3. Untersuchung

Das System wird in kleinere Subsysteme unterteilt. Für jedes Subsystem werden die Design-Erwartungen und relevante Parameter definiert. Mithilfe von Leitwörtern werden potenzielle Abweichungen identifiziert, deren Ursachen und Konsequenzen analysiert sowie bestehende Schutzmechanismen und mögliche Risikominderungsmaßnahmen erfasst

4. Dokumentation und Nachverfolgung

Die Ergebnisse der Analyse werden systematisch aufgezeichnet, dokumentiert und für die weitere Bearbeitung sowie Nachverfolgung bereitgestellt

5.2.11.3 Hinweise/Besonderheiten/Unterschiede beim Einsatz im Softwarekontext

Die HAZOP-Methode kann auch im Softwarebereich wirkungsvoll eingesetzt werden – vorausgesetzt, die zugrunde liegende Softwarefunktionalität, ihre Parameter sowie die möglichen Konsequenzen von Abweichungen sind auf System- und Softwareebene gut verstanden. Nur bei ausreichendem Verständnis der funktionalen Zusammenhänge lässt sich die Methode effektiv anwenden, um potenzielle Risiken zu identifizieren und geeignete Maßnahmen zur Risikominderung abzuleiten.

5.2.11.4 Voraussetzungen (Input)

Für eine effektive Durchführung der HAZOP-Studie im Softwarekontext müssen eine Funktionsarchitektur (auch funktionelle Architektur oder Komposition genannt) oder eine detaillierte Konstruktion der betrachteten Funktionalität vorliegen. Nur wenn die Struktur und das Verhalten der Software ausreichend beschrieben sind, lassen sich potenzielle Abweichungen systematisch identifizieren und bewerten.

5.2.11.5 Anwendungstiefe und Beendigungskriterien

Die HAZOP-Analyse im Softwarebereich sollte auf Basis einer detaillierten Softwarekonstruktion erfolgen. Erst wenn die funktionalen Details, Parameter und Systemverhalten vollständig beschrieben sind, kann die Untersuchung als abgeschlossen gelten. Die Anwendungstiefe richtet sich dabei nach dem Grad der funktionalen Komplexität und dem Risiko, das von der jeweiligen Softwarefunktion ausgeht.

5.2.11.6 Anforderungen an Aufwand, Team und Tools

Die Durchführung einer HAZOP-Studie im Softwarebereich ist mit einem hohen Aufwand verbunden. Sie erfordert ein interdisziplinäres Team mit fundiertem System- und Softwareverständnis sowie den Einsatz geeigneter Tools zur strukturierten Analyse und Dokumentation. Ohne methodische Unterstützung und klare Rollenverteilung ist eine effektive Durchführung kaum möglich.

5.2.11.7 Nutzen im Softwarekontext (Vorteile)

Die HAZOP-Methode bietet im Softwarebereich besondere Vorteile: Sie ermöglicht eine gezielte Betrachtung jedes einzelnen Parameters innerhalb einer Softwarefunktion und unterstützt durch den Einsatz von Leitwörtern

ein strukturiertes Brainstorming im Team. Dadurch lassen sich potenzielle Fehlerbedingungen systematisch identifizieren sowie geeignete Maßnahmen zur Risikominderung und Reaktionsmechanismen für den Fehlerzustand ableiten.

5.2.11.8 Einschränkungen der Methode im Softwarekontext (Nachteile)

Die HAZOP-Methode stößt im Softwarebereich an bestimmte Grenzen. Risiken, die durch die Interaktion zwischen verschiedenen Komponenten entstehen, lassen sich mit dieser Methode nur eingeschränkt erfassen. Für solche Szenarien sind ergänzende Verfahren wie die Ereignisbaum-Analyse (ETA) oder die Fehlerbaumanalyse (FTA) besser geeignet.

Zudem kann HAZOP nicht alleinstehend zur vollständigen Risikoidentifikation bei komplexen Systemen verwendet werden, sondern sollte stets in Kombination mit weiteren geeigneten Methoden eingesetzt werden.

Ein weiterer Nachteil besteht darin, dass HAZOP das Risiko selbst nicht bewertet, sondern lediglich potenzielle Abweichungen und deren Konsequenzen identifiziert.

Der Erfolg der Analyse hängt stark von der Expertise der Moderatorin oder des Moderators und der beteiligten Fachexpert:innen ab – sowohl in technischer Hinsicht als auch im methodischen Vorgehen.

5.2.11.9 Ergebnis (Output)

Das Ergebnis einer HAZOP-Studie im Softwarekontext umfasst die systematische Identifikation potenzieller funktionaler Fehler, Fehlzustände und Ausfälle. Darüber hinaus werden Maßnahmen zur Minderung der daraus resultierenden Konsequenzen abgeleitet, um die Robustheit und Sicherheit der Softwarefunktionalität zu erhöhen.

5.2.11.10 Referenzen/Weiterführende Literatur

International Electrotechnical Commission. (2016). *IEC 61882:2016 – Hazard and operability studies (HAZOP studies) – Application guide* (2nd ed.). IEC.

5.2.12 SOTIF (Safety Of The Intended Functionality – ISO21448)

5.2.12.1 Ziel der Methode (bzw. des Standards)

SOTIF hat das Ziel, Sicherheitsrisiken zu identifizieren und zu reduzieren, die aus der beabsichtigten Funktionalität eines Systems entstehen – also in Fällen, in denen keine technische Fehlfunktion vorliegt, das System aber dennoch unsicheres Verhalten zeigen kann. Dies ist besonders relevant für Systeme mit Sensorik oder maschinellem Lernen, wie sie in modernen Fahrerassistenzsystemen und automatisierten Fahrfunktionen eingesetzt werden.

SOTIF erweitert die klassische funktionale Sicherheit nach ISO 26262, indem es sich auf Risiken konzentriert, die durch unzureichende Wahrnehmung, Interpretation oder Entscheidungsfindung entstehen – obwohl das System technisch korrekt arbeitet. Dabei ist SOTIF nicht ausschließlich auf ASIL-klassifizierte Systeme beschränkt, sondern auch für Anwendungen ohne sicherheitskritische Einstufung relevant, insbesondere wenn potenziell unsicheres Verhalten durch die beabsichtigte Funktionalität entstehen kann.

5.2.12.2 Grundsätzliches Vorgehen

Die Durchführung erfolgt in sechs Schritten:

- 1. Definition der beabsichtigten Funktionalität**
Klare Beschreibung, was das System leisten soll – inklusive aller Funktionen, die sicherheitsrelevant sein könnten
- 2. Identifikation potenziell unsicherer und unvorhersehbarer Szenarien**
Analyse von Situationen, in denen das System trotz korrekter Funktion ein unsicheres Verhalten zeigen könnte (z. B. Fehlinterpretation durch Sensoren, unzureichende Umgebungswahrnehmung, Szenarien-Abdeckung, unklare Systemgrenzen)
- 3. Bewertung der Risiken**
Einschätzung der sicherheitskritischen Auswirkungen dieser Szenarien – auch ohne klassische Fehlfunktion
- 4. Ableitung von Maßnahmen zur Risikominderung**
Entwicklung technischer oder funktionaler Maßnahmen, um die identifizierten Risiken zu minimieren (z. B. Redundanzen, Plausibilitätsprüfungen, Einschränkungen der Betriebsbedingungen)

5. Validierung und Verifikation

Nachweis, dass die Maßnahmen wirksam sind und die beabsichtigte Funktionalität unter realen Bedingungen sicher ausgeführt wird

6. Iterative Verbesserung

Kontinuierliche Anpassung und Erweiterung der Analyse, insbesondere bei lernenden Systemen oder neuen Einsatzszenarien

5.2.12.3 Hinweise/Besonderheiten/Unterschiede beim Einsatz im Softwarekontext

Im Softwarekontext ist SOTIF anwendbar für Algorithmen zur Umgebungswahrnehmung und Entscheidungslogik. Anders als klassische Fehlermodi (z. B. Speicherfehler) geht es hier um systematisch bedingte Unsicherheiten, etwa durch unzureichende Trainingsdaten, fehlerhafte Sensordateninterpretation oder nicht erkannte Randbedingungen. Die Herausforderung liegt darin, Unsicherheiten zu erkennen, obwohl keine klassische Fehlfunktion vorliegt.

5.2.12.4 Voraussetzungen (Input)

Definition der beabsichtigten Funktionalität:

Es muss klar beschrieben sein, welche Funktion(en) das System ausführen soll – insbesondere im Hinblick auf sicherheitsrelevante Aspekte.

Systemgrenzen und Betriebsbedingungen:

Die Umgebungsbedingungen, unter denen die Software betrieben wird, müssen bekannt und spezifiziert sein (z. B. Wetter, Lichtverhältnisse, Verkehrsszenarien).

Spezifikation der Wahrnehmungs- und Entscheidungslogik:

Die Softwarearchitektur muss nachvollziehbar darstellen, wie Informationen verarbeitet und Entscheidungen getroffen werden – insbesondere bei ML-basierten Funktionen.

Szenarien-Katalog potenziell unsicherer Zustände:

Es sollten typische und kritische Szenarien vorliegen, in denen trotz korrekter Funktion ein unsicheres Verhalten auftreten könnte.

Validierungsstrategie:

Es muss ein Konzept existieren, wie die Sicherheit der beabsichtigten Funktionalität überprüft werden kann – z. B. durch Simulationen oder Tests.

Abgrenzung zu funktionalen Fehlern:

Die Analyse muss sich gezielt auf Risiken konzentrieren, die nicht durch klassische Fehlfunktionen entstehen, sondern durch systemimmanente Unsicherheiten.

5.2.12.5 Anwendungstiefe und Beendigungskriterien

Die Anwendungstiefe der SOTIF-Analyse hängt vom Sicherheitsbedarf und der Komplexität der betrachteten Funktion ab. Die Analyse gilt als abgeschlossen, wenn unsichere Szenarien identifiziert und bewertet wurden, geeignete Maßnahmen zur Risikominderung vorliegen, die verbleibende Restunsicherheit als akzeptabel eingestuft und dokumentiert wurde und eine Validierung durch Tests und Simulationen geplant ist.

5.2.12.6 Anforderungen an Aufwand, Team und Tools

Aufwand

Die Anwendung von SOTIF erfordert einen hohen methodischen Aufwand. Die Identifikation potenziell unsicherer Szenarien ohne klassische Fehlfunktionen ist anspruchsvoll und erfordert tiefes Systemverständnis.

Team

Für eine qualitativ hochwertige SOTIF-Analyse ist ein interdisziplinäres Team mit breiter Expertise unerlässlich. Erforderlich sind Kenntnisse und Erfahrungen in folgenden Bereichen:

- maschinelles Lernen (ML)
- funktionale Sicherheit und SOTIF
- Modellierung funktionaler Zusammenhänge
- Bewertung realistischer Anwendungsszenarien

Tools

Der Einsatz geeigneter Tools unterstützt die Analyse, ist aber nicht zwingend erforderlich. Nützlich sind:

- Modellierungswerkzeuge zur Darstellung von Funktionsarchitekturen und Szenarien (z. B. SysML, UML)
- Simulationsumgebungen zur Validierung sicherheitskritischer Szenarien
- Testframeworks für KI/ML-Funktionen zur Bewertung von Unsicherheiten
- SOTIF-spezifische Erweiterungen

5.2.12.7 Nutzen im Softwarekontext (Vorteile)

- **Erweiterte Sicherheitsbetrachtung:** SOTIF ermöglicht die Analyse von Risiken, die nicht durch klassische Softwarefehler entstehen, sondern durch funktionale Unsicherheiten – z. B. fehlerhafte Interpretation von Sensordaten, unvollständige Entscheidungslogik oder nicht erkannte Umgebungsbedingungen
- **Präventive Risikoidentifikation:** Die Methode unterstützt die frühzeitige Erkennung potenziell unsicherer Szenarien – auch ohne vorherige Fehlfunktionen – und ermöglicht eine vorausschauende Sicherheitsbewertung
- **Relevanz über ISO 26262 hinaus:** SOTIF ist nicht ausschließlich im Kontext funktionaler Sicherheit nach ISO 26262 von Bedeutung. Auch für Systeme ohneASIL-Klassifizierung – etwa Komfortfunktionen oder KI-basierte Assistenzsysteme – bietet SOTIF einen wertvollen methodischen Rahmen zur Analyse und Minimierung funktionaler Risiken
- **Stärkung der Softwarearchitektur durch gezielte Risikoanalyse:** SOTIF unterstützt die Entwicklung robuster Softwarelösungen, indem es gezielt Unsicherheiten in der Wahrnehmung, Interpretation und Entscheidungslogik aufdeckt. Dadurch können Schwachstellen frühzeitig erkannt und systematisch adressiert werden – unabhängig davon, ob klassische Fehlfunktionen vorliegen
- **Verbesserung der Validierungsstrategien:** SOTIF fördert die Entwicklung gezielter Test- und Validierungsmethoden, insbesondere für komplexe oder lernende Systeme, die in dynamischen und schwer vorhersehbaren Umgebungen agieren

5.2.12.8 Einschränkungen der Methode im Softwarekontext (Nachteile)

- **Keine Betrachtung technischer Fehlfunktionen:** SOTIF konzentriert sich ausschließlich auf Risiken, die aus der beabsichtigten Funktionalität entstehen – technische Softwarefehler wie Speicherzugriffsverletzungen oder Laufzeitfehler werden nicht berücksichtigt
- **Fehlende quantitative Risikobewertung:** Der Standard bietet keine systematische Priorisierung oder Gewichtung der identifizierten Risiken. Eine quantitative Risikoanalyse ist nicht Bestandteil von SOTIF

- Keine direkte Ableitung von Risikominderungsmaßnahmen: SOTIF identifiziert funktionale Unsicherheiten, liefert jedoch keine konkreten Maßnahmen zur Risikobeherrschung. Für die Umsetzung sind ergänzende Verfahren wie FMEA oder FMEDA erforderlich
- Hoher Aufwand bei der Szenarienanalyse: Die Identifikation potenziell unsicherer Szenarien – insbesondere bei ML-basierten Funktionen – erfordert tiefes Systemverständnis und ist methodisch anspruchsvoll
- Abhängigkeit von Umgebungsannahmen: Die Risikoanalyse basiert stark auf definierten Betriebsbedingungen. Nicht erkannte oder veränderte Umgebungen können die Aussagekraft der Analyse einschränken
- Begrenzte Toolunterstützung und methodische Reife: Im Vergleich zu etablierten Sicherheitsstandards ist die Tool-Landschaft für SOTIF weniger ausgereift. Die Anwendung des Standards kann uneinheitlich erfolgen und ist stark abhängig von projektspezifischer Interpretation

5.2.12.9 Ergebnis (Output)

Das Ergebnis der Anwendung des SOTIF-Standards besteht in einer systematischen Identifikation und Dokumentation von Risiken, die aus der beabsichtigten Funktionalität eines Systems entstehen können – insbesondere in Fällen, in denen keine technische Fehlfunktion vorliegt, aber dennoch ein unsicheres Verhalten möglich ist.

Im Softwarekontext liefert SOTIF vor allem:

- eine strukturierte Beschreibung potenziell unsicherer Szenarien
- eine Bewertung der funktionalen Grenzen von Wahrnehmung und Entscheidungslogik
- eine Basis für die Ableitung ergänzender Sicherheitsmaßnahmen durch andere Verfahren (z. B. FMEA)

Da SOTIF keine quantitative Risikobewertung vorsieht, liegt der Schwerpunkt auf qualitativen Erkenntnissen, die zur Verbesserung der Softwarearchitektur und zur Absicherung komplexer Systemfunktionen beitragen.

5.2.12.10 Referenzen/Weiterführende Literatur

International Organization for Standardization. (2022). *ISO 21448:2022 – Road vehicles – Safety of the intended functionality*. ISO.

5.2.13 STPA (System-Theoretic Process Analysis)

5.2.13.1 Ziel der Methode (bzw. des Standards)

Die STPA-Methode (System-Theoretic Process Analysis) ist ein systemtheoretischer Ansatz zur Risikoanalyse komplexer, vernetzter Systeme. Im Gegensatz zu klassischen Fehleranalysen betrachtet STPA nicht nur Komponentenversagen, sondern auch fehlerhafte Steueraktionen, unzureichende Kontrolle und systemische Wechselwirkungen. Ziel ist es, unsichere Steueraktionen (UCAs) zu identifizieren und daraus Sicherheitsanforderungen abzuleiten, die sowohl funktionale Sicherheit als auch Robustheit und Cybersecurity berücksichtigen.

5.2.13.2 Grundsätzliches Vorgehen

Die Durchführung erfolgt in vier Schritten:

1. Festlegung des Analyseziels

- Definition, welche **Verluste oder Gefährdungen** vermieden werden sollen (z. B. Unfall, Fehlfunktion, Datenverlust)
- Ableitung daraus: Welche **unsicheren Steuerungsaktionen** könnten zu diesen Verlusten führen?

2. Modellierung der Kontrollstruktur

- Darstellung des Systems als **Steuerungsmodell** mit:
 - Steuerungseinheiten (z. B. Softwaremodule, Steuergeräte)
 - Kontrollaktionen (z. B. Bremsbefehl, Spurhalteaktivierung)
 - Rückmeldungen (Sensorwerte, Statusinformationen)
- Fokus liegt auf **Interaktionen**, nicht nur auf Komponentenfehlern

3. Identifikation unsicherer Steuerungsaktionen

- Analyse, wann eine Steuerungsaktion **unsicher** ist:
 - wird **nicht ausgeführt**, obwohl notwendig
 - wird **falsch ausgeführt** (zu früh, zu spät, zu stark)
 - wird **ausgeführt**, obwohl nicht notwendig
 - wird **unter falschen Bedingungen** ausgeführt

4. Ableitung von Sicherheitsanforderungen

- Für jede unsichere Steuerungsaktion werden **präventive Anforderungen** formuliert
- Diese Anforderungen fließen in das Sicherheitskonzept, die Architektur und die Teststrategie ein

5.2.13.3 Hinweise/Besonderheiten/Unterschiede beim Einsatz im Softwarekontext

Beim Einsatz von STPA im Softwarekontext ergeben sich spezifische Stärken und Besonderheiten. Die Methode eignet sich besonders für Steuerungssoftware, eingebettete Systeme und automatisierte Funktionen. Anders als klassische Fehleranalysen betrachtet STPA nicht nur technische Fehlfunktionen, sondern auch fehlerhafte Interaktionen und Kontrollverluste innerhalb komplexer Systemarchitekturen. Dadurch unterstützt sie die Ableitung präventiver Sicherheitsanforderungen, die bereits vor dem Auftreten konkreter Fehler wirksam werden können.

5.2.13.4 Voraussetzungen (Input)

Für die Durchführung einer STPA sind mehrere Voraussetzungen erforderlich: Ein klar definiertes Analyseziel bildet die Grundlage der Betrachtung. Darüber hinaus ist ein grundlegendes Verständnis des Systems und seiner Steuerungsstruktur notwendig, ebenso wie der Zugang zu bestehenden Anforderungen und Sicherheitszielen. Die Analyse sollte durch ein interdisziplinäres Team erfolgen und mit geeigneten Werkzeugen unterstützt werden, um die Komplexität der Zusammenhänge angemessen erfassen zu können.

5.2.13.5 Anwendungstiefe und Beendigungskriterien

Die STPA-Methode kann auf verschiedenen Ebenen angewendet werden – von der Systemebene über einzelne Komponenten bis hin zu Schnittstellen. Sie eignet sich besonders für Funktionen mit komplexer Kontrolllogik und externen Abhängigkeiten.

5.2.13.6 Anforderungen an Aufwand, Team und Tools

Der Aufwand für die Durchführung einer STPA ist mittel bis hoch und hängt maßgeblich von der Komplexität des Systems sowie der Anzahl der beteiligten Steuerpfade ab. Für eine fundierte Analyse sind Grundkenntnisse in der STPA-Methodik, in Systemtheorie und in der Struktur von Kontrollsystemen

erforderlich. Die Durchführung sollte durch ein multidisziplinäres Team erfolgen, das Expertise aus den Bereichen Software, funktionale Sicherheit, Systemarchitektur und IT-Security vereint. Der Einsatz geeigneter Werkzeuge unterstützt die systematische Erfassung und Auswertung der Analyseergebnisse.

5.2.13.7 Nutzen im Softwarekontext (Vorteile)

Im Softwarekontext bietet STPA eine ganzheitliche Risikoanalyse, die über die Betrachtung technischer Fehler hinausgeht. Sie ermöglicht die Identifikation systemischer Schwächen und unsicherer Steueraktionen, die in komplexen Softwarearchitekturen auftreten können. Auf dieser Basis lassen sich konkrete Sicherheitsanforderungen und einschränkende Bedingungen (Constraints) ableiten. Besonders wertvoll ist die Methode durch ihren präventiven Charakter: Sie kann auch bei unbekanntem Fehlerursachen eingesetzt werden und trägt so frühzeitig zur Erhöhung der funktionalen Sicherheit bei.

5.2.13.8 Einschränkungen der Methode im Softwarekontext (Nachteile)

Trotz ihrer Stärken bringt die STPA im Softwarekontext auch einige Einschränkungen mit sich. Die Methode erfordert ein detailliertes Verständnis des Systems sowie eine strukturierte Modellierung der Steuerungslogik. Maßnahmen zur Risikominderung werden nicht automatisch abgeleitet, sondern müssen manuell erarbeitet und bewertet werden. Der Dokumentationsaufwand ist im Vergleich zu klassischen Methoden deutlich höher. Für eine quantitative Risikoabschätzung, etwa über eine Risikoprioritätszahl (RPZ), ist eine ergänzende Anwendung der FMEA erforderlich.

5.2.13.9 Ergebnis (Output)

Die Ergebnisse einer STPA bestehen aus der Identifikation unsicherer Steuerungsaktionen, der Ableitung systemischer Verlustszenarien und daraus resultierenden Sicherheitsanforderungen. Diese werden weiterverwendet zur Verbesserung von Systemdesign, Sicherheitskonzepten sowie Teststrategien und zur Ergänzung klassischer Methoden wie FMEA, ISO 26262, SOTIF und Cybersecurity-Analysen.

5.2.13.10 Referenzen/Weiterführende Literatur

Leveson, N. G., & Thomas, J. P. (2018): *STPA Handbook – MIT STAMP-001. Practical guidance for applying STPA in real-world projects, including control structures and UCA tables*. Massachusetts Institute of Technology.

Abidi Nasri, S. (2018). *Application of STPA methodology to an automotive system in compliance with ISO 26262* [Bachelorarbeit Universität Stuttgart].

Abdulkhaleq, A., Wagner, S., Lammering, D., Boehmert, H., & Blueher, P. (2017). Using STPA in Compliance with ISO 26262 for developing a safe architecture for fully automated vehicles. In P. Dencker, H. Klenk, H. B. Keller, & E. Plödereder (Hrsg.), *Automotive – Safety & Security 2017. Lecture Notes in Informatics (LNI)*. Gesellschaft für Informatik.

Kaiser, B. (2021). *Applying STPA in the context of SOTIF for ADAS and automated vehicles*. ANSYS Germany.

Leveson, N. G. (2012). Engineering a safer world. In *The MIT Press eBooks*. <https://doi.org/10.7551/mitpress/8179.001.0001>

5.2.14 SWIFT (Structured What-If Technique)

5.2.14.1 Ziel der Methode (bzw. des Standards)

SWIFT wurde ursprünglich als einfachere Alternative zu HAZOP entwickelt. Es handelt sich um eine systematische, teamorientierte Methode, bei der eine Reihe von „Leitworten“ oder Phrasen verwendet wird, die die Moderatorin oder der Moderator in einem Workshop einsetzt, um die Teilnehmer:innen zur Identifikation von Risiken anzuregen. Die Moderatorin oder der Moderator und das Team nutzen standardisierte „Was-wäre-wenn“-Fragen in Kombination mit den Leitworten, um zu untersuchen, wie ein System, eine Anlage, eine Organisation oder ein Verfahren von Abweichungen vom Normalbetrieb und -verhalten betroffen sein könnte.

5.2.14.2 Grundsätzliches Vorgehen

1. Bevor die Analyse beginnt, bereitet die Moderatorin oder der Moderator eine geeignete Liste von Leitworten oder Phrasen vor, die entweder auf einem Standardset basieren oder speziell erstellt werden, um eine umfassende Überprüfung von Gefahren oder Risiken zu ermöglichen

2. Im Workshop werden der externe und interne Kontext des Elements, des Systems, der Änderung oder Situation sowie der Umfang der Analyse besprochen und abgestimmt
3. Die Moderatorin oder der Moderator bittet die Teilnehmer:innen, folgende Punkte einzubringen und zu diskutieren:
 - bekannte Risiken und Gefahren
 - bisherige Erfahrungen und Vorfälle
 - bekannte und bestehende Kontrollen und Schutzmaßnahmen
 - regulatorische Anforderungen und Einschränkungen
4. Die Diskussion wird angeregt, indem eine Frage mit einer „Was-wäre-wenn“-Phrase und einem Leitwort oder Thema gebildet wird. Die zu verwendenden „Was-wäre-wenn“-Phrasen sind: „Was wäre, wenn ...“, „Was würde passieren, wenn ...“, „Könnte jemand oder etwas ...“, „Hat jemals jemand oder etwas ...“. Ziel ist es, das Team dazu zu bringen, potenzielle Szenarien sowie deren Ursachen, Konsequenzen und Auswirkungen zu untersuchen
5. Die Risiken werden zusammengefasst und das Team betrachtet die vorhandenen Kontrollen
6. Die Beschreibung des Risikos, seiner Ursachen und Konsequenzen sowie der erwarteten Kontrollen wird mit dem Team abgestimmt und dokumentiert
7. Das Team prüft, ob die Kontrollen ausreichend und wirksam sind, und einigt sich auf eine Aussage zur Wirksamkeit der Risikokontrolle. Falls diese weniger als zufriedenstellend ist, werden weitere Maßnahmen zur Risikobehandlung diskutiert und potenzielle zusätzliche Kontrollen definiert
8. Während dieser Diskussion werden weitere „Was-wäre-wenn“-Fragen gestellt, um zusätzliche Risiken zu identifizieren

5.2.14.3 Hinweise/Besonderheiten/Unterschiede beim Einsatz im Softwarekontext

Die Methode benötigt eine Anpassung der Leitfragen an softwarespezifische Fehler, wie z. B. „Was wäre, wenn eine Schnittstelle nicht wie spezifiziert funktioniert?“ oder „Was wäre, wenn ein Update fehlschlägt?“.

5.2.14.4 Voraussetzungen (Input)

Für die Durchführung einer SWIFT-Analyse sind mehrere Voraussetzungen erforderlich. Zunächst muss eine Beschreibung des zu analysierenden Systems, Moduls oder Prozesses vorliegen. Darüber hinaus werden vorhandene Anforderungen sowie Architektur- oder Designunterlagen benötigt, um die Analyse fundiert durchführen zu können. Eine zentrale Rolle spielt zudem die Verwendung von Leitworten oder Fragestellungen – etwa „Ausfall“, „Verzögerung“ oder „Fehlbedienung“ –, die als Impulse zur systematischen Identifikation potenzieller Risiken dienen.

5.2.14.5 Anwendungstiefe und Beendigungskriterien

Die Anwendungstiefe der SWIFT-Analyse richtet sich nach dem Ziel der Betrachtung und der Komplexität des Systems. Sie wird typischerweise auf System- oder Subsystemebene durchgeführt, kann aber auch bis hinunter zur Softwaremodulebene angewendet werden. Als Beendigungskriterium gilt, dass alle relevanten „Was-wäre-wenn“-Fragen für die betrachteten Bereiche beantwortet und dokumentiert wurden und keine neuen Risiken mehr identifiziert werden.

5.2.14.6 Anforderungen an Aufwand, Team und Tools

Der Aufwand für die Durchführung einer SWIFT-Analyse ist im Vergleich zu anderen Methoden wie HAZOP eher gering bis mittel. Da keine festgelegte Liste von Guidewords verwendet wird, reduziert sich der Vorbereitungs- und Dokumentationsaufwand deutlich. Besondere Anforderungen an das Team oder spezielle Tools bestehen nicht – die Methode kann flexibel und mit einfachen Mitteln in interdisziplinären Gruppen durchgeführt werden.

5.2.14.7 Nutzen im Softwarekontext (Vorteile)

Dank ihres geringen Aufwands lässt sich die SWIFT-Methode auch im Rahmen regelmäßiger Reviewprozesse effizient einsetzen. Sie eignet sich besonders für dynamische Entwicklungsumgebungen, in denen eine schnelle, strukturierte Risikoabschätzung erforderlich ist.

5.2.14.8 Einschränkungen der Methode im Softwarekontext (Nachteile)

Im Vergleich zu etablierten Verfahren wie FMEA oder HAZOP ist die SWIFT-Methode weniger systematisch und geht nicht so tief in die Analyse. Ihre Wirksamkeit hängt stark von der Erfahrung und dem Fachwissen des

beteiligten Teams ab – fehlt diese Expertise, besteht das Risiko, dass relevante Szenarien übersehen werden. Zudem bietet die Methode keine quantitative Bewertung der identifizierten Risiken, was eine objektive Priorisierung erschwert.

5.2.14.9 Ergebnis (Output)

Die Anwendung der SWIFT-Methode führt zu einer dokumentierten Übersicht identifizierter Risiken und Schwachstellen sowie zu Vorschlägen für mögliche Abhilfemaßnahmen. Diese Ergebnisliste bildet die Grundlage für weitere Bewertungen und Entscheidungen im Rahmen des Risikomanagements.

5.2.14.10 Referenzen/Weiterführende Literatur

International Organization for Standardization. (2019). *IEC/ISO 31010:2019 – Risk Management – Risk Assessment Techniques*. IOS.

5.2.15 TARA (Threat Analysis and Risk Assessment)

5.2.15.1 Ziel der Methode (bzw. des Standards)

Die Threat Analysis and Risk Assessment (TARA) verfolgt das Ziel, Cyber-Bedrohungen systematisch zu identifizieren, zu bewerten und geeignete Maßnahmen zur Risikobehandlung abzuleiten. Im Rahmen der Bewertung werden die identifizierten Risiken unter anderem hinsichtlich der Leichtigkeit eines potenziellen Angriffs, der Auswirkungen auf die Betriebssicherheit, den Schutz personenbezogener Daten und die Produktverfügbarkeit sowie finanzieller und reputationsbezogener Folgen analysiert. Auf Basis dieser Bewertung werden gezielte Gegenmaßnahmen definiert, um ein akzeptables Restrisiko sicherzustellen.

5.2.15.2 Grundsätzliches Vorgehen

1. Definition des Betrachtungsobjekts (Context Definition)

Ausgangspunkt ist die Beschreibung des Systems, seiner Schnittstellen und Kommunikationspfade. Dabei werden relevante Funktionen, beteiligte Steuergeräte, Netzwerke und externe Schnittstellen (z. B. Cloud, Diagnose, App-Verbindungen) identifiziert

2. Identifikation der Assets

Bestimmung der zu schützenden Werte (Assets), wie sicherheitsrelevante Funktionen, Kommunikationsnachrichten, Softwarekomponenten, kryptografische Schlüssel oder personenbezogene Daten

3. Bedrohungsanalyse

Systematische Ermittlung möglicher Angriffsvektoren und Bedrohungen, die auf die identifizierten Assets wirken können. Dabei werden bekannte Angriffsszenarien, Schwachstellen und potenzielle Angreiferprofile berücksichtigt. Die Analyse kann durch Bedrohungskataloge, STRIDE- oder HEAVENS-Ansätze unterstützt werden

4. Risikobewertung

Bewertung der identifizierten Bedrohungen anhand relevanter Kriterien wie

- Angriffsaufwand und technische Machbarkeit
- mögliche Auswirkungen auf Betriebssicherheit, Datenschutz, Verfügbarkeit und Finanzen
- Eintrittswahrscheinlichkeit unter Berücksichtigung der Angreiferfähigkeiten und -motivation
- Das Ergebnis ist eine priorisierte Liste von Risiken
- Entscheidung zum Umgang mit den Risiken (Risikovermeidung, Risikoreduzierung, Teilen oder Akzeptanz des Risikos)

5. Festlegung von Cyber-Sicherheitszielen

Für alle Risiken, die nicht geteilt oder akzeptiert werden, sind übergeordnete Cyber-Sicherheitsziele zu formulieren, um das Risiko auf ein akzeptables Maß reduzieren zu können

6. Ableitung technischer Sicherheitsanforderungen

Aus den Cyber-Sicherheitszielen werden konkrete technische Anforderungen für Architektur, Implementierung und Betrieb des Systems abgeleitet (z. B. Authentifizierung, Verschlüsselung, Zugriffskontrolle)

7. Validierung und Rückverfolgbarkeit

Sicherstellung, dass alle Risiken mit entsprechenden Maßnahmen adressiert und die abgeleiteten Anforderungen in der System- und Softwareentwicklung nachverfolgbar umgesetzt sind

8. Kontinuierliche Aktualisierung

Die TARA ist kein einmaliger Prozess. Neue Bedrohungen, Schwachstellen oder Systemänderungen erfordern eine laufende

Überprüfung und Anpassung der Analyse, um dauerhaft ein akzeptables Restrisiko sicherzustellen. Insbesondere bei Softwareupdates, Architekturänderungen oder Einführung neuer Kommunikationsschnittstellen muss die TARA überprüft werden

5.2.15.3 Hinweise/Besonderheiten/Unterschiede beim Einsatz im Softwarekontext

Die TARA-Methode weist einen klaren Fokus auf softwarebezogene Bedrohungsszenarien auf und ist eng mit sicherheitsrelevanten Entwicklungspraktiken wie Secure Coding, dem Vulnerability Management sowie den Patch- und Update-Prozessen verknüpft. Ihre Anwendung im Softwarekontext ermöglicht eine gezielte Bewertung von Risiken, die speziell durch Schwachstellen in digitalen Komponenten entstehen, und unterstützt die Ableitung technischer und organisatorischer Maßnahmen zur Absicherung softwareintensiver Systeme.

5.2.15.4 Voraussetzungen (Input)

Grundlage für die Durchführung einer TARA ist eine vollständige Item Definition, die das betrachtete System oder die Funktion beschreibt. Darüber hinaus sollten bereits eine HARA (Hazard Analysis and Risk Assessment) sowie eine Bewertung hinsichtlich Datenschutzaspekten (Data Privacy Evaluation) vorliegen, um eine fundierte und kontextbezogene Risikoanalyse zu ermöglichen.

5.2.15.5 Anwendungstiefe und Beendigungskriterien

Für alle identifizierten Risiken, die nicht als akzeptabel eingestuft werden können, werden Cyber-Sicherheitsziele sowie – sofern erforderlich – technische Sicherheitsanforderungen abgeleitet. Da Bedrohungsszenarien im digitalen Umfeld dynamisch entstehen und sich kontinuierlich verändern können, wird die TARA als eine „living analysis“ verstanden. Ihre Anwendung erfolgt daher fortlaufend und iterativ, wobei eine vollständige Beendigung nicht vorgesehen ist. Stattdessen orientiert sich der Abschluss einzelner Analysezyklen an den jeweiligen Projektphasen.

5.2.15.6 Anforderungen an Aufwand, Team und Tools

Die TARA wird häufig in tabellarischer Form dokumentiert. In einigen Fällen kommen spezialisierte Analyse-Tools zum Einsatz, die durch grafische Darstellungen – etwa in Form von Angriffsbäumen – die systematische Bewer-

tung unterstützen. Der Aufwand für die Durchführung der TARA hängt maßgeblich von der Komplexität des betrachteten Systems sowie von der Verfügbarkeit und Qualität bestehender Bedrohungskataloge ab. Insbesondere die Identifikation neuer, bislang nicht dokumentierter Risiken kann den Analyseaufwand deutlich erhöhen. Für eine fundierte Durchführung ist die enge Zusammenarbeit mit Expert:innen aus der System- und Softwareentwicklung sowie aus dem Testbereich erforderlich.

5.2.15.7 Nutzen im Softwarekontext (Vorteile)

Die TARA-Methode ermöglicht eine strukturierte Schutzbedarfsanalyse über verschiedene digitale Asset-Typen hinweg. Dazu zählen funktionale Aspekte wie sicherheitsrelevante Systemfunktionen (z. B. Airbag-Aktivierung), kritische Nachrichten wie Warnanzeigen oder Steuerbefehle (z. B. Parkbremse aktivieren), sensible Daten wie persönliche Nutzerdaten oder Zugangsinformationen sowie softwarebezogene Komponenten einschließlich kryptografischer Schlüssel. Durch diese breite Anwendbarkeit unterstützt TARA die gezielte Identifikation und Bewertung von Risiken in softwareintensiven Systemen und schafft die Grundlage für wirksame Schutzmaßnahmen.

5.2.15.8 Einschränkungen der Methode im Softwarekontext (Nachteile)

Die Qualität der TARA-Ergebnisse hängt stark von der Erfahrung und Expertise des durchführenden Teams sowie von der Vollständigkeit und Aktualität des verwendeten Bedrohungskatalogs ab. Ohne eine gezielte Fokussierung auf realistische oder besonders kritische Szenarien besteht die Gefahr, dass die Analyse unnötig aufgebläht oder unübersichtlich wird. Eine klare Priorisierung und Strukturierung ist daher essenziell, um die Aussagekraft und Effizienz der Methode zu gewährleisten.

5.2.15.9 Ergebnis (Output)

Für alle Risiken, die im Rahmen der TARA als nicht akzeptabel eingestuft werden, müssen spezifische Cyber-Sicherheitsziele definiert werden. Aus diesen Zielen lassen sich anschließend technische Sicherheitsanforderungen ableiten, die zur Risikobehandlung beitragen. Die Ergebnisse der TARA können direkte Auswirkungen auf die funktionale Sicherheit haben und sollten daher mit der HARA abgestimmt werden. Fehlende Gefährdungen, die sich aus der Cybersecurity-Perspektive ergeben, sind gegebenenfalls in der

HARA zu ergänzen. Beide Methoden sollten eng verzahnt betrachtet werden, da Sicherheitslücken – etwa durch gezielte Manipulationen – auch funktionale Gefährdungen nach sich ziehen können, wie beispielsweise die ungewollte Aktivierung oder Deaktivierung sicherheitskritischer Fahrzeugfunktionen.

5.2.15.10 Referenzen/Weiterführende Literatur

International Organization for Standardization. (2019). *IEC/ISO 31010:2019 – Risk Management – Risk Assessment Techniques*. ISO.

6 Risikoanalyse entlang der Integrationskette

6.1 Einleitung

Risikoanalysen können parallel entlang der Integrationskette (aktuell laufendes Projekt) oder sequenziell für Folgeprojekte erfolgen.

Angesichts der zunehmenden Vernetzung von Lieferketten und der steigenden Komplexität sowie der Wiederverwendung von Softwaremodulen oder ganzen Systemen sind folgende Aspekte zu berücksichtigen:

- Integration von Risikoanalysen in laufende Projekte
- Übernahme in neue Projekte
- Weitergabe zwischen verschiedenen Unternehmen der Lieferkette

6.2 Zielsetzung

Ziel ist die Ableitung von Handlungsempfehlungen, wie bereits identifizierte Risiken systematisch bewertet und anschließend in angrenzende Risikoanalysen integriert werden können.

6.3 Motivation und Nutzen

- Effizienzsteigerung: Reduzierung von Aufwand und Kosten durch Vermeidung redundanter Analysen bei ähnlichen Funktionen oder Systemen
- Qualitätsverbesserung: Nutzung bewährter Analysen und Erkenntnisse aus existierenden Projekten, für eine frühzeitige, umfassende Risikoidentifikation und -bewertung
- Beschleunigung der Entwicklung: kürzere Entwicklungszeiten durch die Verfügbarkeit von Voranalysen, die eine gezielte Priorisierung ermöglichen und Entwicklungszyklen effizient verkürzen
- Konsistenz: Sicherstellung einer durchgängigen Betrachtung von Risiken über Projekt- und Unternehmensgrenzen hinweg, insbesondere bei Standardfunktionen

6.4 Herausforderungen

- **Kontextabhängigkeit:** Die klassifizierten Risiken und die Wirksamkeit von Maßnahmen sind stark vom spezifischen Projektkontext (Hardware, Umgebungsbedingungen, Gesamtarchitektur, Anwendungsfall, gesetzliche/normative Anforderungen) abhängig
- **Unterschiedliche Analysemethoden und -tiefen:** Verschiedene Unternehmen oder Projekte können unterschiedliche Methoden, Tools, Bewertungskriterien und Detaillierungsgrade für Risikoanalysen verwenden
- **Datenqualität und -aktualität:** Die Qualität, Vollständigkeit und Aktualität der Risikoanalyse ist für den spezifischen Projektkontext zu gewährleisten
- **Informationsbasis:** Die zu übernehmende Risikoanalyse muss in einem vereinbarten Format und mit ausreichendem Detaillierungsgrad bereitgestellt werden
- **Schnittstellen und Abhängigkeiten:** Identifikation der Schnittstellen und Abhängigkeiten der Softwarefunktionalität im (neuen) Systemkontext
- **Vertraulichkeit und IP-Schutz:** Informationen in Risikoanalysen können dem Schutz geistigen Eigentums unterliegen, was den Austausch erschwert
- **Tool-Kompatibilität:** Mangelnde Kompatibilität von Risikoanalyse-Tools zwischen den beteiligten Parteien kann den Datenaustausch behindern
- **Rückverfolgbarkeit (Traceability):** Die lückenlose Nachverfolgbarkeit von Anpassungen und Ergänzungen an der Risikoanalyse ist sicherzustellen

6.5 Übernahme von Risikoanalysen

Risikoanalysen können in laufende/neue Projekte integriert oder zwischen verschiedenen Unternehmen der Lieferkette weitergegeben werden. Hierfür werden folgende Phasen empfohlen:

Phase 1: Anforderung und Prüfung der Eingangsdokumentation

- **Spezifikation und Anforderung:** Spezifikation der Anforderungen an die bereitzustellenden „Risikoanalyse-Ergebnisse“, z. B.:

- Definition des Zwecks und Umfangs der zu übernehmenden Risikoanalyse
 - Abstimmung von Format, Struktur und Bewertungsmethodik
 - Festlegung der Schnittstellen, Ansprechpartner und Kommunikationswege sowie der Verantwortlichkeiten für die Risiken
 - Planung von Review-Terminen und Freigabeprozessen
 - Liste der getroffenen Annahmen („Assumptions of Use“)
 - Ergebnisse der Analyse (z. B. identifizierte Risiken, abgeleitete Sicherheitsanforderungen)
- **Formale und methodische Prüfung (Plausibilitäts-Check)**
Das Ziel ist es, sicherzustellen, dass die Risikoanalyse vollständig und formal korrekt übergeben wurde. Aktivitäten hierbei sind:
 - Eingangsprüfung der gelieferten Unterlagen (Vollständigkeit, Version, Freigabestatus)
 - Prüfung auf vereinbartes Format und Detaillierungsgrad
 - Beschreibung der Systemgrenzen und Schnittstellen
 - Konsistenz: Ist die Dokumentation in sich widerspruchsfrei?
 - Methodenkonformität: Wurde die vereinbarte Methode (z. B. FMEA nach AIAG & VDA FMEA Handbuch) korrekt angewendet?
 - Nachvollziehbarkeit: Ist die Herleitung von Ursachen, Folgen und Maßnahmen logisch dokumentiert?

Hinweis: Bei Abweichungen sollten Klärungsschleifen mit den Integrationspartnern erfolgen

Phase 2: Technische Bewertung der gelieferten Analyse im Projektkontext

- Der Detaillierungsgrad der Prüfung und Übernahme orientiert sich an der Kritikalität der Komponente (z. B. ASIL-Einstufung), der Neuheit der Technologie und dem Reifegrad (z. B. Assessments nach Automotive SPICE®)

- Kontextabgleich (System-Ebene): Abgleich u. a. von Funktionsumfang, Architektur, Schnittstellen, Umgebungsbedingungen sowie Anwendungsfällen und gesetzlichen und normativen Anforderungen
 - *Beispiel für Umgebungsbedingungen: Ein Sensor liefert stabile Ergebnisse bis zur maximalen Betriebstemperatur von 85 °C. Im Fahrzeug wird der Sensor jedoch in einem Bereich verbaut, wo bis zu 100 °C auftreten können. Diese Diskrepanz („Gap“) muss als neues oder modifiziertes Risiko bewertet werden*
 - *Beispiel für Schnittstellen: Ein Steuergerät sendet Daten. Die Risikoanalyse des Lieferanten deckt den Ausfall des Sendevorgangs ab. Der Auftraggeber muss jedoch analysieren, was passiert, wenn Steuergeräte von anderen Herstellern auf dem Bus fehlerhafte Daten in die Integrationskette senden und eine Komponente diese empfängt*

Phase 3: Integration, Konsolidierung und Rückkopplung

Diese Phase ist in drei Schritte unterteilt:

1. Integration der Risikoanalysen

- Übernahme validierter Risiken: Als valide eingestufte Risiken werden mit Rückverfolgbarkeit (Traceability) in die Gesamtrisikoaanalyse des Auftraggebers übernommen
- Behandlung von Delta-Risiken: Neue und modifizierte Risiken aus der Delta-Analyse werden in die Gesamtanalyse integriert und bewertet

2. Maßnahmenableitung und Rückkopplung: Aus der konsolidierten Analyse werden Maßnahmen abgeleitet. Neue Anforderungen, die den Lieferumfang betreffen, werden an den Lieferanten zurückgespielt (z. B. „Sensor muss bis 100 °C spezifiziert werden“), was u. a. zu Vertragsanpassungen führen kann

3. Dokumentation der Übernahme: Der gesamte Prozess wird als Nachweis (z. B. für einen Safety Case, ein Assessment/Audit) in einem Integrationsbericht dokumentiert

Phase 4: Verifizierung und Validierung

Die konsolidierte Risikoanalyse und die daraus abgeleiteten Maßnahmen müssen verifiziert und validiert werden.

Die formale und inhaltliche Freigabe (u. a. durch Reviews) der konsolidierten Risikoanalyse und der zugehörigen Maßnahmen erfolgt im Zielprojekt.

6.6 Handlungsempfehlungen für die Praxis

- Frühzeitige Einbindung: frühzeitige Übernahme der Risikoanalyse in der Projektplanung sowie Berücksichtigung in Vertragsverhandlungen
- Informationstransfer: Training und Austausch zwischen den Teams, um ein gemeinsames Verständnis der Analysen und Kontexte zu schaffen inkl. Lessons Learned
- Iterativer Prozess: Übernahme mit Feedback-Schleifen zwischen den Beteiligten der Integrationskette
- Standards: Prüfung, ob geeignete bestehende Standards für die Risikoanalyse genutzt werden können, bevor neue entwickelt werden

7 Annex – Fallbeispiele

Die in diesem Band enthaltenen Fallbeispiele dienen ausschließlich der Veranschaulichung und Orientierung. Sie stellen keine verbindlichen Handlungsempfehlungen oder normativen Vorgaben dar. Der Verband der Automobilindustrie e. V. (VDA) übernimmt keine Gewähr für die Aktualität, Richtigkeit, Vollständigkeit oder Anwendbarkeit der dargestellten Inhalte in konkreten betrieblichen Situationen.

Die Verantwortung für die Bewertung, Auswahl und Umsetzung geeigneter Maßnahmen liegt ausschließlich bei den Anwender:innen. Eine Haftung des VDA für etwaige Schäden, die aus der Nutzung oder Nichtnutzung der Fallbeispiele entstehen, ist ausgeschlossen.

7.1 Praxisbeispiele für die Anwendung der Methoden

7.1.1 ATAM – Spurhalteassistent

System: Spurhalteassistent

Der Spurhalteassistent ist eine sicherheitsrelevante Softwarefunktion in modernen Fahrerassistenzsystemen. Er erkennt Fahrbahnmarkierungen mithilfe einer Frontkamera und greift bei unbeabsichtigtem Verlassen der Fahrspur unterstützend in die Lenkung ein oder warnt die Fahrer:innen. Die Funktion steht in enger Wechselwirkung mit anderen Systemen wie Lenksteuergerät, Kamerasystem, Brems- und Stabilitätssteuerung.

Phase 0

Für die Bewertung des Spurhalteassistenten wird ein interdisziplinäres Team aus Softwarearchitekt:innen, Entwickler:innen, Sicherheitsexpert:innen und Vertreter:innen des Qualitätsmanagements gebildet. Die relevanten Architekturunterlagen werden geprüft und Workshops terminiert.

Phase 1

- Schritt 1: Erläuterung des Vorgehens der ATAM
- Schritt 2: Business Goals und wichtigste Systemfunktionen werden identifiziert
 - Unfallvermeidung durch Spurhalteunterstützung
 - Komfortsteigerung (sanfte Lenkeingriffe, sprachliche und visuelle Warnungen)
 - Erfüllung rechtlicher Anforderungen

- Schritt 3: Architektur wird vorgestellt
- Schritt 4: Architekturansätze werden identifiziert
 - Layered Architecture: Trennung in Sensor-, Logik- und Ak-
tor-Schichten
 - Pipes-and-Filters Pattern: schrittweise Filterung von Ka-
mera- und Sensordaten
 - Feedback Control Loop: geschlossener Regelkreis zur
Lenkkorrektur
 - Event-driven Architecture: ereignisbasierte Reaktion auf
Spurabweichungen
 - Fault Containment Pattern: isolierte Fehlerbehandlung in
Subsystemen
 - Watchdog Pattern: Überwachung kritischer Komponenten
mit automatischer Fehlererkennung
 - Observer Pattern: Beobachtung und Benachrichtigung bei
Systemzustandsänderungen
- Schritt 5: Utility Tree erstellen (Qualitätsattribute und deren Szenarien werden definiert und priorisiert)

In Schritt 5 wird ein Utility Tree erstellt, der die wichtigsten Qualitätsattribute und ihre Szenarien strukturiert darstellt. Diese Szenarien werden anschließend gemeinsam mit den Stakeholdern nach Wichtigkeit (Importance) und Schwierigkeit bzw. Risiko (Difficulty/Risk) auf einer High–Medium–Low-(H/M/L-)Skala bewertet. So werden die kritischsten Szenarien mit hoher Wichtigkeit und hoher Schwierigkeit identifiziert und für die weitere Analyse priorisiert.

Tabelle 7-1: Utility Tree

Sicherheit	Reaktionszeit Lenkeingriff < 50 ms Sanfte Lenkkorrektur ohne Übersteuerung < 0,5 m/s ² Warnung bei Kameraausfall innerhalb 100 ms
Performanz	Bilddatenverarbeitung < 50 ms Latenz Verarbeitungsrate > 20 Frames/Sekunde
...	...

- Schritt 6: Für jedes hochrangige Szenario werden Risiken und Trade-offs analysiert

Beispiel: Warnung bei Kameraausfall innerhalb 100 ms

- Architekturansatz: Fault Containment Pattern mit Watchdog-Monitoring
- Analyse: Ein Watchdog überwacht kontinuierlich den Datenstrom der Kamera. Bei Ausfall wird sofort ein Fallback-Modus aktiviert, der die Fahrer:innen warnt, ohne einzugreifen
- Risiko: Falsch-positive Ausfallerkennungen könnten zu unnötigen Systemdeaktivierungen führen und das Vertrauen der Fahrer:innen in das System beeinträchtigen
- Trade-off: Robustheit vs. Verfügbarkeit – strengere Überwachung erhöht Sicherheit, kann aber zu häufigeren Fehlalarmen führen
- Sensitivitätspunkt: Timeout-Schwellwerte des Watchdogs
- Empfehlung: mehrstufiges Monitoring mit Plausibilitätsprüfung über mehrere Frames

Phase 2

Hier werden Schritt 5 und 6 nochmals mit weiteren Stakeholdern durchgeführt, um weitere Perspektiven auf die Architektur einzubeziehen. Daraufhin werden die ermittelten Risiken zusammengefasst und den Business Goals gegenübergestellt, um den Bezug nochmals herzustellen.

Phase 3

Risiken, Trade-offs und Empfehlungen werden zusammengefasst.

Conclusio

Die ATAM-Analyse des Spurhalteassistenten macht deutlich, dass die Kombination aus Event-driven Architecture, Pipes-and-Filters, Feedback Control Loop, Layered Architecture sowie Fault Containment mit Watchdog-Monitoring zentrale Qualitätsziele erfüllt, aber auch spezifische Risiken und Zielkonflikte erzeugt. Anpassungen wie Priorisierung sicherheitskritischer Events, adaptive Filterpipelines und mehrstufiges Monitoring reduzieren Latenzen, erhöhen Genauigkeit und minimieren Fehlalarme. Die Ergebnisse liefern eine klare Grundlage für effektive Architektur- und Softwareverbesserungen sowie Tests im sicherheitskritischen Umfeld.

Hinweis: Für Baustellen-Situationen wird eine ergänzende technische Risikoanalyse mit SOTIF empfohlen.

7.1.2 ETA – OTA-Update

Szenario: OTA-Update für ein sicherheitskritisches Steuergerät

Ziel ist die Analyse der möglichen Ereignisse, die nach dem Start eines OTA-Updates für ein sicherheitskritisches Steuergerät – beispielsweise ein Bremssystem – auftreten können.

Ausgangsereignis: OTA-Update wird gestartet.

Ereignisbaum

1. **Update-Datei wird korrekt heruntergeladen?**
 - **Ja** → Weiter zu Schritt 2
 - **Nein** → **Folge:** Update abgebrochen, Fahrzeug bleibt im alten Softwarestand (Auswirkung: keine neue Funktion, aber sicher)
2. **Integritätsprüfung (Checksumme) erfolgreich?**
 - **Ja** → Weiter zu Schritt 3
 - **Nein** → **Folge:** Update wird verworfen, Fahrzeug bleibt im alten Softwarestand (Auswirkung: sicher, aber kein Update)
3. **Installation erfolgreich?**
 - **Ja** → Weiter zu Schritt 4
 - **Nein** → **Folge:** Steuergerät möglicherweise in inkonsistentem Zustand → **Risiko: Fahrzeug nicht fahrbereit**
4. **Neustart des Steuergeräts erfolgreich?**
 - **Ja** → Update abgeschlossen, Fahrzeug funktionsfähig
 - **Nein** → Steuergerät fällt aus → **Risiko: sicherheitskritischer Ausfall (z. B. Bremsen)**

Ereignisbaum (Textdiagramm)

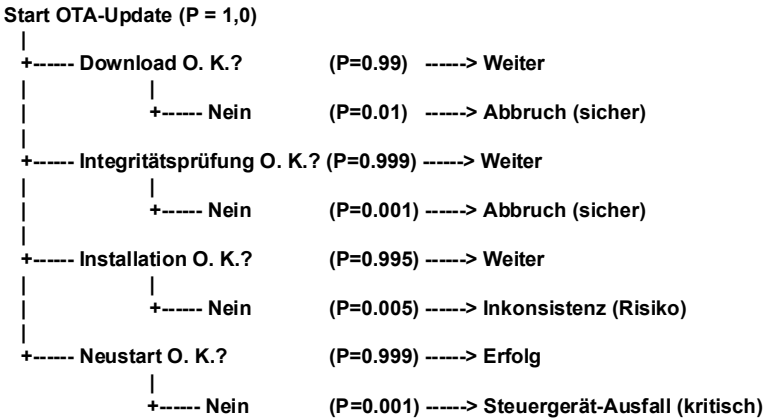


Abbildung 7-1: Ereignisbaum (Textdiagramm)

Wahrscheinlichkeiten und Konsequenzen

Jedem möglichen Ereignis innerhalb eines Entscheidungsbaums kann eine Eintrittswahrscheinlichkeit zugeordnet werden – beispielsweise 0,01 für einen Download-Fehler oder 0,001 für einen Integritätsfehler. Die daraus resultierenden Konsequenzen werden entsprechend ihren Auswirkungen bewertet, etwa mit den Kategorien „kritisch“, „sicher“ oder „funktional eingeschränkt“.

Tabelle 7-2 ETA-Tabelle: OTA-Update für Steuergerät

Schritt / Ereignis	Bedingung	Wahrscheinlichkeit	Konsequenz
Start OTA-Update	-	1,0	Ausgangspunkt
Download erfolgreich?	Ja	0,99	Weiter
	Nein	0,01	Abbruch (sicher)
Integritätsprüfung O. K.?	Ja	0,999	Weiter
	Nein	0,001	Abbruch (sicher)
Installation erfolgreich?	Ja	0,995	Weiter
	Nein	0,005	Inkonsistenz (Risiko: Fahrzeug nicht fahrbereit)
Neustart erfolgreich?	Ja	0,999	Erfolg
	Nein	0,001	Steuergerät-Ausfall (kritisch)

Berechnung der Gesamtwahrscheinlichkeiten

- **Erfolgreiches Update:**

$$P=0.99 \times 0.999 \times 0.995 \times 0.999 \approx 0.983$$
$$P = 0.99 \times 0.999 \times 0.995 \times 0.999 \approx 0.983$$

→ **98,3 %**

- **Kritischer Ausfall (Neustart schlägt fehl):**

$$P=0.99 \times 0.999 \times 0.995 \times 0.001 \approx 0.00099$$
$$P = 0.99 \times 0.999 \times 0.995 \times 0.001 \approx 0.00099$$

→ **0,099 %**

- **Inkonsistenz nach Installation:**

$$P=0.99 \times 0.999 \times 0.005 \approx 0.00495$$
$$P = 0.99 \times 0.999 \times 0.005 \approx 0.00495$$

→ **0,495 %**

Conclusio

Die Ereignisbaumanalyse (ETA) für das OTA-Update eines sicherheitskritischen Steuergeräts zeigt, dass der Update-Prozess insgesamt eine sehr hohe Erfolgswahrscheinlichkeit von 98,3 % aufweist. Die Analyse identifiziert jedoch auch potenzielle Risiken: Ein kritischer Ausfall des Steuergeräts nach dem Neustart ist selten (0,099 %), kann aber gravierende sicherheitsrelevante Folgen haben. Ebenso besteht ein mittleres Risiko für Inkonsistenzen nach der Installation (0,495 %), die die Fahrzeugverfügbarkeit beeinträchtigen.

Die ETA verdeutlicht, dass die größten Risiken nicht im Download oder der Integritätsprüfung liegen, sondern in den letzten Schritten der Installation und des Neustarts. Daraus ergibt sich die Notwendigkeit, Diagnosemechanismen und Fallback-Strategien insbesondere für diese Phasen zu stärken.

Insgesamt liefert die ETA eine transparente Grundlage für die Priorisierung von Maßnahmen zur Risikominimierung und unterstützt die Argumentation in Sicherheits- und Qualitätsaudits.

7.1.3 CPA – Verkehrszeichenerkennung

Die Verkehrszeichenerkennung ist eine zentrale Softwarefunktionalität in modernen Fahrerassistenzsystemen und automatisierten Fahrfunktionen. Sie dient der Erkennung und Interpretation von Verkehrszeichen wie Geschwindigkeitsbegrenzungen, Überholverbote oder temporären Baustellenhinweisen. Die korrekte Verarbeitung dieser Informationen ist essenziell für die Einhaltung gesetzlicher Vorgaben und die Sicherheit im Straßenverkehr.

Die vorliegende CPA-Vorlage (Criticality and Prioritization Analysis, CPA) ermöglicht eine strukturierte Bewertung dieser Funktionalität. Diese erfolgt qualitativ anhand von Experteneinschätzungen und kann durch quantitative Methoden ergänzt werden. Die Ergebnisse dienen als Grundlage für Architekturentscheidungen, Testplanung und weiterführende Analysen wie FMEA oder Threat Modeling.

CPA-Kriterium	Risikoidentifikation	Kritikalität	Priorität der Absicherung
Softwarefunktionalität	Verkehrszeichenerkennung	Hoch	Hoch
Systemabhängigkeit	Geschwindigkeitsassistent, Navigationssystem	Mittel	Hoch
Nutzungskontext	Autobahn, Stadt, Landstraße, Baustellen	Hoch	Hoch
Kritikalität	Fehlinterpretation kann zu Geschwindigkeitsverstößen führen	Hoch	Hoch
Robustheitsanforderung	Erkennung bei schlechten Lichtverhältnissen, Verschmutzung, Wetter	Hoch	Hoch
Cybersecurity-Relevanz	Manipulation von Kameradaten oder Software möglich	Mittel	Mittel

Redundanz/ Absicherung	Kombination mit Kartendaten, V2X-Kommunikation	Mittel	Hoch
Fehlertoleranz/ Recovery	Warnung an Fahrer:in, temporäre Deaktivierung der Funktion	Mittel	Mittel
Testbedarf/ Szenarien	Erkennung temporärer Zeichen, Baustellen, Nachtfahrten	Hoch	Hoch
Normbezug (optional)	ISO 26262 ASIL B/C, UNECE R155	Mittel	Hoch
Bewertungsskala (optional)	Kritikalität: hoch / Priorität: hoch	Hoch	Hoch

Conclusio

Die Bewertung der Verkehrszeichenerkennung zeigt eine hohe Kritikalität und Priorität in nahezu allen relevanten Bereichen. Besonders die Auswirkungen bei Fehlinterpretationen und die Anforderungen an Robustheit und Testabdeckung unterstreichen die sicherheitsrelevante Bedeutung dieser Funktion. Es wird empfohlen, die Absicherung durch redundante Informationsquellen wie Kartendaten und V2X-Kommunikation zu stärken und gezielte Testszenarien für temporäre und schwer erkennbare Verkehrszeichen zu definieren. Die Ergebnisse dieser CPA-Analyse sollten in die Sicherheitsarchitektur und Testplanung einfließen.

7.1.4 HAZOP – Türöffner-App

Die Mobile-App als Türöffner ermöglicht es den Nutzer:innen, das Fahrzeug per Smartphone zu öffnen und zu schließen. Über eine sichere Verbindung (z. B. Bluetooth, NFC oder Mobilfunk) sendet die App ein Öffnungssignal an das Fahrzeug, nachdem die Authentifizierung der Nutzer:innen erfolgt ist. Die App ersetzt damit den klassischen Fahrzeugschlüssel und bietet zusätzliche Komfortfunktionen wie Statusanzeige, Fernsteuerung und ggf. Protokollierung der Zugriffe.

Die Bewertung erfolgt qualitativ anhand von Experteneinschätzungen. Die Ergebnisse dienen als Grundlage für Architekturentscheidungen, Testplanung und weiterführende Analysen wie FMEA oder Threat Modeling.

System: Mobilephone-to-Car-App, Übertragung des Öffnungssignals

Leitwort	Abweichung	Ursache	Folge (Konsequenz)	Maßnahme
Kein	Kein Öffnungssignal gesendet	App-Absturz, Netzwerkfehler	Fahrzeug bleibt verschlossen	App-Stabilität, Verbindung prüfen
Mehr	Mehrfaches Öffnungssignal	Doppelklick, Softwarefehler	Tür öffnet mehrmals, Fehlfunktion	Signal-Entprellung, Logging
Weniger	Öffnungssignal zu schwach	Bluetooth gestört, Reichweite	Fahrzeug reagiert nicht	Signalstärke prüfen, alternative Wege
Falsch	Falsches Signal gesendet	Softwarebug, Manipulation	Unberechtigtes Öffnen, Sicherheitsrisiko	Authentifizierung, Verschlüsselung
Verzögert	Öffnungssignal kommt verspätet	Netzwerklatenz, App-Performance	Tür öffnet mit Verzögerung	Timeout, Nutzerhinweis

Conclusio

Die HAZOP-Analyse der Mobile-App als Türöffner hat gezeigt, dass verschiedene Abweichungen und Fehlerquellen – wie Verbindungsprobleme, fehlerhafte Authentifizierung oder Softwarefehler – die Funktion und Sicherheit der Fahrzeugöffnung beeinträchtigen können. Durch die systematische

Betrachtung von Leitworten und Szenarien konnten relevante Risiken identifiziert und geeignete Maßnahmen zur Risikominderung abgeleitet werden. Die Analyse unterstreicht die Bedeutung von stabiler Kommunikation, zuverlässiger Authentifizierung und robuster Fehlerbehandlung, um sowohl die Benutzerfreundlichkeit als auch die Sicherheit der App zu gewährleisten.

7.2 Methodenauswahl in einem Safety-relevanten Projekt

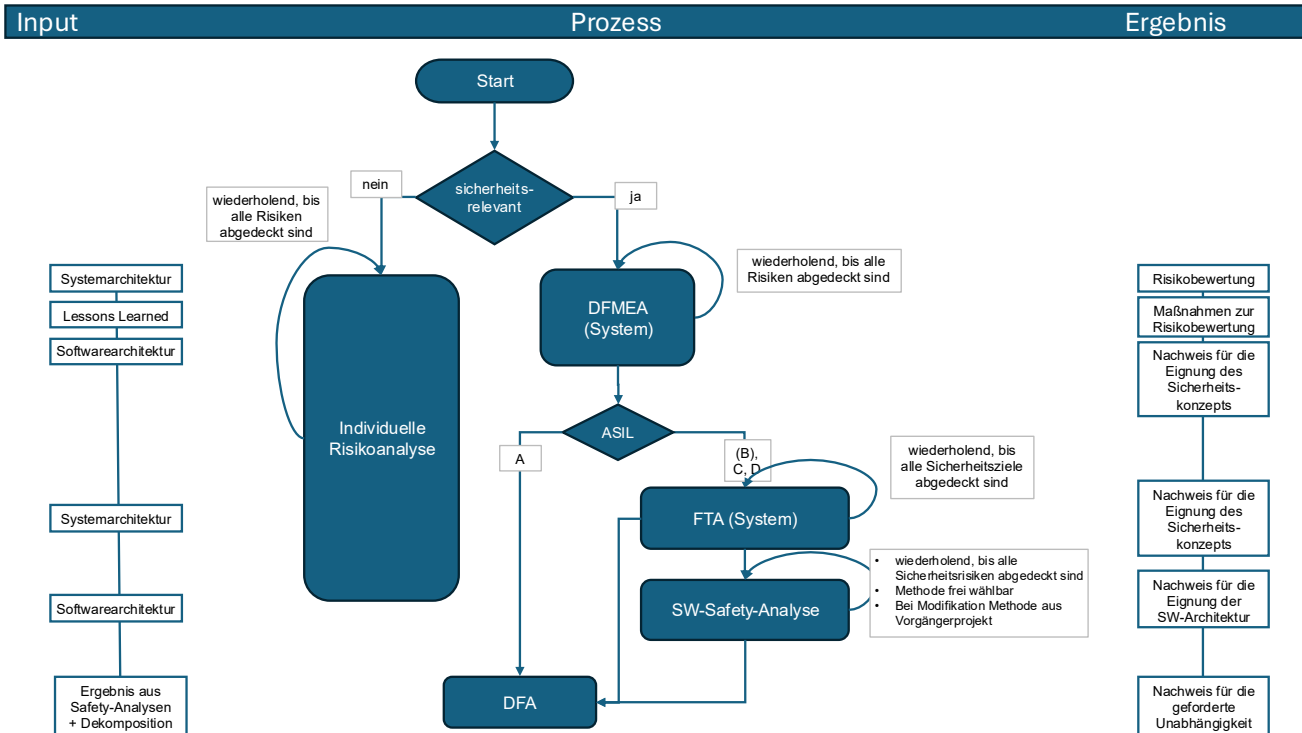


Abbildung 7-2: Prozessablauf einer Methodenauswahl in einem Safety-relevanten Projekt

Der dargestellte Prozessablauf gliedert sich in die drei Hauptbereiche Input, Prozess und Ergebnis und beschreibt die strukturierte Vorgehensweise zur Risikoanalyse und Sicherheitsbewertung im Rahmen der Systementwicklung.

Der Ablauf beginnt mit dem Startpunkt, an dem geprüft wird, ob das betrachtete System oder Element sicherheitsrelevant ist. Falls dies nicht der Fall ist, erfolgt eine individuelle Risikoanalyse (siehe Kapitel 5), die iterativ durchlaufen wird, bis alle Risiken angemessen bewertet und adressiert sind.

Ist das System sicherheitsrelevant, wird eine DFMEA (Design Failure Mode and Effects Analysis) auf Systemebene durchgeführt. Diese Analyse dient der Identifikation potenzieller Fehlerursachen und deren Auswirkungen im Gesamtsystem.

Auch bei nicht sicherheitsrelevanten Risiken wird die DFMEA typischerweise auf Systemebene durchgeführt. Risiken aus softwarebasierten Funktionalitäten könnten mithilfe der DFMEA analysiert und entsprechende Entdeckungs- oder Vermeidungsmaßnahmen abgeleitet werden.

Je nach Einstufung des Automotive Safety Integrity Level (ASIL) sind gemäß ISO 26262 weiterführende Risikoanalysen erforderlich – z. B. bei höheren ASIL-Stufen (B, C oder D):

- eine FTA (Fault Tree Analysis) auf Systemebene zur strukturierten Untersuchung von Fehlerursachen
- eine SW-Safety-Analyse, die sich auf die sicherheitsrelevanten Aspekte der Softwarearchitektur konzentriert

Anmerkung: Die Hardware-Sicherheitsanalysen sind nicht aufgelistet.

Sind alle sicherheitsrelevanten Maßnahmen erfolgreich umgesetzt, liefern die Ergebnisse der DFMEA, FTA und SW-Safety-Analyse den Nachweis für die Eignung des Sicherheitskonzepts bzw. der Softwarearchitektur. Die abschließende DFA – unabhängig ASIL – soll sicherstellen, dass die geforderte Unabhängigkeit gemäß Sicherheitsanforderungen gegeben ist.

Anmerkungen zur Methodenanwendung und Optimierung:

Fehlermodes sollten sinnvoll geclustert werden, z. B. mithilfe von Guided Words.





- Beispiel: Die Unterscheidung zwischen Fahrer-, Beifahrer- und Seitenairbag ist funktional unterschiedlich, jedoch ist der Signalfluss vergleichbar. Daher können die Fehlermodes zusammengeführt werden

- Vergleichbare Fehlerarten wie Datenüberschreibung sollten ebenfalls gruppiert werden, um Redundanzen zu vermeiden

Lessons Learned aus der aktuellen Nutzungsphase (insbesondere aus der DFMEA) sollten aktiv in neue Projekte einfließen. Aus diesem Grund ist auch eine methodische Zuordnung zur Nutzungsphase sinnvoll und notwendig.

8 Annex – Anwendbarkeit der Methoden (Übersicht)

In dieser Übersicht werden neben den in diesem Band aufgeführten Methoden (erkennbar an dem Verweis auf vorhergehende Kapitel) weitere Risikoanalysemethoden und ihre Eignung aufgezeigt.

Methode (Standard)	Anwendbarkeit für technische Risikoanalyse von softwarebasierten Funktionalitäten	Norm/Referenz	Beschreibung
ATAM (Architecture Tradeoff Analysis Method) Kapitel 5.2.1	 Geeignet	SEI	Bewertet Architekturentscheidungen hinsichtlich Qualitäts- und Sicherheitsrisiken; für Softwarearchitekturen nützlich, aber keine klassische Risikoanalyse.
CCA (Common Cause Analysis) Kapitel 5.2.2	 Eingeschränkt anwendbar	ISO 26262-5/6, IEC 60812	Identifikation gemeinsamer Ursachen für Fehler in redundanten Systemen; z. B. gemeinsame Logikfehler über Module.
CPA (Criticality and Priority Assessment) Kapitel 5.2.3	 Geeignet	NIST Interagency Report 8179	Strukturierte Methode zur Bewertung von Softwarefunktionalitäten hinsichtlich ihrer Kritikalität im Gesamtsystem.
CPA (Critical Path Analysis)	 Nicht geeignet	PMBOK, DIN 69901	Projektmanagement-Methode zur Terminplanung, nicht für Risikoanalyse von Software.

Methode (Standard)	Anwendbarkeit für technische Risikoanalyse von softwarebasierten Funktionalitäten	Norm/Referenz	Beschreibung
DFA (Design Failure Analysis / Design for Assembly)	♦ Eingeschränkt anwendbar	AIAG / VDA	Ursprünglich für mechanische Bauteile; für softwarebasierte Funktionalitäten nur indirekt anwendbar, z. B. bei modularer Softwarearchitektur oder Schnittstellenintegration.
DRBFM (Design Review Based on Failure Mode) Kapitel 5.2.4	✔ Geeignet	AIAG/VDA	Fokus auf Änderungen in Design oder Softwaremodulen, um neue Risiken früh zu identifizieren; sehr nützlich bei Softwareänderungen.
ETA (Event Tree Analysis) Kapitel 5.2.5	♦ Eingeschränkt anwendbar	ISO 26262-5, IEC 61511	Deduktive Methode zur Analyse von Ereignisfolgen; für Software in Verbindung mit Sicherheitsmaßnahmen einsetzbar, aber limitiert bei Logikfehlern.
FMEA (Failure Mode and Effects Analysis) Kapitel 5.2.6	✔ Geeignet	AIAG-VDA FMEA, ISO 26262-5/6	Induktive, strukturierte Methode zur systematischen Erfassung von Fehlerarten, Ursachen und Auswirkungen auch für softwarebasierte Funktionalitäten. Software-FMEA als Begriff nicht normiert.

Methode (Standard)	Anwendbarkeit für technische Risikoana- lyse von softwareba- sierten Funktionalitäten	Norm/ Referenz	Beschreibung
FMEA-MSR (Monitoring and System Response) Kapitel 5.2.7	✔ Geeignet	ISO 26262-5	Erweiterung der FMEA für Software- und System-Monitoring, Analyse von Diagnose- und Reaktionsmechanismen zur Risikominimierung.
FMEDA (Failure Modes, Effects and Diagnostic Analysis) Kapitel 5.2.8	♦ Eingeschränkt anwendbar	ISO 26262-5	Quantitative Analyse nur für Hardwarekomponenten, für Software weniger geeignet.
FSA (Functional Safety Assessment)	✘ Nicht geeignet	ISO 26262-2	Audit zur Überprüfung von Sicherheitsprozessen, keine Methode zur Risikoidentifikation.
FTA (Fault Tree Analysis) Kapitel 5.2.9	♦ Eingeschränkt anwendbar	ISO 26262-5, IEC 61025	Deduktive Ursachenanalyse, gut für Ursachenfindung, weniger geeignet für strukturierte Risikoidentifikation in Software.
HARA (Hazard Analysis and Risk Assessment) Kapitel 5.2.10	♦ Eingeschränkt anwendbar	ISO 26262-3	Systematische Identifikation und Bewertung von Gefährdungen auf Systemebene. Software kann Ursache sein, Fokus liegt auf funktionaler Sicherheit.

Methode (Standard)	Anwendbarkeit für technische Risikoanalyse von softwarebasierten Funktionalitäten	Norm/Referenz	Beschreibung
HAZOP (Hazard and Operability Study) Kapitel 5.2.11	✔ Geeignet	IEC 61882	Systematische Risikoanalyse von Prozess- und Funktionsabweichungen; für softwarebasierte Funktionalitäten adaptierbar.
ORA (Operational Risk Assessment)	♦ Eingeschränkt anwendbar	UNECE R156, ISO/SAE 21434	Bewertung von operationellen Risiken, z. B. im Betrieb, teilweise auf Software anwendbar.
PHA (Preliminary Hazard Analysis)	♦ Eingeschränkt anwendbar	ISO 12100, ISO 26262-3	Qualitative, frühe Gefährdungsanalyse, auch für Softwarefunktionen geeignet, um Risiken frühzeitig zu erkennen.
PRA (Process Risk Assessment)	✘ Nicht geeignet	IATF 16949, ISO 9001	Bewertung von Fertigungs- und Prozessrisiken.
Reliability Prediction / Weibull Analysis	✘ Nicht geeignet	IEC 61709, MIL-HDBK-217	Statistische Methoden zur Hardware-Zuverlässigkeitsvorhersage, nicht auf Software anwendbar.
Safety Case / Safety Argumentation	✘ Nicht geeignet als technische Risikoanalyse-Methode	ISO 26262-10, UL 4600	Dokumentation und Nachweis der funktionalen Sicherheit, keine eigenständige Risikoidentifikationsmethode.

Methode (Standard)	Anwendbarkeit für technische Risikoanalyse von softwarebasierten Funktionalitäten	Norm/Referenz	Beschreibung
SOTIF (Safety Of The Intended Functionality) Kapitel 5.2.12	<input checked="" type="checkbox"/> Geeignet	ISO 21448	Analyse der Risiken aus nicht-funktionaler Sicherheit oder Systemverhalten in der vorgesehenen Betriebsweise; insbesondere relevant für autonome Systeme und Software.
SRA (Security Risk Assessment)	<input checked="" type="checkbox"/> Geeignet (Cybersecurity)	UNECE R155, ISO/SAE 21434	Bewertung von Cybersecurity-Risiken in Fahrzeugsoftware und -systemen.
SWIFT (Structured What-If Technique) Kapitel 5.2.13	<input checked="" type="checkbox"/> Geeignet	AIAG / VDA	Qualitative Methode zur Identifikation potenzieller Fehler oder Risiken, auch auf Software anwendbar; eher explorativ.
STPA (System-Theoretic Process Analysis) Kapitel 5.2.14	<input checked="" type="checkbox"/> Geeignet	Leveson, MIT	Sicherheitsanalyse auf Systemebene basierend auf Kontroll- und Feedbacktheorie; kann für Software und Systeme umfassend Risiken identifizieren.
TARA (Threat Analysis and Risk Assessment) Kapitel 5.2.15	<input checked="" type="checkbox"/> Geeignet (Cybersecurity)	ISO/SAE 21434	Strukturierte Identifikation und Bewertung von Bedrohungen und Schwachstellen in Software und IT-Systemen.

✔ **Bemerkungen / Hinweise zur Tabelle:**

1. **Eingeschränkt anwendbar (♦)** = Methode ist eher deduktiv oder hardwarebezogen, kann aber teilweise auf Software oder Schnittstellen angewendet werden.
2. **Geeignet (✔)** = Methode ist explizit oder gut für Software-Risikoanalyse geeignet.
3. **Nicht geeignet (✘)** = Methode deckt andere Bereiche ab (Hardware, Prozess, Audit), nicht für Software-Risikoanalyse.

Qualitätsmanagement in der Automobilindustrie

Den aktuellen Stand der veröffentlichten VDA-Bände zum Qualitätsmanagement in der Automobilindustrie (QAI) finden Sie im Internet unter <http://www.vda-qmc.de>.

Auf dieser Homepage können Sie auch direkt bestellen.

Bezug:

Verband der Automobilindustrie e. V. (VDA)
Qualitäts Management Center (QMC)
10117 Berlin, Behrenstr. 35
Telefon +49 (0) 30 89 78 42-235, Telefax +49 (0) 30 89 78 42-605
E-Mail: info@vda-qmc.de, Internet: www.vda-qmc.de

