Erforschung von Ausbreitungswegen von Softwarefehlern in softwarebasierter Leittechnik in Kernkraftwerken

GRS - 599

Gesellschaft für Anlagen-
und Reaktorsicherheit
(GRS) gGmbH

Erforschung von Ausbreitungswegen von Softwarefehlern in softwarebasierter Leittechnik in Kernkraftwerken

Henriette Gatz
Felix Gärner
Patrick Gebhardt
Hervé Mbonjo
Christian Müller
Ewgenij Piljugin
Birte Ulrich
Dagmar Sommer

Juli 2020

Anmerkung:

Das diesem Bericht zugrunde liegende Forschungsvorhaben wurde mit Mitteln des Bundesministeriums für für Umwelt, Naturschutz und nukleare Sicherheit (BMU) unter dem Förderkennzeichen 4717R01331 durchgeführt.

Die Verantwortung für den Inhalt dieser Veröffentlichung liegt bei der GRS.

Der Bericht gibt die Auffassung und Meinung der GRS wieder und muss nicht mit der Meinung des BMU übereinstimmen.

GRS - 599
ISBN 978-3-947685-85-1

Deskriptoren

elementarer Funktionsblock (EFB), FMEA, Funktionsblockdiagramm (FBD), Leittechnikfunktionen (LEFUs), Softwarebasiertes Leittechniksystem, Softwarefehler

Abstract

This report presents results that have been developed within the framework of a BMU-funded research project (Promotion Code 4717R01331). The focus of this project was to systematically investigate the effects and the propagation of software faults/errors within software-based I&C systems.

For this purpose, a concept for performing a software failure mode and effects analysis (FMEA), in short Software-FMEA (SFMEA), on a software-based I&C system was developed with the aim to identify potential failures due to latent software errors in a software-based I&C system and to investigate their effects on the I&C functions (failure on demand, faulty operation, etc.). Such latent errors result, for example, from errors during operation, maintenance and modification of the system as well as signal paths that have not been tested during system tests. Since I&C functions of software-based I&C systems are typically specified on the basis of interconnected elementary function blocks (EFBs), the EFBs are defined in this project as the lowest level of abstraction for performing an FMEA. Consequently, in the SFMEA concept developed in the course of this work, the elementary function blocks are subjected to an FMEA in order to systematically identify their relevant failure modes. The effects of the identified failure modes of the elementary function blocks on the underlying I&C function can then be evaluated by means of dynamic simulation or fault tree analysis. For this purpose, an automation of the developed SFMEA method by means of dynamic simulations was elaborated within the framework of this project.

On the basis of some selected application examples of the developed SFMEA concept, it was shown within the scope of the work that particular faulty states of the input signals (undetected failures of input signals or input signals identified as faulty but correct) essentially determine the identified failure modes of the elementary function blocks and consequently also co-determine the propagation of faulty states in a software-based I&C system and their effects on the correct functioning of the I&C system.
Furthermore, the suitability of the developed SFMEA concept for determining the failure behaviour of I&C functions in the presence of faulty input signals (signal status and signal value) was tested using an example of an I&C function consisting of several elementary function blocks. Dynamic simulations performed to this end have shown that undetected input signal failures or incorrectly identified correct input signals can lead to faulty execution of the I&C function or non-execution of the I&C function.

Kurzfassung

Der vorliegende Bericht umfasst Ergebnisse, die im Rahmen des vom BMU geförderten Forschungsvorhabens „Erforschung von Ausbreitungswegen von Softwarefehlern in softwarebasierter Leittechnik in Kernkraftwerken“ (Förderkennzeichen 4717R01331), erarbeitet wurden. Der Schwerpunkt dieses Vorhabens lag darin, die Auswirkungen und die Ausbreitung von Softwarefehlern innerhalb von softwarebasierten Leittechniksystemen systematisch zu untersuchen.

Hierfür wurde im Rahmen dieses Vorhabens ein Konzept zur Durchführung einer Software- Fehlerart- und Auswirkungsanalyse (FMEA–Failure Mode and Effects Analysis), kurz Software-FMEA (SFMEA), an einem softwarebasierten Leittechniksystem mit dem Ziel entwickelt, potenzielle Ausfälle aufgrund latenter Softwarefehler in einem softwarebasierten Leittechniksystem zu identifizieren und deren Auswirkungen auf die Leittechnikfunktionen (Ausfall bei Anforderung, Fehlanregung usw.) zu untersuchen. Derartige latente Fehler resultieren beispielsweise aus Fehlern bei Betrieb, Instandhaltung und Modifikation des Systems sowie aus nicht im Rahmen von Systemtests erprobten Signalpfaden.

Da Leittechnikfunktionen softwarebasierter Leittechniksysteme typischerweise anhand von miteinander verknüpften elementaren Funktionsbausteinen (EFBs) spezifiziert werden, werden in diesem Vorhaben die EFBs als die niedrigste Abstraktionsebene zur Durchführung einer FMEA definiert. Folglich werden bei dem im Rahmen der Arbeit entwickelten SFMEA-Konzept die elementaren Funktionsblöcken einer FMEA unterzogen, um die relevanten Ausfallmodi der elementaren Funktionsblöcke systematisch zu identifizieren. Die Auswirkungen der identifizierten Ausfallarten der elementaren Funktionsblöcke auf die zu untersuchende Leittechnikfunktion können anschließend mittels dynamischer Simulation oder Fehlerbaumanalyse evaluiert werden. Hierfür wurde im Rahmen dieses Projektes eine Automatisierung der entwickelten SFMEA-Methode mittels dynamischer Simulationen erarbeitet.

Anhand einiger ausgewählter Anwendungsbeispiele des entwickelten SFMEA-Konzeptes konnte im Rahmen der durchgeführten Arbeiten gezeigt werden, dass insbesondere fehlerhafte Zustände der Eingangssignale (nicht erkannte Ausfälle von Eingangssignalen bzw. als fehlerhaft identifizierte aber korrekte Eingangssignale) die identifizierten Ausfallmodi der elementaren Funktionsblöcke wesentlich bestimmen und demzufolge die Ausbreitung fehlerhafter Zustände in einem softwarebasierten Leittechniksystem und deren Auswirkungen auf die korrekte Funktion des Leittechniksystems ebenfalls mitbestimmen.

Am Beispiel einer aus mehreren elementaren Funktionsbausteinen bestehenden Leittechnikfunktion, wurde die Eignung des entwickelten SFMEA-Konzeptes zur Bestimmung des Ausfallverhaltens von Leittechnikfunktionen bei Vorliegen fehlerhafter Eingangssignale (Signalstatus und Signalwert) erprobt. Hierbei konnte anhand durchgeführter dynamischer Simulationen gezeigt werden, dass nicht erkannte Ausfälle von Eingangssignalen bzw. fehlerhaft als ausgefallen identifizierte korrekte Eingangssignale zur fehlerhaften Ausführung bzw. zur Nichtausführung der Leittechnikfunktion führen können.

Inhaltsverzeichnis

1 Einleitung, Aufgabenstellung und Zielsetzung

In den letzten 20 Jahren haben softwarebasierte und programmierbare leittechnische Einrichtungen eine wachsende Bedeutung in deutschen Kernkraftwerken gewonnen. Wesentliche Ursachen hierfür sind die abnehmende Verfügbarkeit der Ersatzteile für die bisher eingesetzten konventionellen leittechnischen Einrichtungen z. B. aufgrund von Abkündigungen von Produkten und die erforderlichen Verbesserungen in der Prozessführung und -überwachung, die durch den Einsatz softwarebasierter leittechnischer Einrichtungen effizient realisiert werden können. Der Umfang der eingesetzten softwarebasierten leittechnischen Komponenten erstreckt sich je nach Anlage von einzelnen Betriebsmitteln bis zu komplett in softwarebasierter oder programmierbarer Leittechnik ausgeführten Systemen. Dies betrifft sowohl betriebliche Systeme und Komponenten als auch Einrichtungen, Komponenten und Systeme mit sicherheitstechnischer Bedeutung.

Die bisherige Betriebserfahrung mit softwarebasierten und programmierbaren leittechnischen Einrichtungen (u. a. Geräte der Instrumentierung, Steuerung und Prozessüberwachung) in deutschen Kernkraftwerken hat gezeigt, dass diese andersartiges Fehlerpotential im Vergleich zu Einrichtungen der konventionellen Leittechnik aufweisen. Dazu zählen insbesondere latente Fehler in der Software, die im Anforderungsfall die Funktion der betroffenen Einrichtungen beeinträchtigt oder verhindert haben oder bei der Überprüfung bereits in Betrieb befindlicher Einrichtungen entdeckt wurden. Softwarefehler umfassen u. a. Fehler in der Firmware und Programmierfehler in Algorithmen und können sich auf die Integrität und Funktion von Komponenten und Systemen mit sicherheitstechnischer Bedeutung eines Kernkraftwerkes auswirken.

Im Vorhaben „Auswirkungsbereiche von Softwarefehlern in sicherheitstechnisch wichtigen Einrichtungen von Kernkraftwerken“ (Vorhaben 3614R01304) hat die GRS ihre Wissensbasis über die Ursachen und Wirkungsweise von Softwarefehlern in softwarebasierter Leittechnik erweitert, indem Recherchen zur Betriebserfahrung mit Softwarefehlern in softwarebasierter Leittechnik angestellt, verschiedene Arten von Softwarefehlern ermittelt und die ermittelten Softwarefehlerarten ursachenorientiert klassifiziert wurden. Aufbauend auf den so identifizierten Softwarefehlerarten wurden potenzielle Auswirkungen von einzelnen Softwarefehlerarten auf einige spezielle Leittechnikfunktionen und auf die damit verknüpfte Verfahrenstechnik untersucht. Diese Vorgehensweise ist in /MBO 16/ und /GRS 17/ dokumentiert.

Bei der im Vorhaben „Auswirkungsbereiche von Softwarefehlern in sicherheitstechnisch wichtigen Einrichtungen von Kernkraftwerken“ entwickelten und erprobten Methode lag der Schwerpunkt der Untersuchungen auf der Analyse der Auswirkungen von Softwarefehlern auf die Systemebene (z. B. Auswirkung auf die angesteuerte verfahrenstechnische Komponente) /GRS 17/. Für eine umfassende Analyse der Auswirkungen von Softwarefehlern in softwarebasierter Leittechnik ist es ebenfalls von Bedeutung, die Auswirkungen von Softwarefehlern innerhalb von Leittechniksystemen systematisch zu untersuchen. Diese Untersuchungen bilden den Schwerpunkt des vorliegenden Vorhabens.

Dazu wurde im Rahmen dieses Vorhabens zunächst der relevante Stand von Wissenschaft und Technik (W&T) zu den Erkenntnissen zu Softwarefehlern, deren Auswirkungen in softwarebasierter Leittechnik und zu den Analysemethoden zur Untersuchung von Auswirkungen von Softwarefehlern in softwarebasierter Leittechnik in Kernkraftwerken ermittelt und aufbereitet (Arbeitspaket 1).

In einem weiteren Arbeitsschritt wurde ein Konzept zur Durchführung einer Software-Fehlerart- und Auswirkungsanalyse (FMEA–Failure Mode and Effects Analysis), kurz Software-FMEA, an einem softwarebasierten Leittechniksystem entwickelt. Hiermit sollen die Auswirkungen eines postulierten Softwarefehlers in einem Leittechniksystem untersucht werden können (Arbeitspaket 2).

In einem weiteren Arbeitsschritt wird ein Konzept zur Validierung der Analyseergebnisse einer durchgeführten Software-FMEA an einem softwarebasierten Leittechniksystem auf Systemebene anhand von Testanordnungen entwickelt (Arbeitspaket 3). Die drei Arbeitspakete im Rahmen dieses Vorhabens sind nachfolgend beschrieben.

1.1 Aufarbeitung des für das Vorhaben relevanten Standes von Wissenschaft und Technik (Arbeitspaket 1)

In diesem Arbeitsschritt wurde zunächst der relevante Stand von Wissenschaft und Technik hinsichtlich Methoden zur Analyse der Auswirkungen von Fehlern in softwarebasierter Leittechnik ermittelt und aufbereitet. Der Schwerpunkt der Recherchen lag dabei auf Methoden zur Modellierung von Software und zur systematischen Untersuchung von Software hinsichtlich Fehlereffekten und -auswirkungen.

Es wurde ebenfalls im Rahmen der Bearbeitung des Arbeitspaketes 1 nach relevanten Regelwerksanforderungen zur Vermeidung der Ausbreitung von Softwarefehlern in softwarebasierter Leittechnik recherchiert und die sich daraus ergebenden relevanten Aspekte, welche im Rahmen der Zuverlässigkeitsanalyse softwarebasierter Leittechnik zu berücksichtigen sind, dargestellt.

Weiterhin wurden ausgewählte relevante in der Betriebserfahrung mit softwarebasierten leittechnischen Systemen und Einrichtungen auf nationaler und internationaler Ebene beobachtete Ereignisse mit Softwarefehlern im Hinblick auf die Modellierung der Ausbreitung von Softwarefehlern in softwarebasierten Leittechniksysteme exemplarisch aufbereitet und zusammenfassend dargestellt.

Bei der Aufbereitung des Standes von Wissenschaft und Technik wurden u. a. folgende Informationsquellen herangezogen:

- Ergebnisse bisheriger GRS-Forschungsvorhaben zum Themenkomplex "Einsatz und Bewertung von softwarebasierter Leittechnik in Kernkraftwerken".
- Einschlägige Fachliteratur und weitere Informationsquellen wie z. B. NUREG-Berichte (Veröffentlichungen der USNRC), EPRI-Berichte, und Berichte von Gutachterorganisationen und Forschungsinstituten.
- Erkenntnisse aus relevanten Fachtagungen, Fachgesprächen und/oder Konferenzen zum Themenkomplex „Softwarebasierte Leittechnik in Kernkraftwerken“.

Der auf diese Weise aufbereitete Stand von Wissenschaft und Technik ist im Kapitel 2 dieses Abschlussberichtes dokumentiert.

1.2 Entwicklung eines Konzeptes zur Untersuchung von Softwarefehlern in softwarebasierter Leittechnik (Arbeitspaket 2)

Ziel des Arbeitspaketes 2 war die Entwicklung eines Konzeptes zur Untersuchung von softwarebedingten Ausfallarten in softwarebasierten Leittechniksystemen. Hiermit sollen die Auswirkungen eines postulierten Softwarefehlers innerhalb eines Leittechniksystems bis auf dessen Ausgangssignale untersucht werden können. Zunächst wurde ein generisches Modell eines softwarebasierten Leittechniksystems entwickelt. Unter Berücksichtigung von Methoden zur Modellierung von Software und zur systematischen Untersuchung von Software hinsichtlich Fehlereffekten und -auswirkungen in softwarebasierter

Leittechnik wurde ein Konzept zur Durchführung einer Software-FMEA an einem softwarebasierten Leittechniksystem entwickelt. Dieses Konzept wurde anschließend auf das entwickelte generische Modell eines softwarebasierten Leittechniksystems angewandt.

Bei den Untersuchungen im Arbeitspaket 2 wurden u. a. folgende Aspekte und Leitfragen zugrunde gelegt:

- Welche Komponenten eines softwarebasierten Leittechniksystems sind für die Ausführung einer Leittechnikfunktion notwendig?
- Welche Software ist in einem softwarebasierten Leittechniksystem vorhanden (z. B. Anwendungssoftware, Systemsoftware, Kommunikationssoftware…)?
- Welche Software ist zum Ausführen einer leittechnischen Funktion notwendig? In welchen Komponenten des Leittechniksystems wird sie ausgeführt?
- Wie wird die Software zum Ausführen leittechnischer Funktionen eines softwarebasierten Leittechniksystems modelliert?

Die gewonnenen Erkenntnisse aus diesem Arbeitspaket 2 wurden aufbereitet und sind im Kapitel 3 dieses Berichtes dokumentiert.

1.3 Entwicklung eines Konzeptes zur Validierung der Analyseergebnisse zu den Auswirkungen von Softwarefehlern auf softwarebasierte Leittechniksysteme (Arbeitspaket 3)

Arbeitspaket 3 hatte die Entwicklung eines Konzeptes, zur Validierung der Analyseergebnisse einer durchgeführten Software-FMEA zu den Auswirkungen von Softwarefehlern auf softwarebasierte Leittechniksysteme mittels Testanordnungen, zum Ziel. Unter Testanordnungen sind die speziell für den Testzweck entwickelten Einrichtungen und Modelle zu verstehen, welche mittels geeigneter Soft- und Hardware die zum Test erforderliche Leittechnikfunktionen nachbilden. Diese können z. B. anhand rechnerbasierter Simulationsmodelle (Matlab/Simulink o. ä.) oder mittels Soft- und Hardware eines Leittechniksystems realisiert werden. Die Eignung der Software-FMEA zur Untersuchung der Auswirkungen von Softwarefehlern in softwarebasierter Leittechnik soll somit validiert werden und die daraus gewonnenen Erkenntnisse können u. a. zur Anpassung/Modifizierung der Anwendung der Software-FMEA zur Auswirkungsanalyse von Softwarefehlern in softwarebasierter Leittechnik verwendet werden.

Es wurden die notwendigen Schritte für die Validierung der Ergebnisse einer durchgeführten Software-FMEA zur Untersuchung der Auswirkungen von Softwarefehlern an einem softwarebasierten leittechnischen System anhand von Testanordnungen herausgearbeitet bzw. spezifiziert. Dabei wurden u. a. Spezifika der Software-FMEA (z. B. hinsichtlich Zeitabhängigkeit und Berücksichtigung von Fehlererkennung- und -korrektur), die Testanordnungen hinsichtlich Systemeigenschaften, anzuwendende Softwarefehlermodelle und deren Implementierung in den Testanordnungen sowie die zu betrachtenden Szenarien berücksichtigt.

Im Rahmen der Entwicklung des Validierungskonzepts wurden u. a. folgende Aspekte evaluiert:

- Spezifikation relevanter Phasen der Signalverarbeitung (u. a. Signaleingabe, Kommunikationsaufgaben, Verarbeitung, Signalausgabe, Selbstüberwachungsaufgaben)
- Spezifikation der Betriebsart (z. B. Normalbetrieb, Betrieb nach Abschalten- und Neustart, Test) des softwarebasierten Leittechniksystems in den Testanordnungen
- Spezifikation der Fehlermodelle (Softwarefehlerart, Implementierung des Fehlers in der Testanordnung …)
- Spezifikation der Fehlererkennungs- und Fehlerbehandlungsprozesse des Leittechniksystems

Die gewonnenen Erkenntnisse aus diesem Arbeitspaket wurden aufbereitet und sind im Kapitel 4 dieses Abschlussberichtes dokumentiert.

2 Ermittlung und Aufbereitung des für das Vorhaben relevanten Standes von Wissenschaft und Technik

In diesem Kapitel werden die Ergebnisse der Arbeiten zum Arbeitspaket 1 „Aufarbeitung des für das Vorhaben relevanten Standes von Wissenschaft und Technik“ dargestellt.

Im Rahmen der Bearbeitung dieses Arbeitspakets wurden zunächst im Abschnitt 2.1 auf der Grundlage von /NEA 15/ Modelle zur Beschreibung der Fehlerausbreitung in softwarebasierter Leittechnik erarbeitet. Anschließend wurden die für das Vorhaben relevanten Regelwerksanforderungen an softwarebasierte Leittechniksysteme ermittelt und in Bezug auf die Vorsorge gegen die Ausbreitung von Softwarefehlern in softwarebasierten Leittechniksystemen ausgewertet. Die Ergebnisse dieses Arbeitsschrittes sind im Abschnitt 2.2 dargestellt. Im Abschnitt 2.3 werden in softwarebasierten Leittechniksystemen eingesetzte Fehlererkennungs- und -behandlungsmethoden als Vorsorgemaßnahmen gegen die Ausbreitung von Fehlern vorgestellt. In einem weiteren Arbeitsschritt wurde anhand einer ausführlichen Literaturrecherche und der Teilnahme an einer relevanten Fachtagung der aktuelle Stand von Wissenschaft und Technik zu Methoden der Zuverlässigkeitsanalyse von Software in softwarebasierter Leittechnik insbesondere hinsichtlich der Untersuchung zu den Auswirkungen und der Ausbreitung von Softwarefehlern erforscht. Die hierbei erzielten Ergebnisse sind im Abschnitt 2.4 zusammenfassend dargestellt. Anschließend werden im Abschnitt 2.5 exemplarisch Ereignisse vorgestellt, bei denen es zur Ausbreitung von Softwarefehlern innerhalb eines Leittechniksystems kam.

2.1 Modelle zur Beschreibung der Fehlerausbreitung in softwarebasierten Leittechniksystemen

In DIN 60880 /DIN 10/ und DIN 61513 /DIN 02/ wird der Begriff Fehler allgemein definiert als Mangel an einer Hardware-, Software- oder Systemkomponente. In Zusammenhang mit der Untersuchung der Ausbreitung von Fehlern in softwarebasierten Leittechniksystemen wird diese Definition wie folgt weiter spezifiziert.

Ein Fehler ist ein als latent anzusehender Mangel in einer Hardware- oder Softwarekomponente eines softwarebasierten Leittechniksystems, der bei entsprechender Aktivierung zum Versagen/Ausfall der betroffenen Komponente führen kann. Hierunter fallen auch Mängel in der Dokumentation des softwarebasierten Leittechniksystems.

In der Folge kann es zur Nichtausführung oder zur fehlerhaften Ausführung der betroffenen leittechnischen Funktion kommen.

Ist der Fehler aktiviert, kann er sich in das softwarebasierte Leittechniksystem ausbreiten. Hierfür sind gemäß /NEA 15/ zwei grundlegende Arten der Fehlerfortpflanzung zu berücksichtigen:

- Fehlerausbreitung durch **Folgeausfälle von Komponenten (Kaskadenmodell**): Der latente Fehler wird als primärer lokaler Fehler betrachtet. Der Fehler wirkt sich nach seiner Aktivierung an den Schnittstellen der fehlerbehafteten Komponente zu weiteren Systemkomponenten, z. B. in Form eines fehlerhaften Ausgangssignals, aus. Dieses fehlerhafte Ausgangssignal breitet sich in weitere Komponenten des softwarebasierten Leittechniksystems aus. Die hieraus resultierenden fehlerhaften Eingabesignale verursachen in der Folge weitere Komponentenausfälle im softwarebasierten Leittechniksystem.
- Fehlerausbreitung durch Ausfall **aufgrund gemeinsamer Ursache (CCF**[1]): Bei dieser Art der Fehlerausbreitung fallen mindestens zwei Komponenten gleichzeitig aufgrund der gleichen Ursache aus. Dies kommt z. B. vor, wenn der für die Ausfälle ursächliche latente Fehler in mehreren Systemkomponenten vorhanden ist. Bei seiner Aktivierung führt der Fehler zum Ausfall aller betroffenen Systemkomponenten.

Latente Fehler können in jeder beliebigen Phase des Systemlebenszyklus eines softwarebasierten Leittechniksystems eingebracht werden. Hierzu zählen potenzielle Fehler im Rahmen der Systementwicklung und/oder Systemmodifikation, fehlerhafte Systemspezifikation sowie unerkannte Fehler im Rahmen der Systemintegration, der Systemvalidierung und der Systemerrichtung.

Die Aktivierung latenter Fehler erfolgt unter bestimmten Randbedingungen (Aktivierungsbedingungen) z. B. bei Anforderung der Funktion der fehlerbehafteten Komponente durch ein einleitendes Ereignis. Diese Aktivierungsbedingungen können abhängig vom vorliegenden Anlagenzustand zum Ausfall der fehlerbehafteten Komponente ggfs.

[1] CCF: Englisch, Akronym für “common cause failures“, zu Deutsch gemeinsam verursachter Ausfall (GVA): Gleichzeitiger Ausfall von mindestens zwei Strukturen, Systemen oder Komponenten aufgrund gleicher Ursache. Ursache für CCF kann ein einzelnes spezifisches Ereignis sein. CCF können auch durch latente Fehler verursacht werden, die sich systematisch auswirken.

zum vollständigen Ausfall des Systems (abhängig von den Folgeauswirkungen auf andere Komponenten) führen. Diese Zusammenhänge sind in Abb. 2.1 dargestellt.

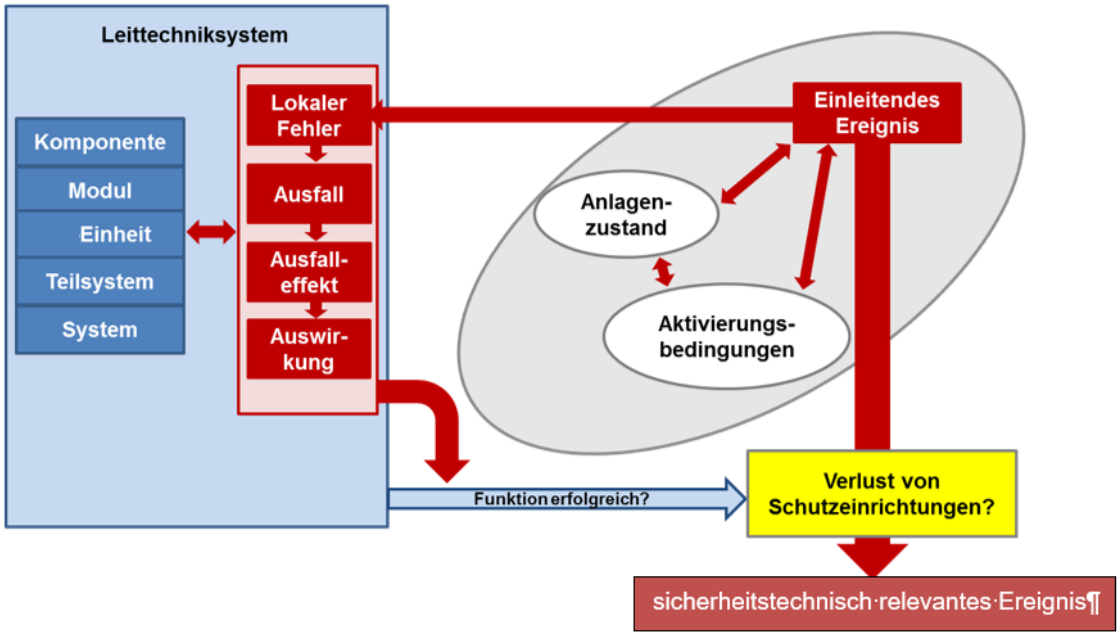

Abb. 2.1 Prinzipbild zur Beschreibung der Fehlerausbreitung in einem softwarebasierten Leittechniksystem basierend auf /NEA 15/, /GRS 15/

Zu den möglichen Ursachen für die Aktivierung latenter Fehler (Erfüllung der Aktivierungsbedingungen) zählen beispielsweise Fehler an der Mensch-Maschine-Schnittstelle bei Betrieb, Instandhaltung und Modifikation des Systems. Nicht erprobte Signalpfade im Rahmen von Systemtests können ebenfalls zur Aktivierung latenter Fehler beitragen.

Das übergeordnete Ziel der Komponenten- und Systemauslegung softwarebasierter Leittechniksysteme ist es, die Wahrscheinlichkeit des Vorhandenseins latenter Fehler zu minimieren, um hieraus resultierende potenzielle sicherheitstechnische Beeinträchtigungen zu vermeiden.

Die hierfür zu erfüllenden Auslegungsanforderungen und -prinzipien sind abhängig von der sicherheitstechnischen Kategorisierung der auszuführenden Leittechnikfunktionen.

Für diese Kategorien gelten abgestufte Anforderungen. Im Abschnitt 2.2 wird auf diesbezügliche Anforderungen eingegangen. Es werden ebenfalls in diesem Zusammenhang die hierzu relevanten Regelwerksanforderungen hinsichtlich der Vorsorge gegen CCF in softwarebasierten Leittechniksystemen vorgestellt.

2.2 Für das Vorhaben relevante Regelwerksanforderungen an softwarebasierte Leittechnik

In diesem Abschnitt werden Regelwerksanforderungen an die Software in softwarebasierten Leittechniksystemen insbesondere hinsichtlich der Vorsorge gegen Softwarefehler, deren Auswirkungen und Ausbreitung dargestellt. Die damit verknüpften Anforderungen an die Zuverlässigkeitsuntersuchung softwarebasierter Leittechnik werden ebenfalls vorgestellt.

Hierzu werden die Anforderungen aus /BMU 15/, /BMU 15b/, /KTA 15/ und /DIN 02/, /DIN 10/, /DIN 19a/ und /IAE 09/ betrachtet.

2.2.1 Anforderungen an die Software in softwarebasierter Leittechnik

Die Anforderungen an Leittechniksysteme richten sich nach der sicherheitstechnischen Bedeutung der realisierten Leittechnikfunktionen. Entsprechend ihrer sicherheitstechnischen Bedeutung werden gemäß /BMU 15/, /BMU 15b/ und /KTA 15/ die Leittechnikfunktionen in die Kategorien A, B und C eingeteilt. Die Kategorie A umfasst demnach alle Leittechnikfunktionen, die erforderlich sind, um Ereignisse der Sicherheitsebene 3 zu beherrschen. Kategorie B umfasst alle Leittechnikfunktionen, die erforderlich sind, um Ereignisse der Sicherheitsebene 2 zu beherrschen sowie das Eintreten von Ereignissen der Sicherheitsebene 3 zu vermeiden. Die Kategorie C umfasst alle übrigen sicherheitstechnisch wichtigen Leittechnikfunktionen.

Übergeordnete Anforderungen an die Software bzw. Softwareentwicklung für Leittechnikfunktionen der Kategorien A bis C sind in /BMU 15/ und /BMU 15b/ angegeben. Demnach ist die Software für Leittechnikfunktionen der Kategorien A bis C in verifizierbaren Schritten nach einem Phasenmodell zu entwickeln. Hierbei ist gemäß /KTA 15/ die Anwendersoftware ausgehend von der verfahrenstechnischen Aufgabenstellung zu entwickeln. Die Softwarearchitektur von leittechnischen Einrichtungen ist gemäß /BMU 15b/ so zu gestalten, dass die Funktionen der Anwendersoftware und der Systemsoftware in

eigenständigen Softwareeinheiten realisiert sind und die Anwendersoftware von der Systemsoftware getrennt ist. Weiterhin heißt es in den /BMU 15b/, die Software sei so auszulegen ist, dass keine unzulässigen Rückwirkungen von leittechnischen Einrichtungen, die Leittechnikfunktionen der sicherheitstechnisch niederwertigeren Kategorie ausführen, auf die leittechnischen Einrichtungen, die Leittechnikfunktionen der sicherheitstechnisch höherwertigeren Kategorie ausführen, auftreten.

Die Software ist so zu gestalten, dass deren anforderungsgerechter Ablauf unabhängig von Art und Umfang der zeitlichen Änderung ihrer Eingangssignale gewährleistet ist.

Hinsichtlich der Auslegung softwarebasierter Leittechniksysteme mit Leittechnikfunktionen auf der Sicherheitsebene 3 wird in /BMU 15/ eine einfache Struktur der Software, eine Begrenzung des Funktionsumfangs von Hard- und Software auf das sicherheitstechnisch notwendige Maß sowie den Einsatz fehlervermeidender, fehlerentdeckender und fehlerbeherrschender Maßnahmen und Einrichtungen gefordert.

Gemäß /BMU 15/ sind die Anforderungen an Entwurf, Implementierung, Qualifizierung, Inbetriebsetzung, Betrieb und Modifizierung der Software und an Auslegung, Fertigung, Errichtung und Betrieb der Hardware (Komponenten, Baugruppen und Teilsysteme) für leittechnische Einrichtungen entsprechend der sicherheitstechnischen Klassifizierung der von ihnen ausgeführten Funktionen festzulegen.

Gemäß /KTA 15/ wird entsprechend der Kategorie der auszuführenden Leittechnikfunktion zwischen Funktionseinrichtungen der Kategorie A, B und C unterschieden.

Hierbei sind Funktionseinrichtungen der Kategorie A gemäß /KTA 15/ Einrichtungen zur Ausführung von Leittechnikfunktionen der Kategorie A. Funktionseinrichtungen der Kategorie B und C sind entsprechend definiert.

In /KTA 15/ wird hinsichtlich der nachzuweisenden Softwarequalität für Funktionseinrichtungen der Kategorie A u. a. aufgeführt, die Entwicklung und Qualifizierung der Software habe so zu erfolgen, dass eine durchgängige Nachweisführung der korrekten Arbeitsweise der Software gewährleistet ist. Zudem wird in /KTA 15/ für Funktionseinrichtungen der Kategorie A gefordert, dass die Ergebnisse der einzelnen Phasen der Softwareentwicklung unter Anwendung systematischer Analysen und daraus abgeleiteter Tests an den Vorgaben vollständig zu verifizieren sind.

Für die Entwicklung und Qualifizierung der Software der Leittechnikfunktionen der Kategorie B sind gemäß /KTA 15/ Beschreibungen und rechnergestützte Testverfahren anzuwenden, die den Nachweis der korrekten Arbeitsweise unterstützen. Die Ergebnisse der einzelnen Phasen der Softwareentwicklung sind einer dokumentierten Prüfung zu unterziehen. Alle sicherheitsrelevanten Programmteile sind durch eine Kombination von Testverfahren zu prüfen, wobei eine vollständige Funktionstestüberdeckung erreicht werden soll.

Bei der Erstellung der Software für Leittechnikfunktionen der Kategorie C sind gemäß /BMU 15/ die Entwicklungsschritte einzeln auszuweisen. Das Erreichen der Phasenziele ist hierbei durch Prüfungen nachzuweisen und zu dokumentieren.

Weiterhin heißt es in /KTA 15/, dass die Programme für Funktionseinrichtungen der Kategorie A und B robust und selbstüberwachend auszulegen sind. Softwarerobustheit bedeutet hierbei, dass undefinierte Zustände verhindert werden. Gemäß /KTA 15/ muss es für jeden Eingangswert zu jedem Zeitpunkt einen wohl definierten Ausgangswert geben. Die Selbstüberwachung bezeichnet gemäß /KTA 15/ die Eigenschaft von Komponenten oder Systemen, ihre Ausfälle selbsttätig erkennbar zu machen.

Den obigen Ausführungen ist zu entnehmen, dass in den Anforderungen aus /BMU 15/ und /BMU 15b/ und aus /KTA 15/ an die Software bzw. an die Softwareentwicklung softwarebasierter Leittechniksysteme Aspekte enthalten sind, die der Vermeidung von und der Vermeidung der Ausbreitung von Softwarefehlern in softwarebasierter Leittechnik dienen. Hierzu zählen der Einsatz von fehlervermeidenden Maßnahmen, die Entwicklung der Software nach einem Phasenmodell in verifizierbaren Schritten, die Gestaltung

der Softwarearchitektur als eigenständige aufgabenspezifische Softwareeinheiten, der Nachweis der korrekten Arbeitsweise der Software sowie die Verifizierung der Ergebnisse der einzelnen Phasen der Softwareentwicklung unter Anwendung systematischer Analysen. Die genannten Maßnahmen unterstützen die Erkennung möglicher Fehler bereits in der Entwicklungsphase der Software. Die geforderte robuste Auslegung der Software dient dazu, ein definiertes Verhalten einschließlich Ausfallverhalten der Software zu erzielen. Der Einsatz von fehlerentdeckenden und fehlerbeherrschenden Maßnahmen ermöglicht eine fehlertolerante Auslegung des Leittechniksystems. Hierbei unterstützt die Selbstüberwachung das fehlerentdeckende Verhalten des Leittechniksystems.

In /DIN 10/ und /DIN 02/ finden sich im Hinblick auf die Vermeidung von und Vermeidung der Ausbreitung von Softwarefehlern in softwarebasierter Leittechnik vergleichbare Auslegungsanforderungen an die Software bzw. Softwareentwicklung softwarebasierter Leittechniksysteme wie in /BMU 15/, /BMU 15b/ und /KTA 15/. Gemäß /DIN 10/ sollte die Programmstruktur der Software auf einer Zerlegung in Module basieren und einfach und leicht verständlich sein, und zwar sowohl hinsichtlich ihrer Gesamtauslegung als auch ihrer Details. Weiterhin sollte entsprechend /DIN 10/ das Quellprogramm dokumentierten Regeln folgen, die der Verbesserung der Klarheit, Modifizierbarkeit und Prüfbarkeit dienen und das Programm sollte so geschrieben sein, dass eine einfache Verifizierung ermöglicht wird.

Hinsichtlich der Ausbreitung und der Auswirkungen von Ausfällen in softwarebasierten Leittechniksystemen werden in /DIN 10/ und /DIN 02/

- die Benutzung gemeinsamer Ressourcen (Verarbeitungskapazität, Kommunikationsbandbreite, Speicher, usw.) und gemeinsamer Daten sowie
- Fehler in einem Softwareprogramm, die zum Versagen anderer Programme oder Hardware eines Leittechnik-Systems führen,

als Ursachen für die Ausbreitung von Ausfällen angeführt. Zur Minimierung der Wahrscheinlichkeit und der Auswirkungen der Ausbreitung von Ausfällen werden in den /DIN 10/ und /DIN 02/ u. a. folgende Maßnahmen genannt:

- Interne Isolierung zwecks Eingrenzung des Auswirkungsbereiches eines Ausfalls.
- Systemüberwachung zwecks Erkennung „verfälschter“ Daten und/oder schadhafter Ressourcen mit internen Mitteln (z. B. Selbstüberwachung) oder externen Mitteln (z. B. andere Systeme oder Operateure).
- Vorgaben für den Fall einer Fehlererkennung, die es dem System ermöglichen, sein Potential gegen die Ausbreitung von Ausfällen und/oder entsprechenden Auswirkungen zu reduzieren.

Der Selbstüberwachung wird hierbei gemäß /DIN 10/ eine wichtige Rolle zugewiesen, denn diese wird als primärer Faktor für die Erzielung hoher Zuverlässigkeit des Gesamtsystems betrachtet. Dementsprechend wird in /DIN 10/ diesbezüglich u. a. Folgendes gefordert:

- Die Software-Auslegung muss eine Selbstüberwachung einschließen.
- Die Selbstüberwachung sollte in der Lage sein, in praktikablem Ausmaß Zufallsausfälle von Hardware-Komponenten sowie fehlerhaftes Verhalten der Software (z. B. Abweichungen von spezifizierten Software-Abläufen und Betriebsbedingungen oder Datenbeschädigung) zu entdecken.

Auslegungsprinzipien, die eine geeignete Selbstüberwachung unterstützen, umfassen gemäß /DIN 10/ u. a die Modularisierung, die Durchführung von Plausibilitätsprüfungen und die Verwendung von Redundanz und Diversität. Diversität kann hierbei gemäß /DIN 10/ als funktionale oder als Software-Diversität realisiert werden. Ausfallsimulationen können verwendet werden, um die Eignung der Selbstüberwachung zu verifizieren.

In Zusammenhang mit der Systemüberwachung muss gemäß /DIN 10/ der Speicherplatz für unveränderbare Parameter und für Software schreibgeschützt oder hinsichtlich Änderungen überwacht sein. Eine derartige Auslegung ist nach /DIN 10/ u. a geeignet zum Schutz gegen nicht autorisierte Änderungen sowie zur Vermeidung des Fortpflanzens von Adressierfehlern oder Hardwarefehlern, einschließlich intermittierenden Fehlern.

Nach Erkennung eines Fehlers/Ausfalls sollen gemäß /DIN 10/ geeignete automatische Maßnahmen ergriffen werden. Hierzu wird bezüglich des Systemverhaltens empfohlen, soweit wie möglich eine sicherheitsgerichtete Auslegung (Fail-Safe-Auslegung) anzustreben. Dies bedeutet, dass bei Erkennung eines Ausfalls/Fehlers, z. B. durch die Selbstüberwachung, das System ein definiertes sicherheitsgerichtetes Ausgangssignal erzeugen muss. Softwarebedingte Fehler müssen hierbei zu einem sicherheitsgerichteten Ausfall der betroffenen Leittechnikfunktion führen. Kann dies nicht sichergestellt werden, darf das Ausgangssignal des Systems nur weniger wichtige Sicherheitsanforderungen verletzen. Dies bedeutet, dass in solchen Fällen das Ausgangssignal nur eine geringere sicherheitstechnische Bedeutung haben darf. Die Folgen des Ausfalls sind hierbei zu minimieren. Ausfallheilende Routinen wie definierte Rückfallebene, Wiederherstellung des Systems, sollten gemäß /DIN 10/ in die Überlegungen mit einbezogen werden. Ein solches ausfallsicheres Verhalten ist zur Vermeidung der Fehlerfortpflanzung in einem softwarebasierten Leittechniksystem und eines damit verknüpften potenziellen Systemversagens vorteilhaft.

Eine hohe sicherheitstechnische Bedeutung wird der Beherrschung von systematischen Ausfällen für Funktionseinrichtungen der Kategorie A in /KTA 15/ beigemessen. Entsprechend /KTA 15/ sind bei der Auslegung von Funktionseinrichtungen der Kategorie A die Potenziale für und die Auswirkungen von systematischem Versagen der leittechnischen Einrichtungen auf die Störfallabläufe unter Berücksichtigung der verfahrenstechnischen Vorgaben zu analysieren. Es sind Vorkehrungen gegen systematisches Versagen zur Minderung von dessen Eintrittswahrscheinlichkeit derart zu treffen, dass es zum Nachweis der Störfallbeherrschung nicht mehr unterstellt werden muss.

Das im vorangegangenen Abschnitt erwähnte systematische Versagen der Funktionseinrichtungen der Kategorie A schließt sowohl einen systematischen Ausfall der Hardware als auch ein systematisches Softwareversagen ein. Beide können Folge eines CCFs sein.

Kann für leittechnische Einrichtungen diese Nachweisführung nach dem Stand von Wissenschaft und Technik nicht erfolgen, sind gemäß /KTA 15/ Vorkehrungen derart zu treffen, dass ein systematisches Versagen von Hardware und Software der Funktionseinrichtungen der Kategorie A durch diversitäre oder dissimilare leittechnische Einrichtungen mit gleichen Qualitätsanforderungen beherrscht wird. Diversitätsgrad und Struktur sind dabei derart zu wählen, dass das systematische Versagen mit den damit verbundenen Auswirkungen die Störfallbeherrschung durch die verbleibenden diversitären Einrichtungen nicht unzulässig beeinflusst.

2.2.2 Anforderungen an die Datenkommunikationsnetzwerke softwarebasierter Leittechniksysteme

Softwarebasierte Leittechniksysteme weisen in der Regel folgende Merkmale auf:

- Modularer Aufbau der Hardware (Eingabemodule, Verarbeitungsmodule, Ausgabemodule…),
- Vorhandensein von Kommunikationsnetzwerken für den Datenaustausch zwischen den einzelnen Modulen,
- Vorhandensein von Schnittstellen zwischen den einzelnen Modulen des Leittechniksystems und zu anderen Rechnersystemen (z. B. Steuerstabfahrrechner, Leistungsverteilungsrechner) und Kommunikationsnetzwerken und
- Vorhandensein von Schnittstellen für Instandhaltung des Leittechniksystems, Konfigurierung und Parametrierung der System- und Anwendungsfunktionen (z. B. Servicerechner, Servicegerät).

Die genannten Schnittstellen und Kommunikationsnetzwerke eines softwarebasierten Leittechniksystems tragen zur Fehlerfortpflanzung im Leittechniksystem bei, da der interne Datenaustausch zwischen den einzelnen Komponenten des softwarebasierten Leittechniksystems sowie der externe Datenaustausch zu anderen Leittechniksystemen hierüber stattfindet. Darüber hinaus können Fehler (Software- und Hardwarefehler) im Kommunikationsnetzwerk dazu führen, dass entweder keine Datenübertragung zwischen den Leittechnikkomponenten stattfindet oder Daten fehlerhaft übertragen werden.

Zur Vermeidung von Fehlern und deren Fehlerfortpflanzung in der Datenkommunikation softwarebasierter Leittechniksysteme, die Leittechnikfunktionen der Kategorie A ausführen, sind in /DIN 19a/ Anforderungen an Aufbau, Funktion und Zuverlässigkeit zu finden. Diese sind nachfolgend zusammenfassend aufgelistet:

Physikalische Trennung

- Das Kommunikationsgerät sollte so ausgelegt sein, dass sich Fehler nicht von einem Teil des Geräts zum anderen Teil oder zu einem anderen System fortpflanzen.

Funktionale Unabhängigkeit

- Verarbeitungsmodule, die von der Kommunikation unabhängige Aufgaben wahrnehmen, müssen so ausgelegt sein, dass sie ihren Betrieb selbst beim Versagen eines Kommunikationspartners fortsetzen.

- Verarbeitungsmodule müssen für unabhängige Kommunikationsverbindungen separate Kommunikationsschnittstellen haben. Bei der Auslegung sollten separate Softwaremodule für die Verarbeitung von Anwendungsdaten und für die Handhabung der Kommunikation verwendet werden.

Entdecken von Kommunikationsfehlern

- Kommunikationsgeräte sollten selbstüberwachende Funktionen beinhalten. Entdeckte Fehler müssen in der Warte angezeigt werden. Kommunikationsgeräte müssen die Integrität der übertragenen Daten prüfen, um die korrekte Übertragung zu bestätigen oder Übertragungsfehler zu melden bzw. aufzuzeichnen.

- Kommunikationsgeräte müssen mit Einrichtungen zur Fehlererkennung ausgestattet sein. Damit muss sichergestellt werden, dass Fehler in der Datenkommunikation entdeckt werden, so dass fehlerhafte Daten die Ausführung von Funktionen der Kategorie A nicht beeinträchtigen können. Insbesondere sollte Folgendes erkannt werden:

  a) Verfälschung von Bits der übertragenen Nachricht;
  b) Übertragen von überholten Daten (die von einer ungewollten Wiederholung alter Nachrichten stammen);
  c) Verlust der Nachricht;
  d) falsch adressierte Nachricht;
  e) inakzeptable Verzögerung der Nachricht;
  f) inkorrekte Nachrichtensequenz.

Reaktion auf Fehler

- Leittechnische Systeme, die Funktionen der Kategorie A ausführen, müssen bei Erkennen von Übertragungsfehlern entsprechende Aktionen auslösen.

- Entdeckte Ausfälle von Kommunikationsgeräten, die zu einer inakzeptablen Verschlechterung von nuklearen Sicherheitsfunktionen des leittechnischen Systems führen, müssen dem Schichtpersonal in der Warte entsprechend angezeigt werden.

- Wenn Ausfälle von Kommunikationsgeräten entdeckt werden, sollten geeignete automatische Maßnahmen greifen, z. B.:

  a) Isolation des ausgefallenen Kommunikationskanals,

  a) Anzeige des ausgefallenen Geräts, um das Schichtpersonal zu warnen.

- Die bei Entdecken von Ausfällen zu ergreifenden Maßnahmen müssen spezifiziert werden, z. B. Aufzeichnung, Warnung des Wartungspersonals, Alarmmeldung für sofortige korrigierende oder abmildernde Maßnahmen.

- Im Zuge des Auslegungs- und Verifizierungsprozesses müssen Kommunikationsgeräte und -prozesse mit Hilfe geeigneter Methoden systematisch analysiert werden, z. B. FMEA im Hinblick auf die Folgen von Ausfällen für Funktionen der Kategorie A.

- Ausfall oder Fehlverhalten einzelner Kommunikationsknoten[2] darf sich auf die Verfügbarkeit des leittechnischen Systems nicht auswirken.

- Die möglichen Auswirkungen des Ausfalls von Kommunikationsknoten und -kanälen auf die Leistungsfähigkeit von Funktionen der Kategorie A muss bei der Auslegung analysiert und dokumentiert werden. Alle von dem System als Folge eines entdeckten Ausfalls geforderten Aktionen müssen festgelegt werden, z. B. zeichne den Ausfall auf, löse eine Meldung aus, bringe die Anlage in einen sicheren Zustand.

- Kommunikationskanäle sollten unempfindlich gegenüber transienten Fehlern sein, etwa eine nicht berücksichtigte Nachricht oder Fehler in einer einzelnen Nachricht, vorausgesetzt die Häufigkeit solcher Fehler ist nicht so groß, dass die Leistungsfähigkeit von Funktionen der Kategorie A beeinträchtigt wird. Solche transienten Fehler sollten nicht zur Abschaltung eines Kanals führen, sollten aber von dem System aufgezeichnet werden.

[2] Kommunikationsknoten: Verbindungspunkt an einem Kommunikationsnetzwerk, zu oder von dem Daten über Kommunikationskanäle von oder zu anderen Punkten im Netzwerk übertragen werden.

Vorsorge gegen Ausfälle (einschließlich CCF)

- Auf Datenkommunikationsgeräte können Bedingungen einwirken, die zu einem zeitnahen Ausfall mehrerer redundanter Teile des Systems führen. Um die Möglichkeit eines solchen zeitnahen Ausfalls mehrerer Module durch Gefährdungen, gegen die ein System ausgelegt ist, auszuschließen oder dessen Folgen zu minimieren, müssen die folgenden Gefährdungen in Betracht gezogen werden:

  a) seismische Störungen oder andere relevante externe Gefährdungen

  b) Brand, Rauch oder Überflutung in Geräte- oder Kabelbereichen

  c) Ausfall der Klimaanlage, Heizung und Lüftung

  d) exzessive Strahlung oder andere Umgebungsfaktoren von außerhalb des Geräts

  e) interne Faktoren des Geräts selbst

- Kabelpritschen, auf denen Kabel für die Datenkommunikation zwischen getrennten redundanten Strängen verlegt werden, müssen in Übereinstimmung mit den Anforderungen in IEC 60709 /DIN 19/ ausgelegt und separiert sein, so dass mögliche Gefährdungen begrenzt sind und die geforderte Fehlertoleranz für das gesamte leittechnische System eingehalten wird.

- Die Datenkommunikation in softwarebasierten Leittechniksystemen zur Ausführung von Leittechnikfunktionen der Kategorie A muss so ausgelegt sein, dass Fehlerfortpflanzung, z. B. durch Übertragung verfälschter Daten, verhindert wird. Diesbezügliche Anforderungen sind in /IEC 07/ angegeben. Hierzu zählen u. a.:

  - Leittechnische Systeme müssen so ausgelegt sein, dass ihr Betrieb durch zentrale Teilsysteme nicht behindert werden kann, die z. B. Informationen zur Darstellung in der Warte zur Verfügung stellen oder die Modifikation von aus dem Anlagenprozess abgeleiteten Parametern unterstützen, und die zur Ausführung solcher Funktionen eine Kommunikation zu allen redundanten Strängen eines leittechnischen Systems brauchen, das Funktionen der Kategorie A ausführt.

  - Fehlerhafte Daten müssen von der weiteren Bearbeitung innerhalb der Anwendungssoftware ausgeschlossen werden

  - Alle Funktionen der Systemsoftware, die für die Übertragung von Nachrichten vorgesehenen sind, müssen so realisiert werden, dass die korrekte Ausführung

dieser Softwarefunktionen nicht durch irgendwelche Werte von prozessabhängigen Daten gestört werden können, die selbst Gegenstand dieser Übertragung sind.

- Die Korrektheit der empfangenen Daten muss vor einer weiteren Bearbeitung überprüft werden.
- Die physikalische Trennung redundanter Teilsysteme muss entsprechend /IEC 09/ ausgelegt sein
- Der Austausch von Eingangsdaten zwischen redundanten Einheiten kann Abhängigkeiten zwischen den Einheiten generieren und muss daher hinsichtlich des CCF – Potentials untersucht werden.
- Online Validierung der Eingangsdaten (z. B. mittels Auswahlschaltungen) sollte eingesetzt werden, um die Fortpflanzung fehlerhafter Daten zu begrenzen. Solche Eingangssignale, deren Fehlerhaftigkeit bereits bekannt ist (z. B. durch Bereichsüberschreitung), sollten gekennzeichnet und von weiterer Bearbeitung ausgeschlossen werden.

- Die berücksichtigten potenziellen Ausfälle und die dagegen vorgesehenen Maßnahmen müssen analysiert und dokumentiert werden.

2.2.4 Relevante Aspekte der Zuverlässigkeitsuntersuchung softwarebasierter Leittechniksysteme

In diesem Abschnitt werden die zu berücksichtigenden Aspekte im Rahmen der Zuverlässigkeitsuntersuchung softwarebasierter Leittechnik zur Vermeidung der Ausbreitung von Fehlern/Ausfällen ermittelt. Es werden hierzu die Anforderungen aus /KTA 15/, /BMU 15/ und /BMU 15b/ sowie aus /DIN 02/, /DIN 10/ und /IAE 09/ betrachtet.

In /DIN 02/ wird ein angemessener Nachweis der Zuverlässigkeit der durch das leittechnische System ausgeführten Anwendungsfunktionen gefordert. Hierbei soll gemäß /DIN 02/ die Strenge des Nachweises abhängig von der Kategorie der Leittechnikfunktionen (Funktion der Kategorie A, Kategorie B oder Kategorie C gemäß /DIN 05/[3]) sein. Im Rahmen dieser Nachweisführung sind u. a. folgende Aspekte zu berücksichtigen /DIN 02/, /KTA 15/, /BMU 15/:

- Die Ermittlung des Anteils möglicher Hardwareausfälle an der Zuverlässigkeit der Funktion muss mit einer quantitativen probabilistischen Analyse erfolgen, die auf Ausfallraten von Komponenten beruht. /DIN 02/
- Die Ermittlung des Anteils möglicher Softwareauslegungsfehler an der Zuverlässigkeit der Funktion sollte auf einer qualitativen Abschätzung beruhen, bei der die Komplexität der Auslegung, die Qualität des Entwicklungsvorgangs und der Rückfluss an Betriebserfahrung berücksichtigt wird. Die Abschätzung sollte auf einem zuvor vereinbarten Verfahren beruhen und sollte dem Nachweis dienen, dass die Softwarequalität der Zielzuverlässigkeit entspricht. /DIN 02/
- Für eine quantitative Abschätzung können die Ergebnisse von Analysen und Simulationsprüfungen herangezogen werden, es gibt jedoch kein anerkanntes anzuwendendes Verfahren. /DIN 02/
- Die Auswirkungen von Einzelausfällen, CCF und Ausbreitung von Ausfällen innerhalb aller Systeme. /DIN 02/

[3] Die Definitionen von Leittechnikfunktionen der Kategorien A, B und C sind in /DIN 05/ angegeben. Sie sind mit denen aus /KTA 15/ äquivalent.

- Die Auswirkungen von systematischem Versagen (systematischer Ausfall der Hardware oder systematisches Softwareversagen) von Funktionseinrichtungen der Kategorie A auf die Störfallabläufe unter Berücksichtigung der verfahrenstechnischen Vorgaben. /BMU 15/, /KTA 15/

Eine hohe sicherheitstechnische Bedeutung wird hierbei den Aspekten CCF und systematisches Versagen, insbesondere dem durch Softwarefehler potenziell ausgelösten CCF, beigemessen. Gemäß /DIN 10/ kann z. B. ein CCF auftreten, wenn ein latenter Fehler in zwei oder mehreren Komponenten oder Systemen vorhanden ist und alle diese Komponenten oder Systeme unter denselben oder ähnlichen Bedingungen betrieben werden, so dass ein Versagen auf zeitlich korrelierte Weise ausgelöst werden kann, oder wenn Versagensbedingungen über die Datenkommunikation weitergetragen werden. Derartig auftretende CCF können zum Versagen von Sicherheitsfunktionen bei Anforderung oder zur fehlerhaften Auslösung von Sicherheitsfunktionen führen. Folglich ist die Beherrschung von potenziellen CCF und deren Auswirkungen im Rahmen der Nachweisführung zur Zuverlässigkeit softwarebasierter Leittechnik anhand einer CCF-Analyse zu belegen. Das Erfordernis der Durchführung einer solchen CCF-Analyse ist abhängig von der sicherheitstechnischen Kategorie der betroffenen Leittechnikfunktionen. Für Leittechnikfunktionen der Kategorie A ist gemäß /KTA 15/ eine CCF-Analyse durchzuführen. Mögliche Vorkehrungen bzw. anzuwendende Prinzipien zur Vermeidung von CCF bzw. systematischem Versagen in softwarebasierter Leittechnik umfassen u. a: /DIN 10/, /IAE 09/

Minimierung von Fehlern:

- Um die Anzahl der Fehler zu minimieren, werden gemäß /IAE 09/ zwei Hauptansätze verfolgt: Fehlervermeidung, -erkennung und -beseitigung. Ansätze zur Fehlervermeidung oder -prävention werden während des Designs und der Entwicklung softwarebasierter Leittechniksysteme eingesetzt. Hierzu gehören u. a. die während des gesamten Lebenszyklus verwendeten Verfahren zur Erzeugung zuverlässiger Software wie beispielsweise der Einsatz von Software-Diversität und Mehrfachprogrammierung sowie die Ablaufüberwachung von Programmen /EHR 02/. Fehlererkennungs- und -beseitigungsmethoden werden während der System-/Softwareentwicklung, hauptsächlich bei Verifikations- und Validierungsaktivitäten (V&V), eingesetzt. Die gebräuchlichste Methode zur Fehlererkennung und -beseitigung im Rahmen von V&V-Aktivitäten ist das Testen. Ein Überblick über einzusetzende Verifizierungs- und Prüfmethoden findet sich in /DIN 10/.

Vermeidung gemeinsamer Fehler:

- Trotz der Maßnahmen zur Minimierung von Fehlern bei der System-/Software-Entwicklung wird postuliert, dass Restfehler im System verbleiben. Insbesondere bei voneinander unabhängigen Systemen ist sicherzustellen, dass gemeinsame Fehler nicht gleichzeitig vorhanden sind oder sich nicht gleichzeitig auswirken. Um dies zu erreichen, kann z. B. Diversität als Hauptmittel eingesetzt werden. Das Diversitätsprinzip kann auf verschiedene Bereiche im Rahmen der System-/Softwareentwicklung softwarebasierter Leittechniksysteme angewendet werden. Hierzu zählen die menschliche Diversität, die Design-Diversität, die Software-Diversität und die funktionale Diversität. Detailliertere Ausführungen zur Anwendung der Diversität zwecks Vermeidung von CCFs in softwarebasierter Leittechnik sind in /GRS 17b/ enthalten.

Vermeidung der (gleichzeitigen) Aktivierung von Fehlern:

- Auch im Falle eines Vorhandenseins gemeinsamer Fehler in unabhängigen Funktionseinheiten können CCF vermieden werden, wenn diese Fehler nicht gleichzeitig aktiviert werden. Zu diesem Zweck können neben der Minimierung von gemeinsamen Fehlern, wie im vorhergehenden Abschnitt erörtert, zwei sich ergänzende Hauptansätze verwendet werden: die Vermeidung der Aktivierung von Fehlern und die Vermeidung der gleichzeitigen Aktivierung von Fehlern. Dies kann beispielsweise dadurch erreicht werden, dass die Signalpfade in unabhängigen Systemen/Funktionseinheiten unterschiedlich sind.

Vermeidung der Fehlerausbreitung:

- Der Schutz gegen die Ausbreitung von Fehlern in softwarebasierten Leittechniksystemen stellt einen weiteren beitragenden Ansatz zur Vermeidung von CCF dar. Dies betrifft sowohl auf elektrischen Effekten beruhende Fehler (z. B. Kurzschlüsse, Erdschlüsse, elektromagnetische Störgrößen) als auch Kommunikationsfehler und Datenfehler. Letztere können durch fehlerhafte Sensoren und/oder Übertragungsfehler in der Datenkommunikation verursacht werden. Maßnahmen zur Vermeidung der Ausbreitung von Fehlern können auf der Ebene der gesamten Leittechnikplattform einschließlich deren Kommunikationsarchitektur, der Ebene der Datenkommunikation zwischen Teilsystemen und auf der Ebene kommunizierender Funktionseinheiten realisiert werden.

Da Fehler in einem Softwareprogramm zum Versagen anderer Programme oder der Hardware eines Leittechniksystems führen können, ist die Begrenzung der Auswirkungen des Versagens von Software /DIN 10/ auch ein wirksames Mittel gegen die Ausbreitung von Fehlern innerhalb von Systemen.

Fehlertoleranz:

- Fehlertoleranz bezeichnet die Eigenschaft eines Systems, die ihm zugeordnete Aufgabe auch dann weiterhin korrekt zu erfüllen, wenn Fehler vorliegen. Ein solches System wird auch fehlertolerantes System genannt. Die fehlertolerante Auslegung ist abhängig von der Kategorie der auszuführenden leittechnischen Funktionen. Gemäß /BMU 15b/ sind die leittechnischen Einrichtungen, die Leittechnikfunktionen der Kategorien A und B ausführen, fehlertolerant aufzubauen. Ansätze zur fehlertoleranten Auslegung softwarebasierter Leittechniksysteme umfassen u. a. die Fehlererkennung, die Fehlersignalisierung, die Fehlereingrenzung und die Fehlermaskierung. Die Zielsetzungen dieser Ansätze werden im Abschnitt 2.3 näher erläutert. Die Wirksamkeit der Fehlertoleranzstrategien zur Vermeidung der Ausbreitung von Fehlern in einem softwarebasierten Leittechniksystem ist im Rahmen der Zuverlässigkeitsanalyse zu untersuchen. In /NEA 15/ wird hierzu vorgeschlagen zunächst eine sogenannte Fehlerausbreitungsanalyse zur Identifizierung der Auswirkungen eines sich im softwarebasierten System ausbreitenden Fehlers durchzuführen. Die darauf aufbauende Untersuchung der Wirksamkeit der eingesetzten Fehlertoleranzmethoden soll die folgenden Schritte enthalten:

  - Bestimmung der maximal möglichen Auswirkungen des Fehlers unter der Annahme nicht vorhandener oder unwirksamer Fehlertoleranzstrategien.

  - Bestimmung der wahrscheinlichsten Auswirkung des Fehlers unter der Annahme vorhandener und wirksamer Fehlertoleranzstrategien.

2.3 Fehlertoleranz in softwarebasierter Leittechnik

2.3.1 Überblick über Fehlertoleranzverfahren

Erfüllt ein fehlertolerantes System im Fehlerfall weiterhin seine Sicherheitsspezifikation und fällt dabei aus, spricht man von „fail-safe“-Fehlertoleranz /GÄR 01/. Hierdurch wird ein sicherheitsgerichtetes Ausfallverhalten des Systems bei Vorhandsein von Fehlern gewährleistet.

Bleibt ein fehlertolerantes System trotz Fehler weiterhin im Betrieb und erfüllt dabei seine ursprüngliche Spezifikation, wird dies als maskierende (engl. fault masking) Fehlertoleranz bezeichnet /GÄR 01/. In diesem Fall ist insbesondere die Signalisierung bzw. das Monitoring von Fehlfunktionen von Bedeutung, um hieraus beispielsweise Informationen zur Erkennung und Beseitigung systematischer Fehler gewinnen zu können /IAE 09/.

Zur Erzielung eines fehlertoleranten Systemverhaltens sind grundsätzlich zwei Aufgaben zu erfüllen: die Fehlererkennung und die Fehlerbehandlung. Abb. 2.2 gibt einen allgemeinen Überblick über die Fehlertoleranzverfahren aufgeteilt nach Fehlererkennung- und Fehlerbehandlungsmethoden.

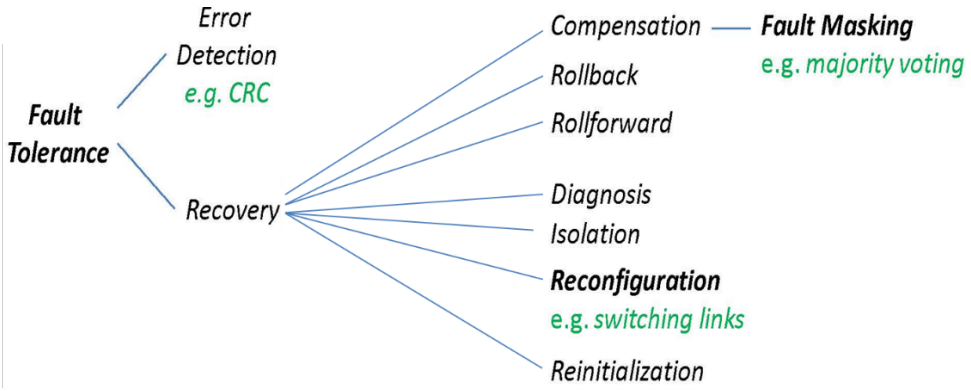

Abb. 2.2 Überblick über Fehlertoleranzprinzipien

Aufteilung nach Fehlererkennungs- (Error Detection) und Fehlerbehandlungsmethoden (Recovery) gemäß /SPE 13/

Die Fehlerbehandlung erfolgt in der Regel unter Ausnutzung des Redundanzprinzips. Sie dient dem Wiederherstellen (Recovery) eines fehlerfreien Systemzustands nach dem Auftreten/Erkennen eines Fehlers. Hierfür werden verschiedene Techniken wie beispielsweise:

- das Rücksetzen des fehlerbehafteten Systems in einen vor dem Auftreten des Fehlers fehlerfreien Systemzustand (Rollback-Methode),
- die Entfernung des fehlerhaften Zustandes aus dem System mit anschließender Bestimmung des fehlerfreien Zustandes (Rollforward-Methode),
- die **Maskierung** des aufgetretenen Fehlers (fault masking),
- die Isolierung/Eingrenzung der betroffenen fehlerbehafteten Komponente,
- die **Rekonfiguration** des Systems zwecks Überführung des fehlerbehafteten Systems in einen fehlerfreien Systemzustand und
- der Neustart des Systems

sowie verschiedene Redundanzarten eingesetzt.

Nachfolgend werden auf Grundlage der Ausführungen in /DUB 13/, /UMA 07/ und /UPO 16/ die eingesetzten Fehlererkennungs- und Fehlerbehandlungsverfahren näher beschrieben.

2.3.2 Fehlererkennungsverfahren

Fehlertoleranzstrategien beginnen wie bereits erwähnt mit der Fehlererkennung. Diese dient der **Erkennung** von Abweichungen vom beabsichtigten Verhalten (*error detection*). Die Fehlererkennung kann hierbei kontinuierlich oder periodisch erfolgen. In softwarebasierten Leittechniksystemen werden hierfür verschiedene Techniken eingesetzt, wie z. B. die Selbstüberwachung und die Überwachung durch externe Mittel (z. B. der sogenannte Watchdog-Timer).

Als Selbstüberwachungsfunktionen zählen u. a. die Selbstprüfung der Verarbeitungseinheiten z. B. beim Anlauf/Systemstart, die Einsteckü̈berwachung der Ein- und Ausgabebaugruppen, die Rechenzeitüberwachung und die Kommunikationsüberwachung zwischen den Baugruppen.

Der Watchdog-Timer überwacht den Bearbeitungszyklus in einem softwarebasierten Leittechniksystem. Er ist als Zeitzähler realisiert, der bei Funktion der überwachten Prozessoreinheit einmal zu Beginn jedes Bearbeitungszyklus zurückgesetzt wird. Hierbei ist

das Überwachungszeitintervall des Watchdog-Timers größer als die Bearbeitungszykluszeit. Ein Unterbleiben des Zurücksetzens des Watchdog-Timers durch die überwachte Prozessoreinheit wird vom System als Fehler interpretiert. Der Watchdog-Timer löst dann aus und initiiert beispielsweise einen Neustart der betroffenen Prozessoreinheit. Der Watchdog-Timer kann als Software- oder Hardware-Timer realisiert werden /URO 15/.

Plausibilitätstests gehören ebenfalls zu den eingesetzten Fehlererkennungsmethoden in softwarebasierten Leittechniksystemen.

Mit Hilfe solcher Tests wird die Gültigkeit von Ein- und Ausgaben anhand festgelegter Kriterien, wie z. B. deren Wertebereich, überprüft. Hierzu zählt beispielsweise die Messbereichsüberwachung der Eingangssignale in den auszuführenden leittechnischen Anwendungsfunktionen in softwarebasierten Leittechniksystemen.

Der Einsatz von Vergleichern in softwarebasierten Leittechniksystemen ermöglicht auch die Erkennung von Fehlern. Mit den Vergleichern werden Ergebnisse zwei redundant zueinander arbeitender Module (siehe Abschnitt 2.3.3) miteinander verglichen. Überschreitet der Unterschied zwischen den Ergebnissen einen vorgegebenen Schwellwert wird eine Fehlermeldung initiiert.

2.3.3 Fehlerbehandlungsverfahren

Die eingesetzten Fehlerbehandlungsmethoden können nach der zu tolerierenden Fehlerart (Hard- und Softwarefehler), der eingesetzten Systemwiederherstellungstechnik, und der verwendeten Redundanzart klassifiziert werden.

Das Redundanzprinzip kann hierbei auf die Hardware-, Software-, Informations- und Zeitebene angewendet werden. In Abb. 2.3 ist ein Überblick über die eingesetzten Fehlerbehandlungsmethoden im Rahmen einer fehlertoleranten Systemauslegung nach der verwendeten Redundanzart dargestellt /UPO 16/.

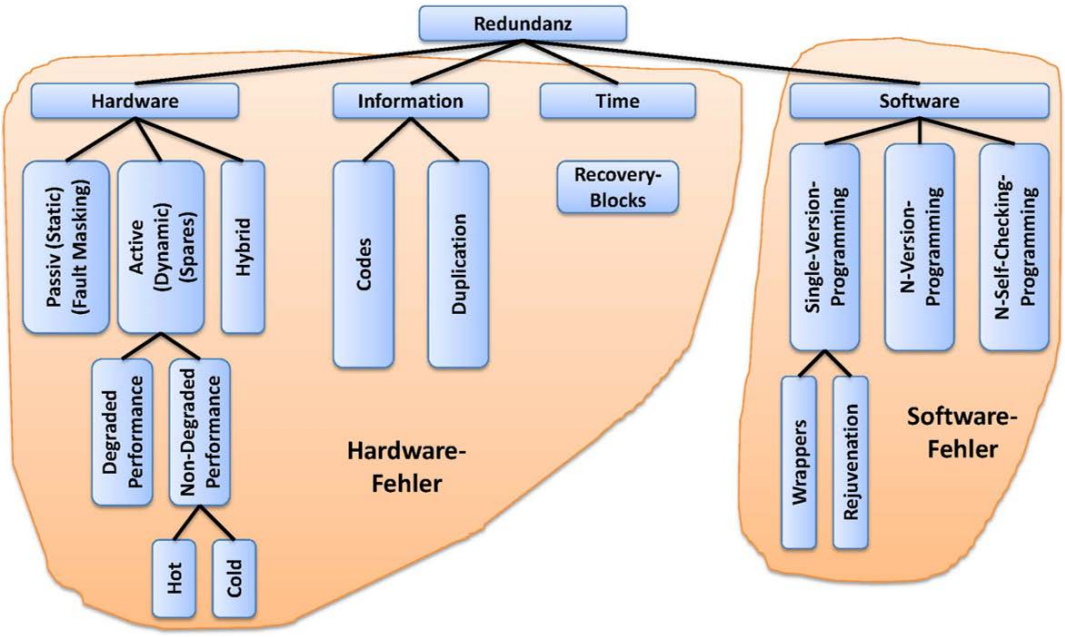

Abb. 2.3 Klassifizierung der Fehlertoleranzmethoden nach der verwendeten Redundanzart und den zu tolerierenden Fehlern gemäß /UPO 16/

Es ist zu erwähnen, dass für zeitkritische Anwendungen, wie sie in softwarebasierten Leittechniksystemen realisiert sind, die Redundanz auf der Zeitebene nicht eingesetzt wird.

Aus diesem Grund wird auf die Beschreibung der Anwendung der Redundanz auf der Zeitebene im Rahmen dieses Abschnittes verzichtet. Für die Beschreibung der Redundanz auf der Zeitebene wird auf /DUB 13/ verwiesen. Nachfolgend werden die eingesetzten Redundanzarten (Hardware, Software und Information) zur Erzielung einer fehlertoleranten Systemauslegung von softwarebasierten Leittechniksystemen beschrieben.

2.3.3.1 Redundante Hardwareauslegung

Bei Anwendung der Redundanz auf Hardwareebene zur fehlertoleranten Systemauslegung, wird zwischen passiver (statischer), aktiver (dynamischer) und hybrider (Kombination aus statischer und dynamischer) redundanter Auslegung der Hardware unterschieden.

Bei der passiven redundanten Hardwareauslegung wird redundante Hardware zur gleichzeitigen Mehrfachausführung derselben Funktion eingesetzt.

Die Architektur des Systems wird nicht verändert. Die Ausgaben der einzelnen von insgesamt *N* Redundanten werden anschließend einem Voter (*N* >2) bzw. einem Vergleicher (*N*=2) weitergeleitet, der nach einem festgelegten Prinzip eine Entscheidung für die Ausgabe trifft (siehe Abb. 2.4) bzw. eine Störung bei Abweichung meldet. Hierbei wird in der Regel beim Einsatz von Votern die Mehrheitsentscheidung (n von N) angewendet. Dies ermöglicht abhängig vom Redundanzgrad die Maskierung eines Fehlers und ggf. mehrerer Fehler, wodurch das System trotz Fehler/n weiter betrieben werden kann.

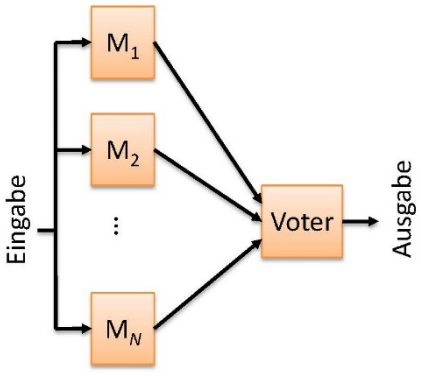

Abb. 2.4 Beispiel einer *N*-fach redundanten Hardwareauslegung nach dem passiven Redundanzprinzip (*N*>2)

M_1, M_2, … M_N sind redundante Systeme /UPO 16/

Bei einer Konfiguration mit einem Voter wie in Abb. 2.4 dargestellt, können Fehler in diesem Voter zu fehlerhaften Ausgaben führen. Um die Zuverlässigkeit des Systems diesbezüglich zu erhöhen, werden in softwarebasierten Leittechniksystemen Konfigurationen mit mehreren, redundanten Votern verwendet (siehe Abb. 2.5).

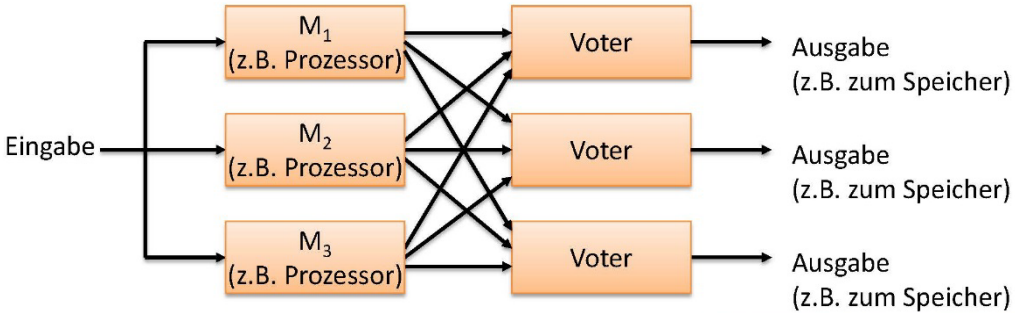

Abb. 2.5 Beispielskonfiguration einer passiven redundanten Hardwareauslegung mit drei redundanten Systemen und drei redundanten Votern /UPO 16/

Gemäß /BMU 15b/ sind die leittechnischen Einrichtungen so aufzubauen, dass die in den aktiven Einrichtungen des Sicherheitssystems vorgegebene Redundanz gewahrt bleibt.

Die Hardware für Funktionseinrichtungen der Kategorie A wird gemäß dieser Anforderung aus /BMU 15b/ redundant ausgelegt. Hierbei wird das Prinzip der passiven Redundanz angewendet.

Bei der aktiven redundanten Auslegung der Hardware führt im Gegensatz zur passiven redundanten Auslegung nur eine der Redundanten, die aktive Redundante, die Funktion aus. Die übrigen Redundanten, Reserveredundanten, werden aktiviert, wenn Fehler bei der aktiven Redundanten erkannt werden. Zuvor wird die fehlerhafte aktive Redundante deaktiviert. Hierdurch wird die Konfiguration des Systems geändert. Das Aktvieren und Deaktivieren von Redundanten wird zentral über ein Fehlererkennungs- und Rekonfigurationsmodul gesteuert (siehe Abb. 2.6).

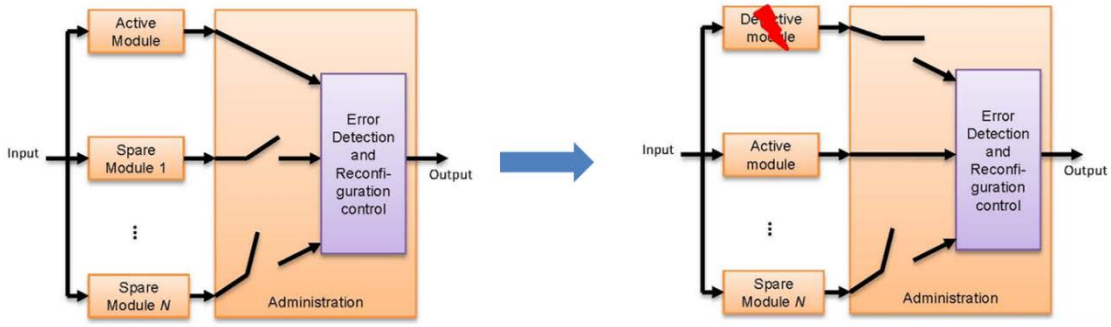

Abb. 2.6 Beispiel einer *N*-fachen redundanten Hardwareauslegung nach dem aktiven Redundanzprinzip

Nach Erkennung der fehlerhaften Redundanten wird diese deaktiviert und eine Reserveredundante aktiviert. /UPO 16/

Die Konzepte der aktiven Redundanz unterscheiden sich nach dem Bereitschaftszustand der Reserveredundanten vor deren Aktivierung. Im sogenannten Hot-Standby-Modus befinden sich die Reserveredundanten im Betrieb und führen bereits vor der Aktivierung die Funktion parallel zur aktiven Redundanten aus. Dies ermöglicht eine schnellere Rekonfiguration des Systems bei Vorliegen eines Fehlers. Im Cold-Standby-Modus sind die Reserveredundanten vor der Aktivierung außer Betrieb.

Ein Anwendungsbeispiel für eine aktive redundante Hardwareauslegung softwarebasierter Leittechniksysteme stellen Backupsysteme dar. Sie werden beispielsweise aus Verfügbarkeitsgründen in leittechnischen Systemen zur Ausführung von Funktionen der Kategorie C (z. B. Regelungsaufgaben) verwendet.

2.3.3.2 Redundante Softwareauslegung

Die fehlertolerante Softwareauslegung zielt darauf ab, Ausfälle aufgrund von Softwarefehlern zu vermeiden. Sie basiert im Wesentlichen auf dem Prinzip der Mehrfachprogrammierung (engl. N-Version Programming) oder der mehrfachen Selbstüberprüfung (N-Self-Checking Programming), wenn mehrere Versionen des Programms (Software-Diversität) vorhanden sind. Abweichend dazu werden die Wrapper-Methode und der Programmneustart als Fehlertoleranztechniken eingesetzt, wenn lediglich eine Version des Programms verfügbar ist.

Der Neustart eines Prozesses bzw. einer Software wird zur Freigabe geblockter Ressourcen oder im Falle eines transienten Fehlers als Fehlertoleranztechnik verwendet. Der Neustart kann auf der Prozessorebene oder auf der Anwendungsebene (Neustart einer Anwendung/eines Prozesses) erfolgen. Der Neustart kann hierbei zeitbasiert, z. B. nach einem festgelegten Zeitintervall) oder vorhersagenbasiert durchgeführt werden. Im letztgenannten Fall basiert der Neustart auf einer Beobachtung der allokierten Ressourcen, z. B der Speichernutzung, und der Erkennung von Tendenzen. Basierend auf den beobachteten Trends kann ein Neustart eingeleitet werden.

Bei der Wrapper-Methode wird das auszuführende Programm durch einen sogenannten Wrapper gekapselt (siehe Abb. 2.7). Hierbei handelt es sich um eine Software, die als Schnittstelle zum auszuführenden Programm fungiert, beispielsweise zwecks Überwachung des auszuführenden Programms.

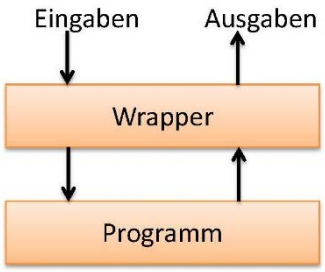

Abb. 2.7 Prinzipbild der Wrapper-Methode zur fehlertoleranten Softwareauslegung /UPO 16/

Eingabedaten zum auszuführenden Programm werden durch den Wrapper überwacht, der sie bei Erfüllung von Vorbedingungen (z. B. Gültigkeit des Wertebereiches) entweder an das auszuführende Programm zur Weiterverarbeitung übergibt oder eine Ausnahme

signalisiert. Ausgabedaten des auszuführenden Programms werden vor deren Ausgabe Plausibilitätstests unterzogen. Durch diese Plausibilitätstests werden im Wrapper die Eigenschaften der Programmergebnisse bzw. deren Plausibilität überwacht. Anhand eines derartigen Plausibilitätstests wird beispielsweise der Wertebereich der Ausgabedaten oder die Abweichungen der Ausgabedaten zu vorherigen Ausgabewerten geprüft. Bei erfolgreichem Plausibilitätstest werden die Ausgabedaten des auszuführenden Programms ausgegeben, andernfalls signalisiert der Wrapper eine Ausnahme. Die Wrapper-Methode eignet sich insbesondere zur Vorbeugung der Auswirkungen von bekannten Fehlermodi der eingesetzten Software, wie z. B. fehlender Schutz vor einem Speicherüberlauf oder fehlerverursachende Eingabedatensätze. Dementsprechend hängt die Wirksamkeit der Wrapper-Methode insbesondere von den verfügbaren Informationen zu den typischen Fehlermodi der zu überwachenden Software ab. /UMA 07/

Bei der auf Mehrfachprogrammierung basierenden fehlertoleranten Technik wird das Prinzip der Softwarediversität zur Maskierung von Programmierfehlern verwendet. Es sind demnach N diversitäre Versionen einer Software vorhanden, die z. B. durch verschiedene Programmierteams auf der Basis einer gleichen Spezifikation entwickelt und implementiert wurden. Es wird angenommen, dass verschiedene Teams nicht die gleichen Fehler in das Programm einbringen. Die N Versionen des Programms werden im Anforderungsfall parallel ausgeführt. Sie verarbeiten dann dieselben Eingabedaten und liefern jeweils ein Programmergebnis. Die N Programmergebnisse werden einem Voter zugeleitet, der nach einem festgelegten Algorithmus eine Entscheidung, z. B. Mehrheitsentscheidung, für die Programmausgabe trifft (siehe Abb. 2.8).

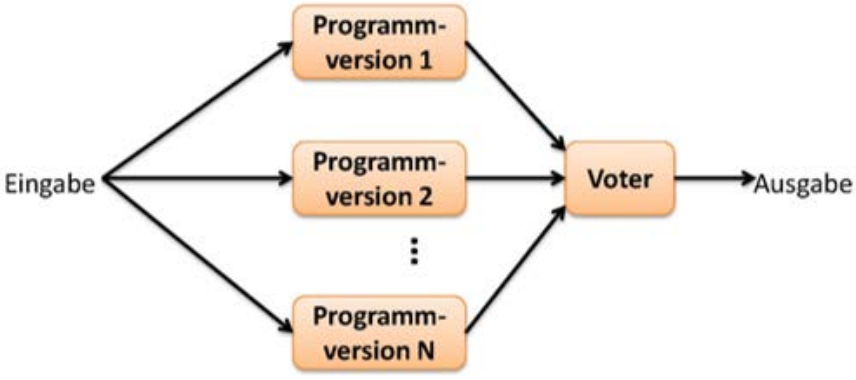

Abb. 2.8 Prinzip der Mehrfachprogrammierung zur fehlertoleranten Auslegung der Software /UPO 16/.

Die mehrfache Selbstüberprüfung entspricht im Wesentlichen der Mehrfachprogrammierungstechnik. Es sind wie bei der Mehrfachprogrammierung N diversitäre Versionen des Programms vorhanden. Jede der N Programmversionen besitzt jedoch einen eigenen Akzeptanztest zur Plausibiltätsprüfung (Plausibilitätstest) der jeweiligen Programmergebnisse. Die Auswahllogik im Voter entscheidet nur auf Grundlage der gültigen Ergebnisse der jeweiligen Programmversionen (siehe Abb. 2.9).

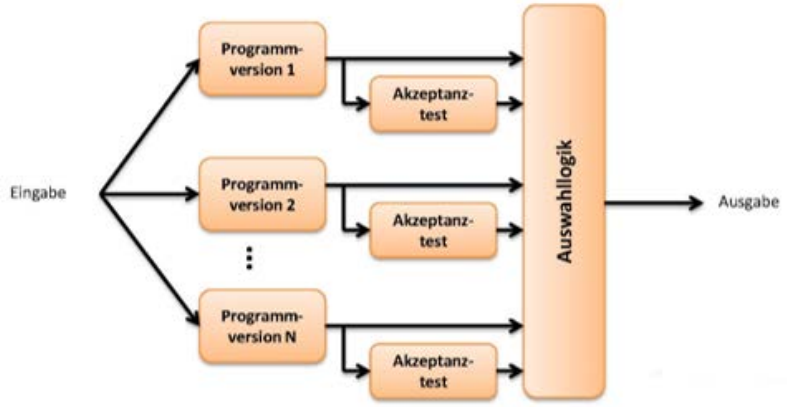

Abb. 2.9 Prinzip der mehrfachen Selbstüberprüfung zur fehlertoleranten Softwareauslegung /UPO 16/

Bei der sogenannten Recovery-Block-Programming-Methode werden ebenfalls N-Programmversionen eingesetzt. Im Gegensatz zu den Varianten N-Version-Programming und N-Self-Checking-Programming wird jedoch nur eine der N-Programmversionen, das sogenannte Primärprogramm, ausgeführt. Die Ergebnisse des primären Programms werden einem Akzeptanztest unterzogen. Bei nicht erfolgreichem Akzeptanztest der Ergebnisse des primären Programms werden die restlichen N-1 Programmversionen, sogenannte Alternativprogramme, solange nacheinander ausgeführt bis die festgelegten Akzeptanzkriterien von einer der N-1 Programmversionen erfüllt sind. Werden die Akzeptanzkriterien von keiner der N Programmversionen erfüllt, wird die Ausführung des Programms mit einer Fehlermeldung unterbrochen. Der sequenzielle Aufruf verschiedener Programmversionen bei der Recovery-Block-Methode ist mit einem erheblichen Zeitaufwand verbunden. Die Recovery-Block-Methode eignet sich daher nicht als fehlertolerante Technik für zeitkritische Anwendungen wie in Kernkraftwerken, da bei diesen eine Echtzeitreaktion erforderlich ist.

In softwarebasierten Leittechniksystemen wird das Prinzip der Softwarediversität als mögliche fehlertolerante Maßnahme eingesetzt /GRS 17b/.

2.3.3.3 Redundanz auf der Informationsebene

Das Prinzip der Informationsredundanz wird zur fehlertoleranten Auslegung der Datenübertragung eingesetzt. Datenfehler können beispielsweise bei der Übertragung von Daten von einer Systemkomponente zu einer anderen oder von einem System zu einem anderen auftreten. Um solche Fehler zu tolerieren, werden den Daten redundante Informationen hinzugefügt, dies wird als Informationsredundanz bezeichnet. Die gebräuchlichste Form der Informationsredundanz ist die Kodierung, bei der den Daten Prüfbits hinzugefügt werden, die es ermöglichen, die Richtigkeit der Daten zu überprüfen, bevor sie verwendet werden. In einigen Fällen wird sogar die Korrektur der fehlerhaften Datenbits ermöglicht. Hierzu werden in softwarebasierten Leittechniksystemen verschiedene Kodierungen zur Fehlererkennung und Fehlerkorrektur verwendet wie z. B. die Checksummenprüfung und die zyklische Redundanzprüfung (engl. Cyclic Redundancy Check (CRC)).

2.4 Zuverlässigkeitsanalysemethoden in softwarebasierter Leittechnik

Das Erfordernis der Durchführung einer Zuverlässigkeitsanalyse softwarebasierter Leittechniksysteme resultiert aus den Auslegungsanforderungen für softwarebasierte Leittechniksysteme wie sie beispielsweise in Abschnitt 2.2 ausgeführt sind. Ziel dieser Analyse ist die Gewährleistung einer hohen Softwarequalität und einer hohen Zuverlässigkeit der Hardware sicherheitsrelevanter Leittechnik. Basierend auf den Ausführungen in /DIN 10/ zu einzusetzenden Prüfverfahren wird die Analyse durchgeführt, um den Grad der Anforderungserfüllung zu messen, die Qualität der betrachteten Software festzustellen und etwaige Fehler aufzudecken. Laut /DIN 02/ sollte zudem der Umfang der vorbeugenden Instandhaltung durch die Anwendung eines systematischen Analyseverfahrens bestimmt werden. Hierzu kann beispielsweise die Failure Mode and Effects Analysis, die Fehlerbaumanalyse oder die Anwendung eines auf einer Zuverlässigkeitsberechnung basierenden Instandhaltungsmodells benutzt werden.

Im Folgenden wird der aktuelle Stand von Wissenschaft und Technik zur Analyse der Zuverlässigkeit softwarebasierter Leittechnik beschrieben. In der Literatur findet sich eine Vielzahl von Analysemethoden mit unterschiedlichen Zielsetzungen. Bei der Vorgehensweise wird u. a. zwischen qualitativer und quantitativer Analyse unterschieden.

Bei der qualitativen Analyse wird die Struktur der Software hinsichtlich Fehlereffekten untersucht, um Ausfallmechanismen, Ursachen und Auswirkungen zu ermitteln. Hierzu eignet sich die Software Failure Mode and Effects Analysis oder/und kombinatorische Methoden, wie beispielsweise die speziell für Software angepasste Fehlerbaumanalyse (Software Fault Tree Analysis/SFTA). Die kombinatorischen Methoden sollen beispielsweise die minimalen Schnittmengen von verschiedenen Ursachen ermitteln, die zu einem unerwünschten Ereignis führen können.

Ziel einer quantitativen Analyse ist die Bestimmung einer Ausfallwahrscheinlichkeit bzw. Ausfallrate der Software. Im Gegensatz zur Hardware lässt sich bei der Software eine Ausfallrate bzw. eine Ausfallwahrscheinlichkeit jedoch nicht auf Basis von Ausfallraten bzw. Ausfallwahrscheinlichkeiten von Komponenten bestimmen. Die Ausfallwahrscheinlichkeit einer Leittechnikfunktion aufgrund von einem oder mehreren Softwarefehlern wird häufig abhängig von der Zielsetzung der Zuverlässigkeitsanalyse auf Basis von Annahmen (u. a. Expertenschätzungen, regulatorische Anforderungen) bestimmt. Hierfür werden z. B. Erkenntnisse aus der generischen Betriebserfahrung mit softwarebedingten Ausfällen leittechnischer Funktionen oder aus dem Validierungs- und Verifizierungsprozess der Software herangezogen. Das Prinzip besteht darin, Ereignisse (z. B. Überlastung der Systemressourcen /SAL 10/ oder fehlerhafte Kommunikationsdaten zwischen Verarbeitungsmodulen /JOC 15/) zu ermitteln bzw. festzulegen, welche zu einem Ausfall der Software führen können. Damit soll ein Vorhersagemodell für den Ausfall der Software abgeleitet werden. Eine solche Vorgehensweise ist beispielsweise in /JOC 15/ dokumentiert. In anderen Ansätzen zur Schätzung der Ausfallrate bzw. Ausfallwahrscheinlichkeit eines softwarebedingten Ausfalls wird für die Entwicklung eines

Vorhersagemodells ein Zusammenhang zwischen Ausfall der Software und zugehörigen Eingangsdaten angenommen /SAL 10/.

Im Rahmen des Projektes wird der Schwerpunkt auf qualitative Methoden gelegt. Es wird daher der aktuelle Stand von Wissenschaft und Technik von qualitativen Analysemethoden beschrieben.

Für eine vollständigere Übersicht über Methoden der Zuverlässigkeits- und Sicherheitsanalyse sei auf die Forschungsvorhaben der GRS zum Themenkomplex „Einsatz und Bewertung von softwarebasierter Leittechnik in Kernkraftwerken“ verwiesen /GRS 15/, /GRS 16/.

Die im Folgenden beschriebenen Methoden zur Softwarezuverlässigkeitsanalyse ermöglichen vor allem eine bessere Fehlervorhersage bei softwarebasierten Leittechniksystemen, so dass noch vorhandene Fehler entfernt werden können. Außerdem können die Erkenntnisse aus Zuverlässigkeitsanalysen in die fehlertolerante Auslegung des softwarebasierten Leittechniksystems eingehen.

2.4.1 Software Failure Modes and Effects Analysis

Die „Failure Modes and Effects“-Analyse ist eine weit verbreitete Methode zur Zuverlässigkeitsbewertung von Hardware in sicherheitskritischen Systemen. Der Fokus der FMEA liegt darin, die Auswirkungen von möglichen Fehlern in einem System festzustellen. Hierzu wird u. a. die Ausbreitung der Fehler durch das System betrachtet. Mit der FMEA werden alle möglichen Ausfallarten eines Systems dokumentiert, deren Auswirkungen auf das System ermittelt und die ermittelten Ausfallarten nach der Schwere ihrer Auswirkungen sortiert. Die Ergebnisse einer FMEA werden in eine Tabelle eingetragen, die die jeweilige Ausfallart, deren Auswirkungen und die daraus folgende Analyse enthält. /TRI 04/ /BEH 09/

Die FMEA ist ein induktives Verfahren (von einer niedrigen zu einer höheren Systemebene), d. h. die Analyse geht von den Fehlerarten einer Basiskomponente aus und untersucht dann die Einflüsse der Fehlerarten auf die nachfolgenden Systemebenen /IEC 06/. Dabei wird angenommen, dass jede Basiskomponente in einen Fehlerzustand geraten kann.

Zusätzlich zur FMEA, welche die Fehlerzustandsart- und –auswirkungsanalyse beinhaltet, kann auch noch eine Bedeutungsanalyse (Analyse der Kritikalität) durchgeführt werden. Bei dieser sogenannten „Failure Modes, Effects and Criticality Analysis“ (FMECA), wird aus der Schwere der Auswirkung S und der Eintrittswahrscheinlichkeit P das Ausmaß einer Ausfallartauswirkung (Risiko R = S·P) bestimmt.

Zur Anwendung der FMEA auf die Hardware sind mehrere Standards in der Wehrtechnik und der Industrie veröffentlicht worden wie z. B. /DOD 80/ und /AIA 08/.

Bisher wurde noch kein spezifischer Standard oder Leitfaden zur Anwendung der FMEA für die Zuverlässigkeitsanalyse softwarebasierter Leittechniksysteme veröffentlicht. Das in /IEC 06/ beschriebene Verfahren zur Anwendung der FMEA stimmt zum einen gemäß /STU 02/ in vielen Punkte mit anderen wichtigen FMEA-Normen und Richtlinien wie der Standard /DOD 80/ und der in der Automobilindustrie verwendete Standard /SAE 00/ überein. Zum anderen ist die Beschreibung der FMEA-Methode in /IEC 06/ allgemein gehalten. Gemäß /STU 02/ stellt daher der Standard /IEC 06/ einen guten Ausgangspunkt zur Anwendung der FMEA für die Zuverlässigkeitsanalyse softwarebasierter Leittechniksysteme dar.

Das prinzipielle Vorgehen zur Durchführung einer FMEA kann auf Basis der Ausführungen in /IEC 06/ wie folgt beschrieben werden:

Ermittlung der Systemstruktur

Bevor mit der Analyse begonnen werden kann, müssen die Systembestandteile mit ihren Aufgaben und Funktionen ermittelt werden. Dabei sollten auch die logischen Verknüpfungen zwischen den Bestandteilen sowie die Eingangs- und Ausgangsgrößen des Systems bekannt sein. Für das bessere Verständnis kann es auch hilfreich sein, Diagramme der Systemstruktur zu erzeugen.

Verständnis der Funktionen und Anforderungen

Für die eigentliche Analyse der Fehlerauswirkungen ist es wichtig, die Anforderungen an die Funktionalität und Sicherheit eines Systems zu verstehen.

Festlegung von Erfolgs- und Ausfallkriterien

Um die Auswirkungen von Fehlern auf das Systemverhalten bestimmen zu können, ist eine genaue Definition von erfolgreichem und fehlerhaftem Systemverhalten festzulegen. Dabei lassen sich bereits zu erwartende Ausfallarten ermitteln.

Festlegung der Systembegrenzung

Im nächsten Schritt wird festgelegt, welche Teile des Gesamtsystems untersucht werden sollen und können. Dabei kann es sich als sinnvoll erweisen, eine Begrenzung des Systems vom funktionellen Gesichtspunkt aus festzulegen und dadurch die Anzahl von Eingabe- und Ausgabeverknüpfungen zu begrenzen.

Zerlegung des Systems in Teilsysteme bzw. Ebenen

Eine Zerlegung des Gesamtsystems in Teilsysteme ist im Rahmen der Analyse möglichst vorzunehmen. Für modular aufgebaute Systeme wie softwarebasierte Leittechniksysteme ist eine derartige Zerlegung sinnvoll. Zudem ist die für die Analyse des Systems gewählte Gliederungsebene bzw. Abstraktionsebene festzulegen.

Bestimmung der Fehlerzustände bzw. Fehlzustände und Fehlerwirkung

In diesem Schritt sind für jedes zu betrachtende Element die potenziellen Fehlerzustände zu definieren.

Bestimmung der Fehlerauswirkungen einzelner Fehlerzustände bzw. Fehlzustände

Für jedes zu betrachtende Element werden die aus den Fehlerzuständen resultierenden Fehlerauswirkungen beschrieben.

Erstellung des Berichtes

Zuletzt wird eine Tabelle erstellt, um die Ausfallarten, mögliche Ausfallursachen, lokale Auswirkungen, Auswirkungen auf die Ziel-Einheit, Erkennungsmethoden etc. festzuhalten. Im Standard /IEC 06/ findet sich beispielsweise eine Vorlage dazu.

Die zuvor beschriebene Vorgehensweise zur Durchführung einer FMEA kann sowohl auf die Hardware als auch auf die Software softwarebasierter Leittechniksysteme angewandt werden. Software weist jedoch im Vergleich zur Hardware einige Besonderheiten auf, so dass die SFMEA in einigen Punkten angepasst werden muss.

Im Unterschied zur Hardware zeichnet sich Software u. a. dadurch aus, dass

- sie im engeren Sinne nicht ausfällt wie Hardware, Softwarefehler führen u. a. zu einem nichtanforderungsgemäßen Systemverhalten,
- sie kein Alterungsverhalten aufweist, so dass sich mit statistischen Methoden keine Ausfallwahrscheinlichkeiten ausgehend von Alterungsphänomenen wie bei Hardwarekomponenten bestimmen lassen,
- ihre einzelnen Komponenten oftmals nicht voneinander unabhängig sind und häufig sehr komplexe Abhängigkeiten (u. a. Funktionsweise der Hardware, Ablaufumgebung, Speichermanagement, Konfigurationsmanagement) voneinander aufweisen können.

In /STU 02/ wird diesbezüglich auf die sich daraus ergebenden Unterschiede und Gemeinsamkeiten zwischen einer Hardware- und einer Software-FMEA eingegangen. Diese sind in der Tab. 2.1 zusammengefasst.

Tab. 2.1 Gegenüberstellung der Hardware- und der Software-FMEA nach /STU 02/

Hardware-FMEA	**Software-FMEA**
Kann auf Funktions- und/oder Komponentenebene durchgeführt werden.	Kann oftmals nur auf Funktionsebene und für aus einzelnen unabhängigen Bausteinen bestehende Software (z. B. Anwendersoftware zur Erstellung der Funktionspläne für Leittechnikfunktionen) durchgeführt werden
Gilt für ein System, das als frei von ausgefallenen Komponenten betrachtet wird.	Gilt für ein System, bei dem davon ausgegangen wird, dass es latente Softwarefehler enthält, die bei entsprechender Aktivierung zum Ausfall führen können.
Postuliert Ausfälle von Hardware-Komponenten aufgrund von Alterung, Verschleiß oder Belastung	Postuliert Ausfälle von Softwarekomponenten aufgrund von potenziellen Softwarefehlern.
Analysiert die Folgen dieser Ausfälle auf Systemebene.	Analysiert die Folgen dieser Ausfälle auf Systemebene.

Die Ziele der Software FMEA lassen sich hierbei wie folgt definieren /STU 02/:

- Identifikation von design- und prozessbedingten Fehlerzuständen,
- Analyse der Auswirkungen der Fehlerzustände auf die Systemfunktion,
- Aufdecken der Ursachen der Fehlerzustände,
- Identifikation, Implementierung und Dokumentation der erfolgversprechenden Korrekturmaßnahmen und
- ggf. Priorisierung der Korrekturmaßnahmen zu bestimmten Fehlerzuständen unter Berücksichtigung der Fehlerauswirkung und Auftretenswahrscheinlichkeit (SFMECA).

Weiterhin sind die folgenden Aspekte im Rahmen der Anwendung einer Software FMEA zu betrachten:

- In Bezug auf die zugrunde zulegende Abstraktionsebene kann bei einer Software FMEA die Analyse beispielsweise auf der Ebene des Modells, der Funktionsblöcke oder des Programmcodes durchgeführt werden. Die niedrigste Ebene (Programmcode) liefert zwar die detailliertesten Analyseergebnisse, sie ist jedoch mit dem größten Analyseaufwand verbunden. In manchen Fällen ist es aufgrund dieses hohen Analyseaufwands sinnvoller, die Analyse auf einer anderen Abstraktionsebene durchzuführen.
- Abhängig von der gewählten Abstraktionsebene kann es sich bei den zu ermittelnden Fehlerzuständen beispielsweise um Fehlerzustände der Funktionsblöcke handeln oder bei der Analyse des Programmcodes um Fehler bei der Variablendefinition, Programmanweisungen, Funktionen etc. handeln.
- Die SFMEA betrachtet nur einzelne Fehlerzustände, nicht jedoch die Auswirkungen von gleichzeitig auftretenden Fehlerzuständen. Somit ist es möglich, dass eine SFMEA ergibt, dass eine Software fehlerfrei ist, obwohl diese in der Realität Fehlerzustände aufweist. /NEU 15/ /BEH 09a/
- Im Falle einer Software FMEA auf der Programmcodeebene kann sich die Ausbreitung eines Fehlers durch das System aufgrund komplexer Abhängigkeiten als kompliziert herausstellen. Einige Autoren verwenden daher zur besseren Übersichtlichkeit der Fehlerausbreitung eine graphische Darstellung /ABU 08/.

2.4.2 Software Fault Tree Analysis (SFTA)

Die Fault Tree Analysis (FTA, dt. Fehlerbaumanalyse bzw. Fehlzustandsbaumanalyse) ist ein bewährtes Werkzeug zur Analyse von Hardwaresystemen; sie kann sowohl qualitativ als auch quantitativ durchgeführt werden. Gemäß /DIN 07/ ist ein Fehlzustandsbaum bzw. ein Fehlerbaum /DIN 90/ ist eine geordnete graphische Darstellung der Bedingungen und Faktoren, die den Eintritt eines unerwünschten Ergebnisses verursachen oder dazu beitragen; dieses wird mit Hauptereignis /DIN 07/ bzw. Topereignis /DIN 90/ bezeichnet. Zur Anwendung der FTA sind mehrere Standards veröffentlicht worden /DIN 81/, /DIN 90/, /DIN 07/. Die FTA kann auch zur Analyse von Softwareausfällen (SFTA), sowie der Wechselwirkungen zwischen Software und Hardware verwendet werden.

Die Software Fault Tree Analysis bedient sich einer deduktiven Vorgehensweise (von einer höheren Systemebene zu einer niedrigeren Systemebene) d. h. von der Fehlerwirkung (Hauptereignis bzw. Topereignis) ausgehend werden die jeweiligen Ursachen der Fehlerzustände/Fehlzustände aufgezeigt, die die Fehlerwirkung hervorgerufen haben können. Die Fehlerzustände/Fehlzustände werden so lange zurückverfolgt, bis die ursprüngliche Fehlerursache gefunden worden ist. Die Ziele der SFTA lassen sich wie folgt definieren /DIN 07/:

- Analyse der Ursachen und Kombinationen von Ursachen, die zum Hauptereignis führen,
- Aufdecken potenzieller Ausfallarten, die die Ausfallwahrscheinlichkeit oder Nichtverfügbarkeit des Systems erhöhen,
- Aufdecken potenzieller Ausfallarten, die ein Sicherheitsrisiko bewirken können und
- Ergänzung zu anderen Analysemethoden, wie beispielsweise FMEA oder Markov-Modellen (siehe Abschnitt 2.4.4).

Für die Anwendung der SFTA wird die folgende Vorgehensweise nach /DIN 07/ empfohlen:

Festlegung des Untersuchungsziels

Zunächst wird festgelegt, ob eine qualitative oder quantitative FTA durchgeführt werden soll. Bei der qualitativen FTA können Ereignisse oder Fehlzustände ermittelt werden, die einen direkten Systemausfall verursachen können und so können gegebenenfalls Fehlzustände entschärft werden. Bei der quantitativen FTA können Ausfallwahrscheinlichkeiten bestimmt werden. Letztere kann nur durchgeführt werden, wenn Wahrscheinlichkeiten der Primärereignisse bzw. Basisereignisse bekannt sind.

Verständnis der Funktionen und Arbeitsweise des Systems

Zur Auswahl einiger relevanter Fehlerwirkungen (Haupt- bzw. Topereignisse) ist ein hohes Maß an Systemkenntnis erforderlich, da nicht alle Fehlerwirkungen offensichtlich sind. Auch für die weitere Erstellung des Fehlerbaums bzw. Fehlzustandsbaums, der ein Modell der Struktur und Funktionalität des Systems ist, sollten detaillierte Kenntnisse des Systems vorhanden sein.

Festlegung des Haupt- bzw. Topereignisses und Konstruktion des Fehlzustandsbaums

Der erste Schritt bei der Erstellung des Fehlzustandsbaums ist die Festlegung des Haupt- bzw. Topereignisses, d. h. der Fehlerwirkung. Anhand dieses Hauptereignisses werden dann in tieferen Ebenen die verursachenden Ereignisse des Hauptereignisses gesucht. Der Fehlzustandsbaum besteht aus zwei unterschiedlichen Elementen, den Ereignissen und den Gattern. Ereignisse sind die Folgen eines Fehlers, die sich weiter verzweigen können.

Primärereignisse sind Ereignisse, die sich nicht weiter aufteilen und damit den Fuß des Fehlerbaums bilden /DIN 07/. Das Gatter steht für die logische Verknüpfung zwischen dem Eingangsereignis und dem Ausgangsereignis. Die Symbole des Fehlerzustandsbaums sind in /DIN 07/ und in /NUR 81/ beschrieben.

Untersuchung des logischen Aufbaus des Fehlzustandsbaums

Nach der Erstellung des Fehlzustandsbaums muss dieser auf Korrektheit und Vollständigkeit überprüft werden.

Erstellung des Berichtes

Zuletzt wird ein Bericht erstellt, um die Entwickler des untersuchten Softwaresystems über Korrekturmaßnahmen zu informieren.

Abb. 2.10 zeigt beispielhaft einen Ausschnitt der ersten beiden Ebenen einer durchgeführten SFTA für Instrumentenanzeigen in Flugzeugen /LUT 98/. Die baumartige Struktur ist deutlich erkennbar. Die einzelnen Ereignisse sind in Rechtecken dargestellt, Ihre logische Verknüpfung ist in einem ODER-Gatter angezeigt. Ausgehend von dem Haupt- bzw. Topereignis „Cue displayed when should be removed“ (erste Ebene) wird zeitlich und unter Berücksichtigung der Wirkungskette der beteiligten Komponenten rückwärtsgegangen und analysiert welches Ereignis oder welche Ereigniskombination(en) dieses Topereignis hervorrufen könnte (z. B. „Pitch Failure not detected“; zweite Ebene). Diese Analyse wird in weiteren, hier nicht dargestellten Ebenen, fortgeführt, bis der Fuß des Fehlerbaums erreicht ist. /LUT 98/

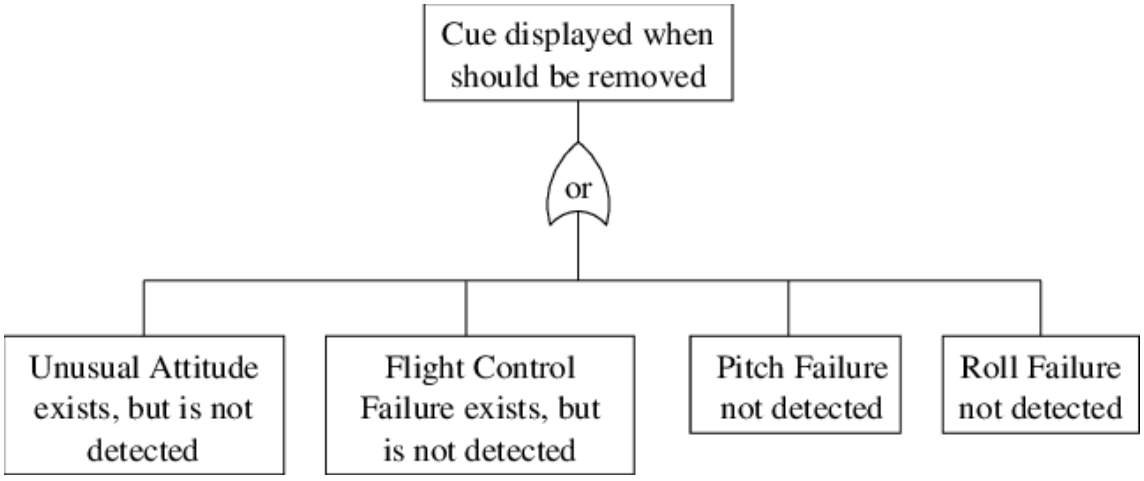

Abb. 2.10 Ausschnitt aus einer SFTA für Instrumentenanzeigen in Flugzeugen /LUT 98/.

FTA weisen vielfältige Vorteile auf. So ermöglicht die grafische Darstellungsweise beispielsweise einen schnellen Überblick über die vorliegenden Zusammenhänge. Anders als die monokausal vorgehende FMEA kann die FTA zudem darstellen, dass ein Haupt- bzw. Topereignis verschiedene Primärereignisse als Ursache haben kann. Auf diese Weise können mit der FTA die Auswirkungen des Eintritts von mehreren Primär- bzw. Basisereignissen veranschaulicht werden. Darüber hinaus kann eine FTA auf Vollständigkeit überprüft werden, indem man die in ihr dokumentierten Systemkomponenten mit der Systemarchitektur vergleicht. /JOH 20/

Nachteil einer FTA der modular aufgebauten Software besteht darin, dass die Modellierung der Einzelausfälle von Komponenten in den redundanten Strukturen insbesondere mit Berücksichtigung der CCF-Ausfälle sehr umfangreich werden kann. Des Weiteren ist es erforderlich, die potenziellen Abhängigkeiten zwischen Softwaremodulen und zwischen Software und Hardware bei der Modellierung konsistent zu berücksichtigen.

Die Fehlerbaumanalysemethode wurde bereits im Rahmen von GRS Forschungsarbeiten insbesondere zur Analyse von verschiedenen Architekturen softwarebasierter Leittechniksysteme hinsichtlich deren Ausfallverhalten, wie z. B. in /GRS 18/, erprobt, so dass in der GRS bereits Erfahrungen hinsichtlich der Eignung dieser Methode zur Zuverlässigkeitsbewertung softwarebasierter Leittechnik gewonnen werden konnten.

2.4.3 Event Tree Analysis (ETA)

Die Event Tree Analysis (ETA, dt.: Ereignisbaum-Analyse oder Ereignisablaufanalyse) ähnelt der Fehlerbaumanalyse, jedoch werden nicht die Ursachen eines Ereignisses, sondern seine Auswirkungen analysiert. Bei dieser Analyse werden Ereignisse, die zum Ausfall von erforderlichen Sicherheitsfunktionen führen (Hauptereignisse der FTA), ermittelt. Laut /KRU 12/ werden ETA bevorzugt im Bereich der Sicherheitstechnik im Bereich der funktionalen Sicherheit genutzt. Zudem werden ETA im Bereich der Kerntechnik eingesetzt /AVE 14/. Es ist laut /LAW 93/ jedoch keine Literatur über die Anwendung von ETA für die Softwarezuverlässigkeitsanalyse bekannt. Auch eine vertiefte Literaturrecherche im Rahmen dieses Vorhabens führte zu keiner gegenteiligen Erkenntnis. Daher wird die ETA im Folgenden lediglich kurz erläutert.

Im Vergleich zur FTA (deduktives Vorgehen) wird bei einer ETA in umgekehrter Reihenfolge vorgegangen und eine induktive Analyse durchgeführt. Mithilfe dieser Analysemethode können Störungen und Ausfälle eines technischen Systems untersucht und die möglichen Folgen eines auftretenden Fehlers bestimmt werden. Ähnlich wie FTAs können ETAs qualitativ oder quantitativ durchgeführt werden. Mithilfe einer qualitativen ETA können beispielsweise die Auswirkungen eines aufgetretenen Fehlers ermittelt werden. Bei einer quantitativen Analyse werden zudem die Wahrscheinlichkeiten der aus dem untersuchten Fehler entstandenen Auswirkungen bestimmt. Während des Analyseverfahrens werden grafische Symbole genutzt, die eine baumartige Struktur ähnlich einer FTA ergeben anhand derer sich die Signal- und Wirkungspfade ablesen lassen. /TÜV 20/

2.4.4 Markov-Modell

Das Markov-Modell ist ein Zustandsübergangsmodell und wird zur Modellierung von sich zufällig verändernden Systemen genutzt. In den meisten Fällen werden Markov-Modelle für quantitative Analysen genutzt. Mit ihrer Hilfe lassen sich Vorhersagen über das Eintreten zukünftiger Zustände voraussagen. Dies betrifft beispielsweise Ausfallwahrscheinlichkeiten, Wahrscheinlichkeiten von Zustandsänderungen und Ausfall- und Reparaturraten von Systemen. Systemzustände und Wahrscheinlichkeiten für den Übergang zwischen diesen Zuständen (Übergangswahrscheinlichkeiten) werden mit Hilfe des Modells abgebildet. Der Begriff „Zustand“ kann sich dabei beispielsweise auf die Funktionsfähigkeit oder möglicherweise gegenwärtig durchgeführte Reparaturen von Komponenten beziehen. Die Wahrscheinlichkeit des zukünftigen Eintretens eines Zustandes ist dabei nur von dem gegenwärtigen Zustand abhängig. Diese Eigenschaft nennt man „Markov-Eigenschaft“. Das Markov-Modell beruht auf einem gedächtnislosen stochastischen Prozess. In einigen Erweiterungen/Verallgemeinerungen von Markov-Modellen ist die Wahrscheinlichkeit des zukünftigen Eintretens eines Zustandes gegebenenfalls auch von einer begrenzten Vorgeschichte der Systeme abhängig. /LAW 93/ /SGS 20/

Die Evaluation grundlegender Blockdiagramme oder FTA-Modelle kann mithilfe von boolescher Algebra durchgeführt werden. Komplexere Modelle werden jedoch im Allgemeinen in Markov-Modelle oder Erweiterungen/Verallgemeinerungen von Markov-Modellen übersetzt. Anwendung finden diese in vielfältigen Bereichen, wie beispielsweise in der Spracherkennung, Biologie, Medizin und im Investmentbanking. Für die Betrachtung von Systemen deren Komponenten repariert werden oder einen Fehlzustand aufweisen können werden im Allgemeinen direkt zyklische Markov-Modelle erstellt. /LAW 93/ /TÜV 20a/ /STO 06/ /SEI 07/

Ein großer Vorteil von Markov-Modellen ist ihre Flexibilität bei der Darstellung des dynamischen Systemverhaltens. So lassen sich die meisten verschiedenen Arten von Systemverhalten modellieren, die sich auch durch kombinatorische Modelle (wie z. B. die Anwendung von Fehlerbäumen) modellieren lassen. Ein weiterer Vorteil von Markov-Modellen ist die Möglichkeit vergleichsweise einfache Modelle zu gestalten. Dies geht jedoch mit Nachteilen einher: Wie alle Modelle beruhen sie auf vereinfachten Annahmen und bergen somit Einschränkungen. Je nach Anwendungsbereich kann zudem die Gedächtnislosigkeit des Markov-Modells einen Nachteil darstellen. Die Wahrscheinlichkeit des Wechsels von einem Zustand in einen anderen Zustand hängt in Markov-Modellen

nicht von dem Zeitpunkt ab, zu dem ein bestimmter Zustand eingenommen wurde. Dies bildet allerdings nicht immer die Realität ab. Es existieren Verfahren zur Integration derartiger Abhängigkeiten in Markov-Modellen, erweiterte/verallgemeinerte Markov-Modelle, jedoch geht dadurch die Einfachheit der Modelle verloren und sie werden oft unhandlich. In diesem Fall kann es sein, dass sie schwierig zu konstruieren und zu validieren sind. Ein weiterer Nachteil von Markov-Modellen ist, dass sie häufig strukturell den physischen oder logischen Organisationen des Systems nicht ähneln. /INS 09/ /BOY 98/

In folgenden Fällen wird laut /BOY 98/ von einem Markov-Modell abgeraten:

- Das System lässt sich auch mit einfacheren kombinatorischen Modellen zufriedenstellend abbilden.
- Das System erfordert eine große Anzahl an Zuständen oder ist sehr detailliert und komplex.
- Die Voraussage detaillierten Verhaltens ist erforderlich.

Es existieren verschiedene Unterarten von Markov-Modellen. Das einfachste ist die Markov-Kette (teilweise synonym auch: Markov-Prozess) erster Ordnung. Markov-Ketten lassen sich grafisch darstellen und mithilfe von Matrizen lösen. Umfangreichere Markov-Modelle lassen sich jedoch nur schwer mit analytischen Methoden lösen, so dass oftmals auf Simulationen zurückgegriffen wird.

Je nach Komplexität des Modells kann die Wahrscheinlichkeit des Übergangs zwischen zwei Zuständen in einem Markov-Modell entweder als konstant oder als zeitlich veränderlich angenommen werden. Für zeitdiskrete Modelle wird hierbei die Zeit in gleichgroße Abschnitte (Zyklen) unterteilt und den jeweiligen Zyklen Zeitpunkte zugeordnet. Diese liegen im Allgemeinen in der Intervallmitte. Zudem wird eine endliche Anzahl an Zuständen festgelegt, die dabei folgende Eigenschaften aufweisen /STO 03/:

- Sie schließen sich gegenseitig aus (disjunkt).
- Alle möglichen Zustände, in denen sich das System befinden kann, sind abgebildet (erschöpfend).

Von Zyklus zu Zyklus sind mit einer gewissen Übergangswahrscheinlichkeit Übergänge zwischen den Zyklen möglich. /STO 03/

Die Eignung der Markov-Modelle zur Zuverlässigkeitsanalyse softwarebasierter Leittechniksysteme auf Systemebene wurde im Rahmen von GRS Forschungsvorhaben erprobt und nachgewiesen /GRS 18/. Hierbei wurden redundant aufgebaute Leittechnikarchitekturen hinsichtlich ihres Ausfallverhaltens analysiert.

Für die Analyse von Software mittels Markov-Modellen sind Ansätze in der Literatur wie z. B in /KAU 15/, /BAI 05/ vorhanden. Diese Ansätze basieren wie im Abschnitt 2.4 erläutert, in der Regel auf Expertenschätzungen zu den Ausfallwahrscheinlichkeiten bzw. Ausfallraten der untersuchten Software, z. B. ausgehend von Annahmen zum Betrieb des Systems.

2.4.5 Petri-Netze

Mithilfe von Petri-Netzen können Systeme, in denen Konkurrenz und Ressourcenteilung existiert, formal beschrieben werden. Der Begriff System bezieht sich hierbei nicht allein auf einen Computer, sondern kann auch allgemein organisatorische, technische und rechnerintegrierte Systeme aller Art, in denen geregelte Flüsse von Gegenständen und Informationen stattfinden, bezeichnen.

Die Modellierung von Petri-Netzen kann qualitativ oder quantitativ erfolgen. Petri-Netze werden häufig für die Modellierung, Analyse und für Verifikationsprotokolle von Netzwerken eingesetzt. Zudem werden mit ihrer Hilfe oftmals Abläufe (z. B. Kommunikationsprotokolle) oder parallele Programmabläufe modelliert und verteilte Datenbanken geplant.

Mithilfe von Petri-Netzen lässt sich das Vorhandensein oder Fehlen von sicherheitsrelevanten Eigenschaften wie beispielsweise gefährliche Situationen oder Systemzustände, unerreichbare Zustände, Nebenläufigkeiten[4], Konflikte[5], Konfusionen[6] oder Verklemmungen[7] (engl.: deadlock) analysieren. /ECK 11/ /LAW 93/ /UNI 20/ /UNI 20a/ /UNI 20b/

Petri-Netze zeichnen sich durch ihre vergleichsweise „einfache“ Sprache aus. Somit lassen sich der jeweilige Ablauf und die Problemstellung gut veranschaulichen. Zudem ermöglichen sie eine einfache Kommunikation zwischen Beteiligten mit verschiedenen fachlichen Hintergründen. Durch die Schaltvorgänge sind Abläufe direkt ablesbar.

Darüber hinaus werden mögliche Umstrukturierungen durch die grafische Darstellungsform erleichtert und eine anschauliche Modellierung von Nebenläufigkeiten ermöglicht. Anhand von Simulationen lassen sich die Netze jederzeit testen. /UNI 20b/

Ein Nachteil von Petri-Netzen ist, dass sich aus dem Modell selbst keine Aussagen über mögliche Optimierungen herleiten lassen. Zudem ist Ihre Anwendbarkeit bei steigender Modellkomplexität begrenzt. Die Untersuchung aller möglichen Systemzustände ausgehend von allen möglichen Ausgangszuständen läuft auf eine sehr komplexe Erreichbarkeitsanalyse[8] hinaus. Hierzu ist je nach Umfang und Komplexität des modellierten Systems eine mehrtägige Rechendauer auf (Groß-)Rechenanlagen notwendig. /UNI 20b/

---

[4] Eine Nebenläufigkeit tritt in Petri-Netzen auf, wenn mehrere Ereignisse aktiviert sind und diese alle willkürlich schalten können (vorausgesetzt sie haben keine gemeinsamen Eingangsstellen). Somit finden gleichzeitig nebeneinander ablaufende Vorgänge statt. /UNI 20c/

[5] In Petri-Netzen bezeichnet „Konflikt“ ein nicht eindeutiges Systemverhalten. In diesem Fall stehen zwei aktivierte Ereignisse in Konkurrenz um eine Bedingung, wobei nur eines der Ereignisse stattfinden kann, da aufgrund seines Eintretens das andere Ereignis nicht mehr aktiviert ist. /UNI 20b/

[6] Eine „Konfusion“ bezeichnet im Zusammenhang mit Petri-Netzen eine Kombination aus Verzweigungs- und Wettbewerbskonflikt /UNI 20b/.
Ein Verzweigungskonflikt tritt in Petri-Netzen auf, wenn zwei Ereignisse eine gemeinsame Vorgängerstelle haben, deren Markierung zu einer Konfliktsituation führen kann und nur eine Transition schalten kann /UNI 20d/
Ein Wettbewerbskonflikt entsteht in einem Petri-Netz, wenn mindestens zwei Transitionen auf die gleiche Nachbedingung schalten wollen. /UNI 20b/

[7] In der Informatik bezeichnet man mit „Verklemmung“ unter anderem einen Systemzustand, in dem mehrere Prozesse auf die Freigabe von Betriebsmitteln warten, die durch andere Prozesse und den jeweiligen Prozess selbst exklusiv belegt sind. Ein anschauliches Beispiel ist eine Straßenkreuzung, an der die Vorfahrtsregel rechts-vor-links gilt und auf jeder Straße der Kreuzung ein Auto steht. Somit würde jedes Auto auf die Freigabe der von Ihm linken Straße durch das dort stehende Auto warten.

[8] Das systematische Abprüfen aller Schaltmöglichkeiten eines Stellen-Transitions-Systems (S/T-System) wird als Erreichbarkeitsanalyse bezeichnet. Die erreichbaren Markierungen werden dabei üblicherweise tabellarisch aufgelistet. Unter S/T-Systemen versteht man die prozessorientierten Interpretationen von Netzgraphen. Diese entstehen unter Hinzunahme von nicht unterscheidbaren Marken. /UNI 20f/

Ein Petri-Netz besteht aus in der Graphentheorie verwendeten „Knoten“ und „Kanten“. Es existieren verschiedene, unterschiedlich komplexe Petri-Netze. Im weiteren Verlauf werden sogenannten „Stellen-Transitions-Netze“ beschrieben. In diesen existieren zwei verschiedene Arten von Knoten, sogenannte „Stellen“ (auch: Plätze), die unterscheidbare, grundlegende Zustände (passive Elemente) mit einer bestimmten Verweildauer repräsentieren, und „Transitionen“, die die Änderungen elementarer Zustände auf Basis einer Schaltregel oder eines Ereignisses (Prozesse, aktive Elemente) repräsentieren /DIN 12/. Die Kanten repräsentieren die Ablaufbeziehungen zwischen Stellen und Transitionen.

Stellen können mit Token („schwarze Punkte“) belegt werden, dadurch wird das eigentliche Auftreten eines Systemzustandes abgebildet (auch „lokale Markierung“). Die Menge aller Stellenmarkierungen wird „Markierung des Netzes“ (en: net marking) oder „globale Markierung“ genannt /DIN 12/. Die Markierung von Stellen kann durch „Schalten“ oder „Feuern“ von Transitionen verändert werden. Dadurch entsteht die Dynamik der Netze, die durch den „Tokenfluss“ (en: token flow) dargestellt werden kann. Die erste Markierung, also diejenige, bevor irgendeine Transition geschaltet hat, ist die initiale Markierung, auch „Anfangsmarkierung“ (en: initial marking) genannt.

In Petri-Netzen werden Stellen als Kreise (oder Ovale), Transitionen als gerade Striche (oder Rechtecke) und Kanten als Pfeile dargestellt. Die Ausführung eines Petri-Netzes findet hierbei über die „Token“ statt.

Abb. 2.11 zeigt beispielhaft den Verlauf der Jahreszeiten als Petri-Netz. Die Zustände (Stellen oder Plätze) zeigen in den Ovalen die vier verschiedenen Jahreszeiten an. Die Transitionen (Übergänge von einer Jahreszeit in eine andere Jahreszeit) sind als Rechtecke dargestellt. Die Ablaufbeziehungen werden durch die Pfeile dargestellt.

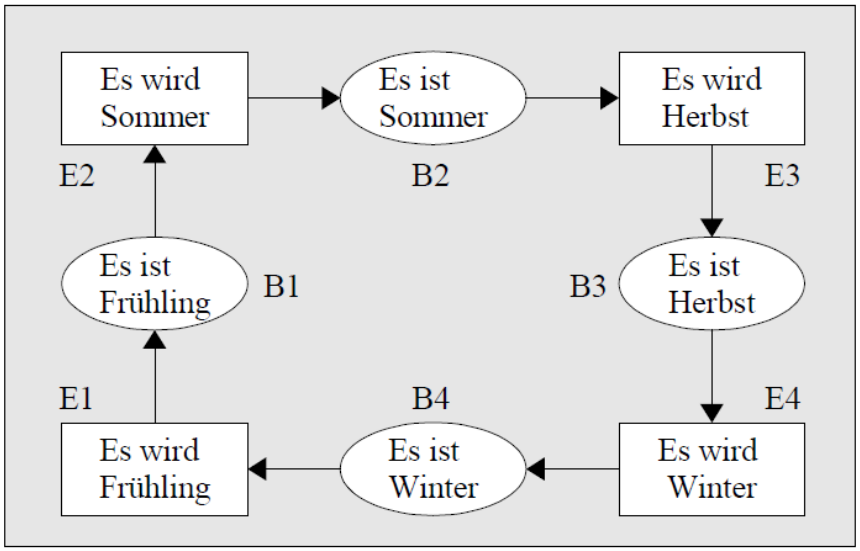

Abb. 2.11 Einfaches Beispiel eines Petri-Netzes zur Darstellung der Jahreszeiten nach /GRU 05/

Die initiale Markierung des Petri-Netzes (Ausgangszustand des Systems) erfolgt beispielsweise am Zustand B1 (Es ist Frühling). Durch „Schalten" der Transition E2 (Es wird Sommer) wechselt das Petri-Netz in den Zustand B2 (Es ist Sommer), welcher entsprechend markiert wird. Der Übergang in eine andere Jahreszeit erfolgt analog.

Das Hauptmerkmal der Petri-Netz-Modellierung ist die Darstellung der gleichzeitigen Ausführung von Aktivitäten eines Systems. Mit Hilfe einer derartigen Darstellung lassen sich zeitliche Abläufe in Systemen/Komponenten, in denen eine Ressourceneinteilung unabdingbar ist, wie z. B die Datenkommunikationsnetzwerke, analysieren.

Petri-Netze werden insbesondere zur Bewertung der Wirksamkeit von fehlertoleranten Maßnahmen in der Software und zur Zuverlässigkeitsanalyse von softwarebasierten Leittechniksystemen auf Systemebene eingesetzt /LEU 91/, /LEV 85/.

Das im Rahmen dieses Vorhabens entwickelte Konzept dient der Analyse der Software auf der Funktionsebene. Daher wird die Anwendung der Petri-Netz-Modellierung zur Zuverlässigkeitsanalyse im Rahmen des Vorhabens nicht weiter betrachtet.

2.4.6 Software Fault Injection (SFI)

Die Software Fault Injection (dt. Software Fehlerinjektion) ist eine bewährte experimentelle Methode, um die Zuverlässigkeit eines softwarebasierten Systems zu testen. Eine hohe Zuverlässigkeit eines softwarebasierten Systems kann u. a. dann gewährleistet

werden, wenn das System mit einer hohen Fehlertoleranz entwickelt wurde, so dass ein System auch in Gegenwart von Fehlern seine Funktion erfüllt. Ziele der Softwarefehlerinjektion sind, das Systemverhalten unter dem Einfluss von vorsätzlich eingeführten Fehlern zu beobachten und dadurch Mängel bei der Fehlertoleranz und -korrektur aufzudecken. Die Softwarefehlerinjektion wird angewandt zur /UIL 97/:

- Identifikation von Schwachstellen der Systemzuverlässigkeit,
- Untersuchung des Systemverhaltens unter dem Einfluss unterschiedlicher Fehler,
- Bewertung der getroffenen Fehlererkennungsmaßnahmen und -korrekturmaßnahmen im Hinblick auf die Abdeckung möglicher Fehler,
- Bewertung der Wirksamkeit von Fehlertoleranzmechanismen und Leistungsverlusten.

Je nach Entwicklungszustand des Systems und den Testzielen kann die Softwarefehlerinjektion an einer Simulation des Systems, an einem Prototyp oder auch an einem in Betrieb befindlichen System durchgeführt werden. Anstatt Fehler direkt ins System zu injizieren, kann die Auslastung eines Systems im Betrieb beobachtet bzw. erfasst werden, um Daten über mögliche Fehlerzustände des Systems z. B. aufgrund fehlerhafter Eingangsdaten zu gewinnen. Durch die Analyse dieser Daten kann ein besseres Verständnis für tatsächliche Fehler und Versagensmechanismen erlangt werden. Bei dieser Methode müssen jedoch zuvor über einen längeren Zeitraum Daten gesammelt werden, die zu Fehlerzuständen geführt haben /UIL 97/.

Die Fehlerinjektionsumgebung nach /UIL 97/ ist in Abb. 2.12 dargestellt. Sie besteht typischerweise aus dem zu testenden Zielsystem und einem Fehlerinjektor, mit dem die Fehler in das System injiziert werden.

Diese Fehler können zuvor in der sogenannten Fehlerbibliothek definiert werden. Damit das Zielsystem möglichst realistisch nachgebildet werden kann, gibt es zudem einen Nutzlastgenerator, der die Anwendungen und Funktionen des Zielsystems aufruft. Der Nutzlastgenerator erhält seine Anweisungen, welche Funktionen auszuführen sind, aus der Nutzlast-Bibliothek. Der Monitor überwacht die Ausführung der Befehle und löst die Aufzeichnung der Daten aus. Die Daten werden im Datensammler online gesammelt und können nach dem durchgeführten Test mit dem Datenauswerter analysiert werden. Das Fehlerinjektionssystem wird von einem Controller gesteuert, der bis auf den Datensammler alle Komponenten kontrolliert /UIL 97/.

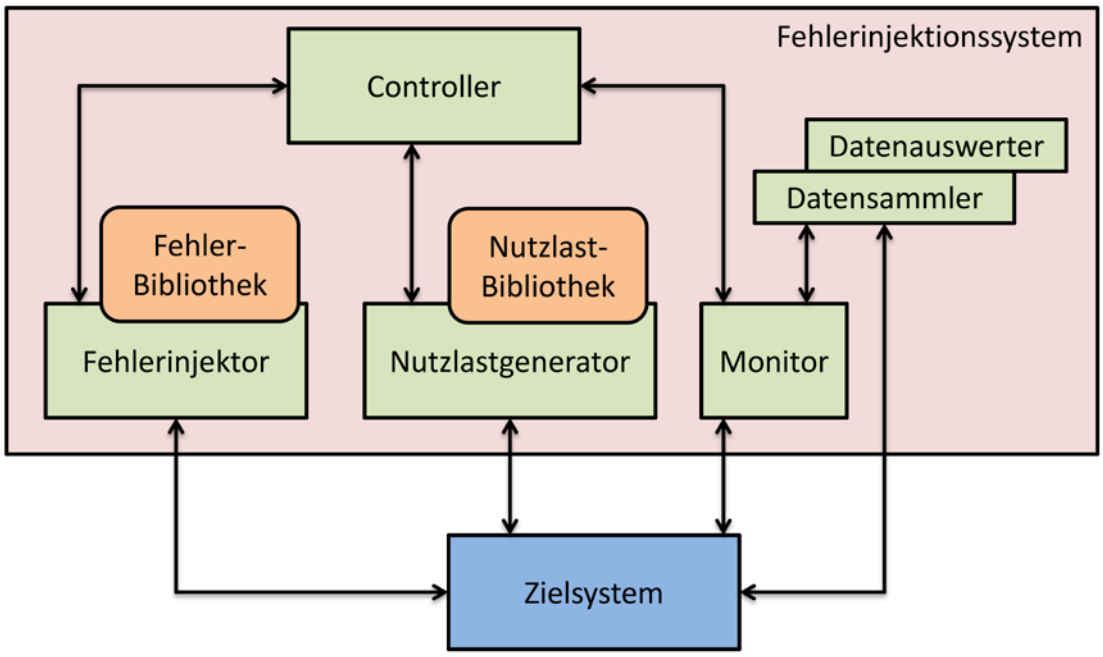

Abb. 2.12 Bestandteile einer Fehlerinjektionsumgebung entnommen aus /TUD 12/.

Da auch Hardwarefehler zum Versagen der Software führen können, kann auch eine auf Hardware basierte Fehlerinjektion durchgeführt werden. Dabei wird unterschieden zwischen der Hardwarefehlerinjektion *mit* und *ohne Kontakt*. Bei der Hardwarefehlerinjektion *mit Kontakt* werden die Leitungen eines Chips mit Elektroden in Berührung gebracht und durch Veränderung der anliegenden Spannungspegel oder des fließenden Stroms Fehler injiziert. Damit lassen sich sogenannte Stuck-at Fehler (Fehler am Eingang/Ausgang eines Logikgatters) in die Hardware injizieren. Alternativ können durch Kurzschließen mehrerer Leitungen auch die Auswirkungen eines Kurzschlusses im System beobachtet werden. Um das System nicht zu beschädigen kann über einen Adapter zwischen dem Chip und der Platine des Zielsystems jede Leitung auf eine Buchse geführt sein. Über diese Buchse können dann mittels Analogsignalen komplexe Logikfehler in das System injiziert werden /UIL 97/.

Bei der Hardwarefehlerinjektion *ohne Kontakt* wird zur Injektion von Fehlern gezielt Schwerionenstrahlung auf eine Leitung der Hardware ausgerichtet oder das System in ein elektromagnetisches Feld eingebracht. Diese Hardwaretests imitieren physikalische Phänomene, wie z. B. Bit-Flip, Störstrom etc. /UIL 97/.

In der folgenden Tab. 2.1 sind Beispiele für Hardware- und Softwarefehler gegeben, die mit der Fehlerinjektionsmethode untersucht werden können.

Tab. 2.2 Hardware- und Softwarefehler, die mit der Fehlerinjektionsmethode untersucht werden können nach /UIL 97/.

Hardwarefehlerinjektion	Softwarefehlerinjektion
Bit-Flip (dt. Bit-Umwandlung): Änderung eines einzelnen oder mehrerer Bits innerhalb eines Datensatzes.	**Speicherdatenfehler** (RAM): Verfälschung von Daten in Registern, Arbeitsspeichern und Festplattenspeichern
Stuck-at (dt. Haftfehler): Fehler bei dem, eine auf Basis integrierter Schaltkreise (ICs) aufgebaute elektronische Komponente, wie z. B. Logikgatter, Mikroprozessor, an ihrem Eingang bzw. Ausgang auf einem bestimmten Wert feststeckt.	**Kommunikationsdatenfehler** (Bus, LAN etc.): Verfälschung von Übertragungsdaten für die Datenkommunikation im Netzwerk.
Bridging-Faults (dt. Kurzschlüsse): Fehler, bei dem der Ein- und Ausgang eines Logikgatters miteinander verbunden ist.	**Programmfehler**: Einbringen von Fehlern in den Code des auszuführenden Programms
Spurious current (dt. Störstrom): Störströme z. B. ausgelöst durch kosmische Strahlung oder Zerfall radioaktiver Isotope (Single Event Upsets[9] als Folge derartiger Störstrahlung).	
Power surge (dt. Überspannung): beispielsweise ausgelöst durch Stromausfälle.	

In den letzten Jahren wurden verschiedene Programme entwickelt, um die Auswirkungen der oben beschriebenen Hardware-Fehlerinjektionen zu untersuchen. Ein Vergleich der unterschiedlichen Methoden zeigt, dass Schwerionenstrahlung meist Adressbusfehler auslöst, während Störströme meistens den Programmkontrollfluss beeinträchtigen /UIL 97/.

Die Software-Fehlerinjektion hat den Vorteil, dass sie auch auf Anwendersoftware und laufende Betriebssysteme angewendet werden kann, was bei der Hardware Fehlerinjektion sehr schwierig ist. Mit der Software-Fehlerinjektion lassen sich sowohl die Auswirkungen von Fehlern im Quelltext als auch von Fehlerzuständen (z. B. überlaufender Speicher) untersuchen. Der Nachteil der Software-Fehlerinjektion ist, dass Fehler lediglich an Stellen, die für die Software zugänglich sind, injiziert werden

[9] Single Event Upset (SEU): Modifikation des elektronischen Zustandes in einer Speicherzelle als Folge Störstrahlung (kosmische oder radioaktive Strahlung). SEU manifestieren sich beispielsweise als Bit-Flips.

können. So können beispielsweise Speicher, Register und Kommunikationskanäle durch Software-Fehlerinjektion untersucht werden, Hardware-Komponenten wie der Cache oder das elektronische Rechenwerk im Prozessor (Arithmetisch-logische Einheit ALU) lassen sich jedoch nicht mit dieser Methode untersuchen /POT 12/.

Bei der Wahl der Fehlerinjektions-Methode sollte zunächst entschieden werden, welche Arten von Fehlerursachen und Fehlerzuständen in den Fehlerinjektionstests simuliert werden sollen und welches Fehlermodell demzufolge als Grundlage verwendet werden sollte.

Um die Tests möglichst realistisch durchzuführen, sollten die Fehler an den betroffenen Stellen im Programm eingefügt werden und der Zeitpunkt der Injektion entsprechend gewählt werden. Bei der Software Fehlerinjektion wird der Fehler in den Quellcode des zu testenden Programms *zum Kompilationszeitpunkt (compile-time)* injiziert und danach der manipulierte Quellcode kompiliert. Beim Abrufen des manipulierten Quellcodetextes wird dieser Fehler dann ausgeführt. Mit dieser Methode lassen sich transiente Hardwarefehler und permanente Hard- und Softwarefehler simulieren. /UIL 97/

Beispiele für die Anwendungsmöglichkeiten dieser Methode sind das Testen der Reaktionen des Systems auf Ausnahmen (Ausnahmebehandlung) oder der Umgang mit Berechnungs- und Wertefehlern.

Der Vorteil dieses Verfahrens liegt darin, dass es einfach durchzuführen ist und Fehler nachgebildet werden, die in ihrer Lokalität auf das Testprogramm beschränkt sind. Der Nachteil dieses Verfahrens liegt jedoch darin, dass keine Fehler injiziert werden können während sich das Programm im Prozess befindet. /UIL 97/

Bei der sogenannten Software-Fehlerinjektion *zur Laufzeit (run-time)*, können Fehler auch während der Ausführungszeit des Programms injiziert werden. Dabei wird eine Bedingung festgelegt, die die Injektion des Fehlers auslöst, während sich das Programm im Prozess befindet. Eine solche Bedingung kann eine festgelegte Zeit sein *(time-out),* die eine vorübergehende Unterbrechung des laufenden Programms *(interrupt)* verursacht, um den Fehler in das System zu injizieren. Es kann aber auch eine Meldung des Programms, eine sogenannte Ausnahmesituation *(exception/trap)* genutzt werden,

um Fehler in das System zu injizieren. So kann beispielsweise eine Ausnahmesituation vor einer bestimmten Anweisung in das Programm eingefügt werden.

Wird diese Ausnahme vom Programm ausgeführt, kommt es zu einer Unterbrechung des Programms durch den Interrupt-Handler, so dass der Fehler genau vor dieser Anweisung injiziert werden kann. /UIL 97/

Im Rahmen dieses Vorhabens wurde zur Validierung des entwickelten SFMEA Konzeptes die Software Fault Injection-Methode angewandt. Dies wird in Kapitel 4 näher beschrieben.

2.4.7 Software HAZOP

Unter einer HAZOP (Hazard and Operability Study) versteht man eine systematische Überprüfungsmethode von Systemen oder Systembestandteilen zur Identifizierung von Gefährdungen, Fehldesigns und Fehlfunktionen sowie der Analyse ihrer Auswirkungen auf die Umgebung. Als Gefährdungszustand (engl. „hazard“) bezeichnet man in diesem Zusammenhang gemäß /KOR 01/ einen Zustand oder eine Reihe von Systembedingungen, die im Zusammenspiel mit anderen System- oder Umgebungsbedingungen zu einem Unfall[10] führen können. In Bezug auf Software entspricht dies einem Softwarezustand, der eine Voraussetzung für einen Unfall darstellt /KOR 01/. Eine HAZOP wird anhand eines methodischen, nach festen Regeln gelenkten Brainstormings in einer Gruppe von Personen mit entsprechenden Fähigkeiten durchgeführt. Diese Überprüfungstechnik wurde von der chemischen Industrie entwickelt und hat sich inzwischen auch in anderen Anwendungsbereichen bewährt. Im deutschsprachigen Raum ist sie auch als PAAG-Verfahren[11] bekannt.

Ziel des PAAG/HAZOP-Verfahrens ist es, mögliche Abweichungen vom bestimmungsgemäßen Betrieb eines Systems („Sollzustand“) aufzudecken, die jeweiligen Ursachen und Auswirkungen zu benennen (und gegebenenfalls zu bewerten) sowie geeignete Gegenmaßnahmen zur Vermeidung von deren Auswirkungen auf die Systemsicherheit

---

[10] Unfälle sind in /KOR 01/ als anomale Betriebszustände/Ereignisse definiert, deren Eintreten nicht erwartet wird, die aber postuliert werden. Sie entsprechen Störfälle gemäß der Definition in /BMU 15/.

[11] PAAG-Verfahren: **P**rognose, **A**uffinden der Ursachen, **A**bschätzen der Auswirkungen, **G**egenmaßnahmen.

festzulegen. Zentraler Baustein des Verfahrens ist die Anwendung so genannter Leitworte/Leitsätze mittels derer die Abweichungen vom „Sollzustand“ des Systems anhand von Leitfragen formuliert werden.

Abb. 2.13 zeigt eine Übersicht der Leitworte eines PAAG-/HAZOP-Verfahrens. In Tab. 2.2 sind Interpretationen dieser Leitworte von Abb. 2.13 beispielhaft nach /IVS 20/ für verfahrenstechnische Anlagen aufgelistet.

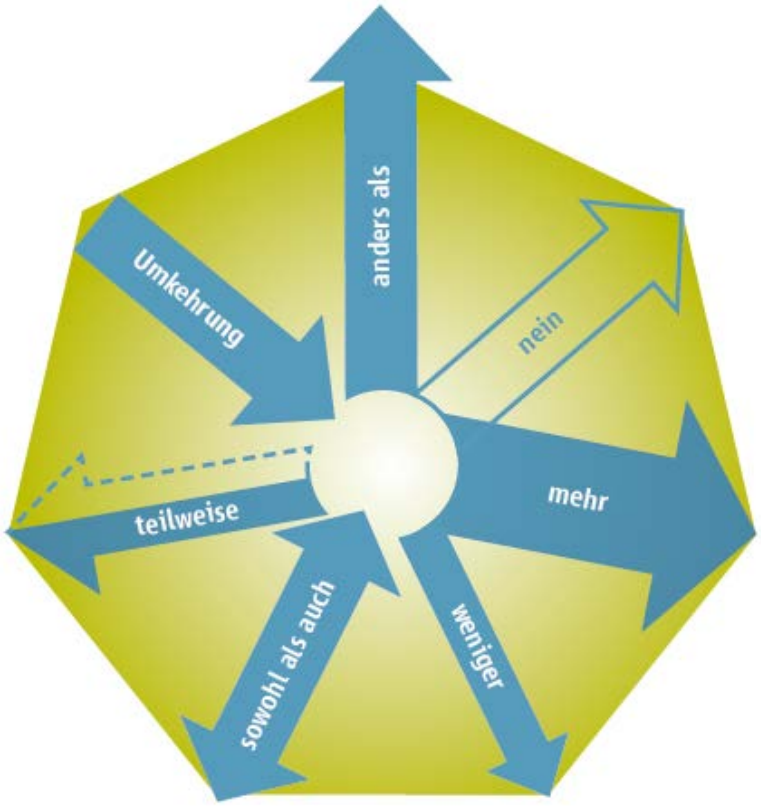

Abb. 2.13 Darstellung der Leitworte eines PAAG-/HAZOP-Verfahrens ausgehend vom „Sollzustand“ (Mitte) des Systems /IVS 20/.

Tab. 2.3 Interpretation PAAG-/HAZOP-Verfahren für verfahrenstechnische Anlagen

Leitworte	Interpretation
Nein/Nicht	Die Sollfunktion oder einzelne Aspekte davon sind nicht gegeben, z. B. die Sollfunktion wird nicht erfüllt oder Komponente ist nicht vorhanden.
Mehr	Quantitative Größen der Sollfunktion nehmen zu, z. B.: • Menge bzw. Mengenstrom, • Temperatur, Druck, • Zeitraum bzw. Geschwindigkeit, • weiterhin z. B. Viskosität, Konzentration, pH-Wert.
Weniger	Quantitative Größen der Sollfunktion nehmen ab, vgl. „Mehr
Sowohl als auch	Die Sollfunktion wird erreicht, zusätzlich zum bestimmungsgemäßen Prozess geschieht noch etwas anderes, z. B.: • mehr Komponenten im System (Verunreinigungen, Reinigungsmittel, Korrosionsprodukte, Heiz-/Kühlmedien, Luft, weitere Stoffe), • mehr Wege (offene Armaturen, Leckage innerhalb des Systems oder nach außen), • mehr Vorgänge (Korrosion, Abrasion, Kavitation, statische Aufladung).
Teilweise	Die Sollfunktion wird nur unvollständig erfüllt bzw. einzelne Teile der Sollfunktion sind nicht vollständig vorhanden, z. B.: • mehrstufige chemische Reaktion verläuft nur unvollständig (durch Anwesenheit eines Inhibitors/ Katalysatorgiftes, durch fehlenden Katalysator), • in einem Stoffgemisch fehlt eine Komponente teilweise oder ganz (durch physikalische Entmischung, Ausgasen, Phasentrennung, Zersetzung), • das System wird entgegen der Vorgabe nicht durchmischt (infolge Ausfalls oder Fehlfunktion des Rührers).
Umkehrung	Etwas geschieht in umgekehrter Richtung oder in umgekehrter Reihenfolge, z. B. umgekehrte Fließrichtung, entgegengesetzte Bewegung, umgekehrte Schrittfolge.
Anders als	Teile der Sollfunktion werden durch etwas Anderes ersetzt, z. B.: • anderer Stoff (Reaktanden, Katalysatoren, Lösemittel), • anderer Zustand (Aggregatzustand, Korngrößenverteilung, Reinheit), • andere chemische Vorgänge (ungewollte Reaktionen, Zersetzung, Polymerisation), • andere physikalische Vorgänge (ungewollte Fällungsreaktion, Phasenwechsel), • anderer Zeitpunkt (zu früh/zu spät), • anderer Ort (falscher Behälter), andere Betriebsweise (An-/Abfahren, Probenahme).

Zu den Stärken des PAAG-HAZOP-Verfahrens zählen u. a. die systematische und detaillierte Vorgehensweise sowie das breite Anwendungsfeld. Das PAAG-HAZOP-Verfahren kann zur Analyse verschiedenartiger Systeme oder Prozesse angewendet werden.

Aufgrund der vergleichsweisen einfachen Handhabung des Verfahrens kann mit einer HAZOP-Analyse sehr schnell ein Überblick über mögliche auslösende Fehler/Ereignisse für Gefährdungszustände eines Systems gewonnen werden.

Nachteilig für PAAG/HAZOP-Analysen ist u. a., dass sie nur auf einzelne Fehlerzustände abzielen. Kombinationen von Fehlern/Ereignissen, welche zu Gefährdungszuständen des untersuchten Systems führen können, werden nicht betrachtet. Zudem können mit PAAG/HAZOP-Analysen nur Gefährdungen aufdeckt werden, die in Verbindung mit den definierten Leitwörtern stehen; von den Leitwörtern unabhängige Gefährdungen können somit übersehen werden. Hinzu kommt, dass der Erfolg des Verfahrens sehr stark vom Fachwissen der Teammitglieder abhängig ist, da das PAAG/HAZOP-Verfahren als gelenktes Brainstorming in einem Expertenteam durchgeführt wird.
Eine detaillierte Analyse mit dem HAZOP-Verfahren benötigt einen sehr großen Dokumentationsaufwand oder umfangreiche Spezifikationen zu System, Prozess oder Verfahren und kann sehr zeitaufwändig sein.

Für die Anwendung des PAAG/HAZOP-Verfahrens zur Zuverlässigkeitsanalyse von Software werden Funktionsmerkale zur Charakterisierung des „Sollzustandes“ der zu untersuchenden Software eingeführt. In /NUR 97/ sind entsprechende Funktionsmerkmale der Software wie z. B. Genauigkeit, Funktionalität, Zuverlässigkeit, Robustheit, Sicherheit und Integrität vorgeschlagen. In /LAW 95/ findet sich ein Konzept zur Generierung von Leitsätzen/Leitworten zur Beschreibung der Abweichungen der Software von den zu betrachtenden Funktionsmerkmalen. Die Software-HAZOP wird durchgeführt, um Gefährdungszustände der Software zu identifizieren, die die Systemsicherheit beeinträchtigen können, wenn eine bestimmte Abweichung der Software vom zu betrachtenden Funktionsmerkmal vorliegt. Hierfür werden Abweichungschecklisten (in Form von Leitfragen) erstellt, mittels denen die Abweichungen der Software von den festgelegten Funktionsmerkmalen systematisch überprüft werden können.

In der Literatur finden sich verschiedene Ansätze zur Durchführung einer Software-HAZOP. Diese Ansätze bauen allesamt auf der konventionellen HAZOP auf und beinhalten daher die wesentlichen Schritte zur Durchführung einer HAZOP. Die Ansätze unterscheiden sich jedoch abhängig vom Anwendungsgebiet in der Umsetzung. Hierzu sind u. a. das betrachtete System, die betrachtete Abstraktionsebene (System- oder Funktionsebene), die Modellierung/Darstellung der zu untersuchenden Software und die Methode zur Generierung von Leitworten zu nennen.

Eine Übersicht über diese verschiedenen Ansätze zur Durchführung einer Software-HAZOP ist in /KOR 01/ aufgeführt. Zur Veranschaulichung des Grades der Unterschiedlichkeit dieser verschiedenen Methoden insbesondere im Bezug auf die Modellierung der Software werden exemplarisch auf Basis der Ausfürhungen in /KOR 01/ /PUM 99/ und /CLA 16/ die folgenden Methoden vorgestellt:

- Mithilfe einer Computer Hazard and Operability Study (kurz: CHAZOP-Studie, CHAZOPS) kann eine Bewertung eines Computersystems bzw. eines rechnerbasierten Steuerungssystems in Bezug auf potentielle Ausfallursachen oder potentielle Probleme der Bedienbarkeit durchgeführt werden. Hierbei wird die Software des Computersystems bzw. des Steuerungsystems als Black-Box betrachtet. Softwareausfälle werden dementsprechend im Rahmen der CHAZOPS hinsichtlich ihrer Auswirkungen auf Systemebene behandelt. Die Art der Implementation einer CHAZOPS unterscheidet sich in den verschiedenen Anwendungsgebieten erheblich.

- Eine weitere Möglichkeit stellt die Anwendung eines dreistufigen Ansatzes dar, in dem eine konventionelle HAZOP für den Anlagenprozess, eine HAZOP von programmierbaren elektronischen Systemen (PES HAZOP) für die Prozesssteuerungssysteme der Anlage und eine Human Factors HAZOP durchgeführt werden. In diesem Fall weist die PES HAZOP starke Ähnlichkeiten zur klassischen FMEA auf. Mit diesem dreistufigen Ansatz werden die wesentlichen Bereiche der Prozessautomatisierung (Verfahrenstechnik, Leittechnik, Mensch-Maschine-Schnittstelle) in der Überprüfung von automatisierten Anlagenprozessen mittels HAZOP betrachtet.

- Bei der Software Hazard Analysis and Resolution in Design (kurz SHARD) Methode wird die HAZOP während der Designphase einer Software eingesetzt, um die Wahrscheinlichkeit von Fehlermodi der Softwarekomponenten zu charakterisieren. Ein wesentliches Merkmal von SHARD ist die Anwendung von Fehlerklassen für die Strukturierung der Leitwörter zur Beschreibung der Abweichungen der Software von ihrem Sollzustand. In /PUM 99/ sind Anwendungsbeispiele sowie mögliche Leitworte aufgeführt.

- Bei dem sogenannten Data Flow Diagramm (DFD)-Ansatz wird die zu untersuchende Software mittels Datenflussdiagramme modelliert. Die Software-HAZOP wird anschließend auf Basis dieses Modells durchgeführt, wobei entsprechende Leitworte/Leitsätze entwickelt werden.

Im nachfolgenden Abschnitt wird die Anwendung der Software HAZOP im Rahmen einer Softwaresicherheitsanalyse eines softwarebasierten Leittechniksystems an einem Beispiel näher erläutert.

2.4.9 Softwaresicherheitsanalyse am Beispiel eines vollständig digitalen Reaktorschutzsystems aus Korea

Im Folgenden wird die Durchführung einer Softwaresicherheitsanalyse (engl.: software safety analysis, SSA) am Anwendungsbeispiel des im Rahmen des KNICS (Korean Nuclear Instrumentation & Control System) -Projektes entwickelten volldigitalisierten Reaktorschutzsystems IDiPS (auch: „KNICS RPS[12]“) betrachtet. Das IDiPS soll in neu konstruierten koreanischen Kernkraftwerken, sowie im Rahmen von Upgrades in bereits existierenden Kernkraftwerken mit bislang analogen Reaktorschutzsystemen, verwendet werden (Stand 2008). Sämtliche Informationen dieses Unterkapitels sind /KOR 01/, /PAR 07/ und /PAR 08/ entnommen.

Jeder einzelne Kanal des vierkanalig aufgebauten IDiPS besteht aus zwei Gruppen von Prozessoren mit jeweils zwei Prozessoren (sogenannte BP- und CP-Prozessoren), einer als ATIP bezeichneten automatisierten rechnerbasierten Testumgebung und einem als COM bezeichneten Schrankmodul.

Die BP-Prozessoren generieren Anregesignale durch Vergleich der Prozessvariablen mit ihren zugehörigen Sollwerten/Grenzwerten. Die von den BP-Prozessoren generierten Anregesignale werden in den CP-Prozessoren entsprechend der Anwendungssoftware (Leittechnikfunktionen) zunächst verarbeitet. Die hieraus resultierenden Ausgangssignale werden durch eine zwei-von-vier-Auswahl ausgewertet und gegebenenfalls Auslösesignale zur Steuerung verfahrenstechnischer Komponenten erzeugt.

In der ATIP werden Testsignale für manuelle und automatische Tests des IDiPS erzeugt. Darüber hinaus führt das ATIP Integritätstests zur Überprüfung des Betriebsstatus der BP- und CP-Prozessoren. Im Schrankmodul COM werden IDiPS-Statusinformationen angezeigt. Zudem können Auslösesignale im Schrankmodul zurückgesetzt werden.

Im Rahmen des KNICS-Projektes wurde auch POSAFE-Q, eine speicherprogrammierbare Steuerung (engl.: programmable logic controller, PLC) für sicherheitsrelevante Anwendungen mit proprietärer[13] Betriebssoftware, entwickelt. Das IDiPS wurde basierend auf der POSAFE-Q-Plattform konfiguriert und die IDiPS-

[12] RPS: engl. Abkürzung für das Reaktorschutzsystem (reactor protection system)

[13] Als proprietär bezeichnet man Software, Betriebssysteme o.Ä., die sich nur auf herstellereigenen Geräten nutzen lassen.

Anwendungssoftware mithilfe von Funktionsblockdiagrammen (engl.: function block diagram, FBD) entwickelt.

Während der Verifikations und Validierungs (V&V)[14] -Aktivitäten für das IDiPS wurde eine SSA der IDiPS-Anwendungssoftware durchgeführt. Das Ziel der SSA ist es, softwarebedingte Gefährdungszustände gemäß /KOR 01/ (siehe Abschnitt 2.4.7) zu identifizieren. Die durchgeführte SSA der IDiPS-Anwendungssoftware beinhaltete eine SFTA und eine Software HAZOP. Abb. 2.14 zeigt eine Übersicht der V&V-Aktivitäten für das IDiPS.

[14] Verifizierung und Validierung sind zwei unabhängige Verfahren. Sie werden gemeinsam genutzt, um zu überprüfen ob eine Dienstleistung, ein Produkt oder ein System den erforderlichen Anforderungen und Spezifikationengenügt genügt (Verifizierung) sowie seinen Zweck erfüllt (Validierung).

Die aufeinanderfolgenden Entwicklungsschritte des IDiPS von der Planung der Software („Software Planning“) bis zur Systemintegration[15] („System Integration“) sind im oberen Teil dieser Abbildung von links nach rechts dargestellt. Darunter befinden sich die jeweils zugehörigen Prozesse der V&V. Für eine Erläuterung der Teilschritte des V&V Prozesses sei auf /KOR 01/ und /PAR 08/ verwiesen.

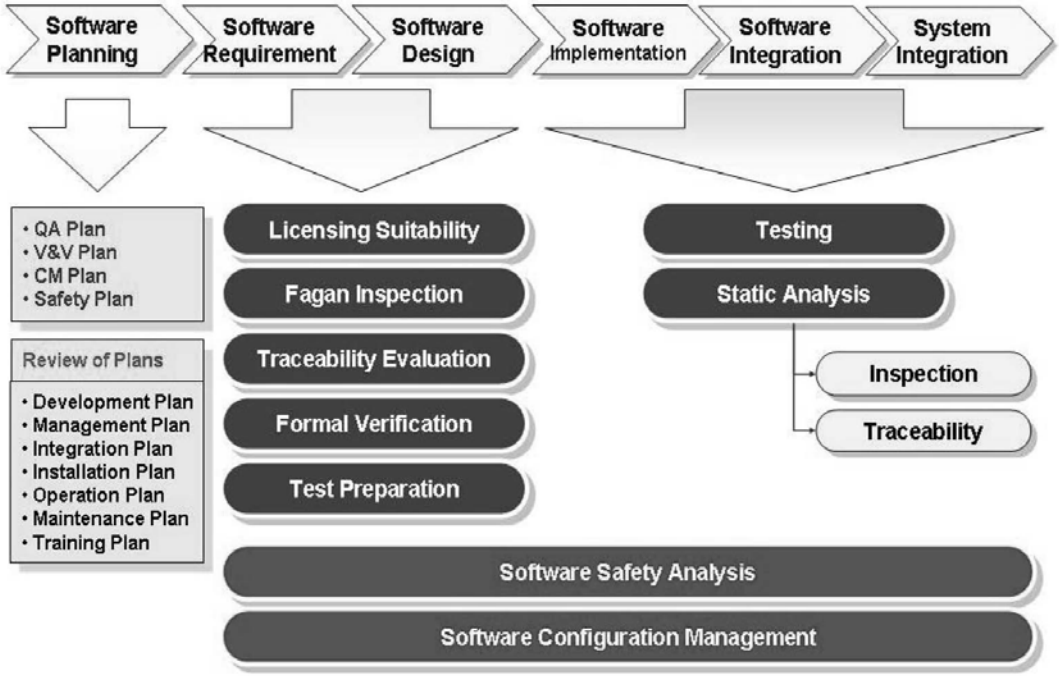

Abb. 2.14 Schematische Übersicht über die V&V Aktivitäten für das IDiPS /PAR 08/.

Während des V&V-Prozesses wurde die SSA in jeder Phase des Software-Lebenszyklus[16] durchgeführt. Zunächst wird während der Softwareplanungsphase („Software Planning“) ein Software-Sicherheitsplan („Software Safety Plan“) erstellt, der die SSA als Teil der Systemsicherheitsanalyse beinhaltet. Während der Anforderungsphase („Software Requirement“) wird eine Software-HAZOP im Rahmen der Gefährdungsanalyse als Teil der SSA durchgeführt. In der darauffolgenden Designphase („Software Design“) wurde eine weitere Software-HAZOP ausgeführt. Aufbauend auf deren Ergebnissen wurde in der Designphase („Software Design“)

[15] Prozess, bei dem verschiedene Computersysteme physisch oder funktionell zu einem Gesamtsystem verknüpft werden.

[16] Unter Software-Lebenszyklus versteht ‚man den Entwicklungsprozess einer Software, der die Phasen „Systemspezifikation“, „Systementwicklung“, „Systemintegration“, „Systemvalidierung“, „Systemerrichtung“ und „Systemmodifikation“ umfasst. /GRS 17/

zusätzlich die SFTA für die Analyse der FBD-Module durchgeführt, die im Rahmen der Software HAZOP als kritisch identifiziert wurden. Diese Vorgehensweise wurde gewählt, da im Rahmen einer SFTA erstellte Fehlerbäume für FBD-Module im Allgemeinen sehr lang und komplex sind. Mit der Software-HAZOP kann aufgrund der vergleichweise einfachen Handhabung des HAZOP-Verfahrens zunächst sehr schnell ein Überblick über potenzielle Gefährdungszustände der FBD-Module gewonnen werden. Die Anwendung dieser beiden Techniken (Software HAZOP und SFTA) führt zudem zu einer sich zum Teil überschneidenden Mehrarbeit im Rahmen der SSA. Da verschiedene Sicherheitsanalysemethoden über unterschiedliche Vor- und Nachteile verfügen und somit komplementär zueinander sind, sind diese Überschneidungen ein wichtiger Bestandteil der Analysen. Nachteile eines Verfahrens können auf diese Weise durch den Einsatz anderer Methoden zum Teil ausgeglichen werden, wodurch eine möglichst abdeckende Sicherheitsanalyse erzielt werden kann. Auch während der Implementierungsphase[17] („Software Implementation“) werden Software HAZOP und SFTA genutzt. Wie Abb. 2.14 zu entnehmen ist, wurden im Rahmen der V&V-Aktivitäten zusätzlich zu den Teilschritten der SSA auch verschiedene Dokumentenevaluationen durchgeführt, um mögliche Schwachstellen der Software zu entdecken. Hierzu gehören beispielsweise eine sogenannte Fagan-Inspektion, bei der Dokumente wie beispielsweise der Quellcode und formale Spezifikationen überprüft werden.

Abb. 2.15 zeigt einen Einblick in die Struktur der Software-Sicherheitsanalyse des KNICS-Projektes. Die einzelnen Schritte dieser Analyse sind beginnend mit der vorbereitenden Gefährdungsanalyse des Systems („Preliminary System Hazard Analysis“) bis zu den abschließenden Empfehlungen und den Zuverlässigkeits- und Sicherheitsanalyseberichten in dem Schaubild von oben nach unten aufgeführt. Es ist zu erkennen, dass die Software HAZOP während der Anforderungsphase („Software Requirements“) und die Software HAZOP und SFTA während der Design- („Software Design“) und Implementierungsphase („Software Code“) zur Anwendung kommen. Genauere Erläuterungen können /PAR 07/ entnommen werden.

[17] Als „Implementierung“ bezeichnet man die Umsetzung der geplanten Software in Code.

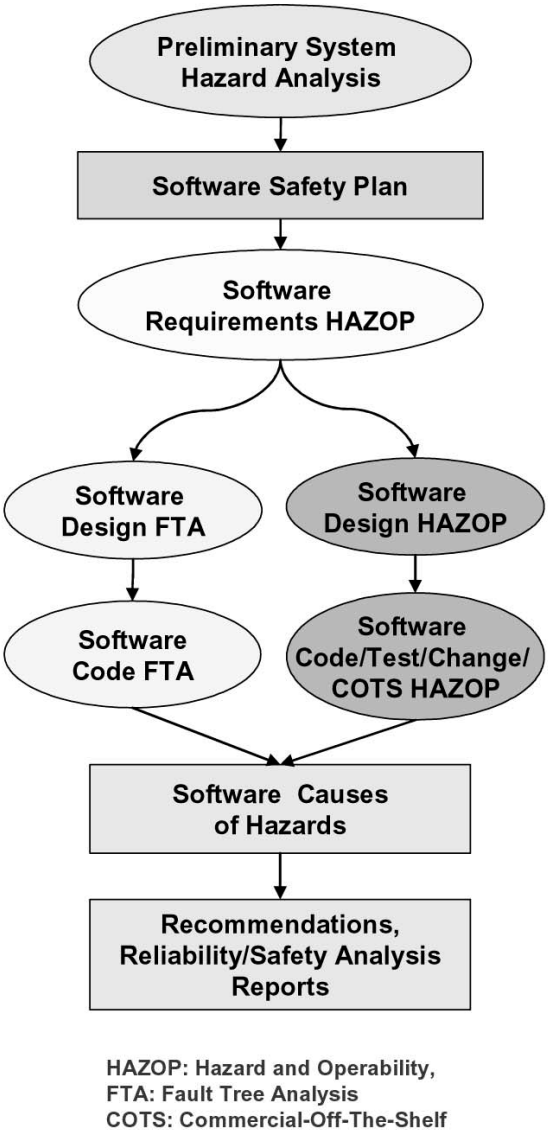

Abb. 2.15 Software-Sicherheitslebenszyklus des KNICS Projektes /PAR 08/

Die für die Durchführung der SSA relevanten softwarebedingten Gefährdungszustände wurden basierend auf den Ergebnissen einer durchgeführten FMEA des IDiPS ermittelt. Hierbei wurden einige Fehlermodi der in den BP-Prozessoren implementierten Anwendungssoftware des IDiPS zur Generierung von Anregesignalen aus den Prozessvariablen identifiziert, die die System- und Anlagensicherheit beeinträchtigen können.

Zur Kennzeichnung der Schwere der Auswirkungen der softwarebedingten systembezogenen Gefährdungen auf die System- und Anlagensicherheit wurden vier Kritikalitätslevel eingeführt. Tab. 2.4 zeigt beispielsweise eine Übersicht über systembezogene Gefährdungen mit Softwareaspekten und ihre zugehörigen

Kritikalitätslevel für die in den BP-Prozessoren laufende Anwendungssoftware des IDiPS. Der Kritikalitätslevel 4 stellt hierbei die höchste Stufe dar. Ein Gefährdungszustand dieses Kritikalitätslevels kann zu einem kerntechnischen Unfall führen. Ein in dem Kritikalitätslevel 3 eingestufter Gefährdungszustand kann signifikante Auswirkungen auf den sicheren Betrieb der Anlage haben, führt jedoch nicht zu einem kerntechnischen Unfall. Der Kritikalitätslevel 2 wird Gefährdungszuständen zugeordnet, bei denen der Anlagenbetrieb beeinflusst ist. Der Kritikalitätslevel 1 kennzeichnet sicherheitstechnisch unbedeutsame Gefährdungszustände und wurde in dieser Übersicht nicht aufgenommen.

Tab. 2.4 Systembezogene Gefährdungen mit Softwareaspekten und ihre zugehörigen Kritikalitätslevel für das IDiPS /PAR 07/.

Lfd.-Nr.	**Gefährdungen mit Softwareaspekten**	**Kritikalitätslevel**
1	IDiPS kann kein Anregesignal generieren, obwohl die Anregebedingungen für eine Prozessvariable erfüllt sind.	4
2	IDiPS generiert ein Anregesignal, obwohl es kein Anregesignal generieren sollte.	3
3	IDiPS kann keine ausreichenden Informationen über seinen Betriebszustand an die Warte übermitteln.	2

Die zugrunde gelegten Funktionsmerkmale sowie die verwendeten Leitworte und Abweichungschecklisten zur Durchführung der Software-HAZOP an der Anwendungssoftware des IDiPS im Rahmen der SSA sind beispielhaft in Tab. 2.4 aufgelistet.

Tab. 2.5 Funktionsmerkmale, Leitworte und Abweichungscheckliste für die Software-HAZOP der Anwendungssoftware des IDiPS /PAR 07/

Funktionsmerkmal	Leitwort	Abweichungscheckliste
Accuracy	Below minimum range	What is the consequence if the sensor value is below its minimum range?
Accuracy	Above maximum range	What is the consequence if the sensor value is above its maximum range?
Accuracy	Within range, but wrong	What is the consequence if the sensor value is within its physical range but incorrect?
Accuracy	Incorrect physical units	What is the consequence if the input has an incorrect physical unit?
Accuracy	Wrong data type or data size	What is the consequence if the input has a wrong data type or data size?
Accuracy	Wrong physical address	What is the consequence if the input variable is allocated to a wrong physical address?
Accuracy	Correct physical address, but wrong varia-	What is the consequence if a wrong input variable is allocated to a correct physical address?
Accuracy	Wrong variable type or name	What is the consequence if wrong type or name for an input /output/internal variable is used in the FBD module?
Accuracy	Incorrect variable initialization	What is the consequence if the input/output/ internal variables are initialized incorrectly?
Accuracy	Wrong constant value	What is the consequence if the internal constant is given a wrong value?
Accuracy	Incorrect update of history variables	What is the consequence if the variable is updated incorrectly?
Accuracy	Wrong setpoint calculation	What is the consequence if the procedure for calculating a setpoint is incorrect?
Functionality	Function is not carried out as specified	What is the consequence if some portions in the FBD module have a defect or cannot perform the intended behavior?
Reliability	Data is passed to incorrect process	What is the consequence if the data is passed to an incorrect process?
Robustness	Incorrect selection of test mode	What is the consequence if the test mode is selected or changed unexpectedly?
Robustness	Incorrect input selection	What is the consequence if the input selection is incorrect?

Alle Softwaremodule der Anwendungssoftware des IDiPS wurden in Bezug auf alle Gefährdungszustände gemäß Tab. 2.3 bewertet, indem iterativ alle Aspekte der Abweichungschecklisten (Tab. 2.4) auf jedes Softwaremodul angewendet wurden. Hierbei wurden verschiedene Gefährdungszustände der Software identifiziert, welche die Verfügbarkeit bzw. die Sicherheit des Systems beeinträchtigen können.

Tab. 2.5 zeigt exemplarische Ergebnisse der durchgeführten Software HAZOP. Betrachtet wurde das Logikmodul „SG1_FLW_Lo Trip“ der Anwendungssoftware, welches zur Generierung eines Anregesignals des IDiPS aufgrund von niedrigem Speisewasserdurchsatz im Dampferzeuger 1 eingesetzt wird.

Im Rahmen der Software HAZOP wurde eine Checkliste in Form von Leitfragen abgearbeitet, die sich mit den Auswirkungen möglicher Abweichungen der entsprechenden Anwendungssoftware von den Funktionsmerkmalen „Genauigkeit“, „Kapazität“ und „Funktionalität“ befasst.

Die möglichen Ursachen dieser Abweichungen, ihre Auswirkungen und ihre entsprechend zugeordneten Kritikalitätslevel sind in Tab. 2.6 beispielhaft für das Funktionsmerkmal „Genauigkeit“ aufgelistet. Diese Tabelle soll dem Leser anhand von Beispielen ein besseres Verständnis für Software HAZOPs ermöglichen.

Tab. 2.6 Ergebnisse der Software HAZOP für das Logikmodul „SG1_FLW_Lo“ des IDiPS /PAR 07/.

Funktions-merkmal	Abweichungs-checkliste	Ursache	Analyse	Auswirkung	Krit-L.
Genauigkeit	Welche Konsequenzen hat ein Messwert unterhalb des Mindestbereiches?	Fehler des Messsensors	Das betreffende Untermodul „TRIP_DECISION“ behandelt den außerhalb des Messbereiches liegenden Messwert korrekt.	Schlechte Bedienbarkeit ohne Auswirkungen auf die Sicherheit der Anlage	2
Genauigkeit	Welche Konsequenzen hat ein Messwert oberhalb des Maximalbereiches?	Fehler des Messsensors			
Genauigkeit	Welche Konsequenzen hat eine Konstante mit falschem Wert?	Der Konstanten wurde ein falscher Wert zugewiesen	Falls „MAXCNT“ auf „0“ gesetzt ist, steht das Auslösesignal unabhängig von dem Status der Auslösebedingungen dauerhaft an.	Schlechte Bedienbarkeit	3
			Bei zu großem „MAXCNT“ wird das Auslösesignal erst zu einem späteren Zeitpunkt generiert.	Verletzung der Reaktionszeit des Systems	4

2.4.10 Zusammenfassung und Schlussfolgerungen für das Vorhaben

In diesem Abschnitt 2.4 wurden die folgenden Methoden zur Zuverlässigkeitsanalyse softwarebasierter Leittechnik beschrieben: FMEA, die FTA, die ETA, Markov-Modelle, Petri-Netze, die Fehlerinjektionsmethode (FI) und das HAZOP-Verfahren.

Da der Schwerpunkt des Vorhabens in der Zuverlässigkeitsanalyse der Software softwarebasierter Leittechniksysteme lag, wurden die genannten Methoden insbesondere hinsichtlich deren Eignung zur Analyse von Software betrachtet.

Die Zielsetzung der FMEA ist, die Auswirkungen von möglichen Fehlern in einem System festzustellen. Die FMEA ist ein induktives Verfahren (von einer niedrigen zu einer höheren Systemebene), d. h. die Analyse geht von den Fehlerarten einer Basiskomponente aus und untersucht dann die Einflüsse der Fehlerarten auf die nachfolgenden Systemebenen. Dabei wird angenommen, dass jede Basiskomponente in einen Fehlerzustand geraten kann. Die FMEA kann sowohl auf die Hardware als auch auf die Software softwarebasierter Leittechniksysteme angewandt werden. In der Anwendung der FMEA auf die Software ergeben sich einige Unterschiede zur Hardware-FMEA. Während beispielsweise die Hardware-FMEA auf Funktions- und Komponentenebene durchgeführt werden kann, ist die SFMEA oftmals nur auf Funktionsebene durchführbar. Für die Untersuchung einer Software auf Funktionsebene sollte die zu untersuchende Software aus einzelnen voneinander unabhängigen Modulen aufgebaut sein, die anwendungsspezifisch zusammengefügt werden können (z. B. Anwendersoftware softwarebasierter Leittechniksysteme). Die Softwaremodule stellen dann die Basiskomponenten für die Anwendung der FMEA dar.

Die Fault Tree Analysis kann zur Analyse von Hardware, Software sowie von Wechselwirkungen zwischen Software und Hardware verwendet werden. Zentraler Baustein der FTA ist eine geordnete graphische Darstellung (Fehlerbaum bzw. Fehzustandsbaum) der Bedingungen und Faktoren, die den Eintritt eines unerwünschten Ergebnisses verursachen oder dazu beitragen; dieses wird als Hauptereignis bzw. Topereignis bezeichnet. Die Fault Tree Analysis bedient sich hierbei einer deduktiven Vorgehensweise d. h. von der Fehlerwirkung (Hauptereignis bzw. Topereignis) ausgehend werden die jeweiligen Ursachen der Fehlerzustände/Fehlzustände aufgezeigt, die das Hauptereignis bzw. Topereignis hervorgerufen haben können.

Für die Zuverlässigkeitsanalyse von Leittechniksystemen mit der FTA werden Hardwareaspekte meist auf der Systemebene betrachtet, z. B. durch Analyse der Systemarchitektur bei redundant aufgebauten Leittechniksystemen. Die Analyse von Leittechnikfunktionen (Anwendungssoftware bei softwarebasierten Leittechniksystemen) erfolgt ähnlich wie bei der FMEA auf Funktionsebene. Hierbei werden die Ausfälle von Leittechnikfunktionen (Nichtausführung oder fehlerhafte Ausführung) als Hauptereignisse betrachtet. Die FTA wird dann auf die spezifizierten Leittechnikfunktionen durchgeführt, die in der Regel als Funktionspläne (Logikdiagramme) dargestellt sind.

Die Event Tree Analysis ähnelt der Fehlerbaumanalyse, jedoch werden bei der ETA nicht die Ursachen eines Ereignisses, sondern seine Auswirkungen analysiert. Im Vergleich zur FTA (deduktives Vorgehen) wird bei einer ETA in umgekehrter Reihenfolge vorgegangen und eine induktive Analyse durchgeführt. Mithilfe der ETA können die möglichen Folgen eines auftretenden Fehlers oder des Ausfalls von erforderlichen Sicherheitsfunktionen bestimmt werden. Bei dieser Analyse werden Ereignisse, die zum Ausfall von erforderlichen Sicherheitsfunktionen führen können, ermittelt. Die so ermittelten Ereignisse entsprechen den Hauptereignissen der FTA, deren Ursachen anschließend mittels FTA untersucht werden können. Für die Zuverlässigkeitsanalyse softwarebasierter Leittechnik können somit mit der ETA relevante Hauptereignisse ermittelt werden, die mit Ausfällen von Leittechnikfunktionen verknüpft sind. Die Analyse dieser Leittechnikfunktionen kann mit der FTA entsprechend den Ausführungen im vorherigen Abschnitt durchgeführt werden.

Das Markov-Modell ist ein Zustandsübergangsmodell und wird zur Modellierung von sich zeitlich zufällig verändernden Systemen genutzt. In den meisten Fällen werden Markov-Modelle für quantitative Analysen genutzt. Mit ihrer Hilfe lassen sich Wahrscheinlichkeiten über das Eintreten zukünftiger Zustände voraussagen. Dies betrifft beispielsweise Ausfallwahrscheinlichkeiten, Wahrscheinlichkeiten von Zustandsänderungen und Ausfall- und Reparaturraten von Systemen. Bei der Analyse wird angenommen, dass der zukünftige Zustand eines Systems nur von seinem gegenwärtigen Zustand abhängt.

Es werden beispielsweise die Systemzustände „funktionsfähig“ und „ausgefallen“ mit Angaben von entsprechenden Wahrscheinlichkeiten betrachtet. Die Ursachen für die Systemausfälle bzw. Zustandsänderungen im System werden im Modell nicht betrachtet.

Mithilfe von Petri-Netzen können ähnlich wie mit den Markovmodellen Systemzustände analysiert werden. Die Petri-Netz-Modellierung basiert dementsprechend auf der Definition von Systemzuständen. Zustandsänderungen erfolgen anhand von Schaltregeln, die mit logischen Regeln oder Ereignissen verknüpft sind. Petri-Netze werden insbesondere eingesetzt, um zeitliche Abläufe in Systemen, in denen eine Ressourceneinteilung unabdingbar ist, wie z. B Datenkommunikationsnetzwerke, zu analysieren. Ein weiteres Anwendungsfeld von Petri-Netzen liegt in der Bewertung der Wirksamkeit von fehlertoleranten Maßnahmen in der Software und zur Zuverlässigkeitsanalyse von softwarebasierten Leittechniksystemen auf Systemebene.

Das Fault Injection-Analyseverfahren ist eine experimentelle Methode, um die Zuverlässigkeit eines softwarebasierten Systems zu testen. Ziele des Fault-Injection-Analyseverfahrens sind, das Systemverhalten unter dem Einfluss von vorsätzlich eingeführten Fehlern zu beobachten und dadurch Mängel bei der Fehlertoleranz und -korrektur aufzudecken. Das Fault-Injection-Analyseverfahren kann sowohl auf die Hardware als auch auf die Software angewandt werden. Mit der Hardwarefehlerinjektion können u. a. die Auswirkungen von Bit-Flips (Änderung eines einzelnen oder mehrerer Bits innerhalb eines Datensatzes) z. B. durch Störströme/Störstrahlung ausgelöst, sowie Kurzschlüsse untersucht werden; mit der Softwarefehlerinjektion können beispielsweise die Auswirkungen von Speicherdaten-, Kommunikations- sowie Programmfehlern auf das Systemverhalten analysiert werden.

Das HAZOP-Verfahren ist eine systematische Überprüfungsmethode für Systeme oder Systembestandteile zur Identifizierung von Gefährdungen, Fehldesigns und Fehlfunktionen sowie der Analyse ihrer Auswirkungen auf die Umgebung. Das Verfahren wurde ursprünglich von der chemischen Industrie zur Analyse chemischer Prozesssysteme entwickelt und hat sich inzwischen auch in anderen Anwendungsbereichen bewährt. Beispielsweise kann das HAZOP-Verfahren für die Überprüfung von rechnerbasierten Steuerungssystemen eingesetzt werden. Ziel des Verfahrens ist es, mögliche Abweichungen vom bestimmungsgemäßen Betrieb eines Systems („Sollzustand“) aufzudecken, die jeweiligen Ursachen und Auswirkungen zu benennen (und gegebenenfalls zu bewerten) sowie geeignete Gegenmaßnahmen zur Vermeidung von deren Auswirkungen auf die Systemsicherheit festzulegen. Zentraler Baustein des Verfahrens ist die Anwendung so genannter Leitworte/Leitsätze mittels derer die Abweichungen vom „Sollzustand“ des Systems anhand von Leitfragen formuliert werden. Für die Anwendung des HAZOP-Verfahrens zur Zuverlässigkeitsanalyse von Software werden Funktionsmerkale

zur Charakterisierung des „Sollzustandes“ der zu untersuchenden Software eingeführt, wie z. B. Genauigkeit, Funktionalität, Zuverlässigkeit, Robustheit, Sicherheit und Integrität. Die Software-HAZOP wird durchgeführt, um Gefährdungszustände der Software zu identifizieren, die die Systemsicherheit beeinträchtigen können, wenn eine bestimmte Abweichung der Software vom zu betrachtenden Funktionsmerkmal vorliegt.

Die Zielsetzung des im Rahmen dieses Vorhabens entwickelten Konzepts war es, die Auswirkungen von postulierten Softwarefehlern in Leittechnikfunktionen eines softwarebasierten Leittechniksystems zu untersuchen. Insbesondere sollte mit dem Konzept die Fehlerausbreitung im Leittechniksystem nachgebildet werden können. Dies lässt sich am geeignetsten bei einer Analyse auf Funktionsebene bewerkstelligen. Für die Ermittlung von Ausbreitungspfaden eignen sich induktive Verfahren besser, da diese von einer niedrigen zu einer höheren Systemebene ausgehen. Aus diesen Gründen wurde die SFMEA-Methode für die Konzeptentwicklung im Rahmen des Vorhabens als geeignete Methode betrachtet und gewählt. Dieses Konzept ist im Kapitel 3 beschrieben.

2.5 Betriebserfahrung zur Ausbreitung von Softwarefehlern in softwarebasierten Leittechniksystemen

In diesem Abschnitt wird anhand von drei Ereignissen die Ausbreitung von Softwarefehlern in einem softwarebasierten Leittechniksystem beispielhaft gezeigt. Die Ereignisse werden entsprechend der in Abschnitt 2.1 beschriebenen Modelle zur Fehlerausbreitung (Kaskadenmodell und CCF-Modell) klassifiziert. Bei der Ereignisbeschreibung werden lediglich die zur Klassifizierung relevanten Aspekte beschrieben.

2.5.1 Fehlerbedingte sekundärseitige Lastabsenkung und nicht erfolgter Stabeinwurf

In der betroffenen Anlage waren die Reaktorregelungen und -begrenzungen, sowie die Handansteuerungen von der Warte der entsprechenden Stellglieder in einem digitalen Leittechniksystem realisiert. Die Anlage befand sich im Streckbetrieb kurz vor der Revision. Planmäßig sollte ein Gliederzug der Neutronenflussaußeninstrumentierung gezogen werden.

Entgegen der sonstigen Praxis die Messwerte mit dem oberen Ende des Messbereichs zu simulieren, wurde die Möglichkeit des digitalen Systems genutzt und die vier Messwerte des Leistungsbereichs dieses Gliederzuges als ausgefallen simuliert (Status „fehlerhaft“). Dieses Vorgehen war im Design der Anwendungssoftware des Systems nicht vorgesehen, aber in der Bedienungsanleitung nicht explizit ausgeschlossen.

Da die vier Messwerte über alle Redundanzen verteilt wurden und sich damit der Status „fehlerhaft“ in allen Redundanzen kaskadenartig ausbreitete, wurden alle Ausgänge des Systems zu den Stellgliedern gesperrt bzw. fielen auf „0“ ab. Damit waren weder Regelungs-, Begrenzungs- oder Handmaßnahmen an den Stellgliedern möglich. Der Ausgang der „Oberen Begrenzung des Generatorleistungssollwertes“ fiel ebenfalls auf „0 MW“ ab, was auslegungsgemäß in der Turbinensteuerung einen Lastabwurf auf den doppelten Eigenbedarf mit Öffnen der Frischdampfumleitstation (FDU) bewirkte.

Die FDU kann maximal nur 60% Leistung auf der Sekundärseite abführen. Dies führte zu einem großen Ungleichgewicht zwischen der abgeführten Leistung auf der Sekundärseite und der Reaktorleistung, so dass es zur Aufheizung des Primärkreises und in der Folge u. a. zur Erhöhung der Kühlmitteltemperatur kam. Da zum Ereigniszeitpunkt die

Anlage sich am Zyklusende befand, war der Kühlmitteltemperaturkoeffizient stark negativ. Aufgrund der damit einhergehenden stärkeren reaktivitätsbindenden Rückwirkung (im Vergleich zum Zyklusbeginn) des Reaktorkerns stabilisierte sich der Reaktor durch die beschriebenen Ausfälle inhärent auf einem neuen Leistungsniveau, ohne dass Reaktorschutzgrenzwerte erreicht wurden. Eine Reaktorschnellabschaltung von Hand wäre zu jedem Zeitpunkt über das autarke, festverdrahtete Reaktorschutzsystem ausführbar gewesen.

Nach ca. 14 Minuten wurden die Simulierungen von Hand aufgehoben, alle Funktionen standen wieder zur Verfügung und der Anlagenzustand konnte wieder normalisiert werden. Die Reaktorleistung wurde zunächst zur Beendigung des FDU-Betriebs und Normalisierung der Anlage entsprechend abgesenkt. Anschließend erfolgte nach Ursachenklärung die Freigabe zur Leistungserhöhung.

Die betroffene Reaktorleistungsbegrenzung wurde 1998 von festverdrahteter auf softwarebasierte Leittechnik umgerüstet. Bei der Projektierung der neuen Leittechnik wurde nicht berücksichtigt, dass eine Simulierung von Signalen anders als in der analogen Ausführung durch Setzen des Status „fehlerhaft“ erfolgt. Die Auswirkungen der Ausbreitung eines Status „fehlerhaft“ in den leittechnischen Funktionen wurden daher nicht untersucht, so dass im Design des Systems keine Vorkehrungen dagegen getroffen wurden. Die Durchführung einer Simulierung durch Setzen des Status „fehlerhaft“ wurde allerdings in der Bedienungsanleitung des Systems auch nicht explizit verboten.

Die Betreiberin hat nach dem Ereignis die leittechnische Fachanweisung für die Simulation überarbeitet und zusätzliche Bausteine zur Blockierung von Signalen mit dem Status „fehlerhaft“ in die Anwendungssoftware des softwarebasierten Leittechniksystems integriert.

Durch diese Bausteine wird die Ausbreitung der Signale mit dem Status „fehlerhaft“ begrenzt. Darüber hinaus wurde in den leittechnischen Funktionsplänen das Blockieren der Signale mit dem Status „fehlerhaft“ gekennzeichnet. In der Dokumentation der Funktionsbausteine wird in einem einleitenden Abschnitt auf die Ausbreitung und Begrenzung von Signalen mit dem Status „fehlerhaft“ hingewiesen.

Kommentar:

Bei diesem Ereignis kam es entsprechend dem im Abschnitt 2.1 vorgestellten Kaskadenmodell zur Ausbreitung eines Status „fehlerhaft“ in allen leittechnischen Funktionen der Reaktorregelung- und -begrenzung. Der zunächst im Design der Anwendungssoftware vorhandene latente Fehler (Nichtberücksichtigung der Möglichkeit der Simulierung des Status „fehlerhaft“ an mehreren Messwerten) breitete sich nach seiner Aktivierung in allen Anwendungsfunktionen des leittechnischen Systems aus.

2.5.2 Baugruppenfehler in einer Brandmeldezentrale

Im August 2016 wurde in einem Kernkraftwerk im Rahmen einer vierteljährlichen wiederkehrenden Prüfung (WKP) der Brandmeldeanlage für das Kühlwasserpumpenhaus und das Hilfskesselgebäude festgestellt, dass nach Anregung der Brandmelder mittels eines Meldeprüfers die Signale bei insgesamt sieben Brandmeldeleitungen nicht weitergeleitet wurden. Alle Signale dieser sieben Brandmeldeleitungen werden auf einer gemeinsamen Anschaltbaugruppe der Firma Siemens verarbeitet. Von der betroffenen Baugruppe werden die Signale von insgesamt acht Stichleitungen mit jeweils sechs Brandmeldern verarbeitet. Die betroffene Baugruppe wurde getauscht. Danach war die Funktion wieder gegeben.

Nach dem Ereignis während der WKP der Brandmeldeanlage im August 2016, kam es einige Tage später während der WKP der Brandmeldeanlage für das Verwaltungsgebäude erneut zu einem Ereignis mit gleichartigem Erscheinungsbild. Wieder wurden die Signale von mehreren prüfbedingt ausgelösten Brandmeldern nicht weitergeleitet.

Auch diese Signale werden auf einer gemeinsamen Anschaltbaugruppe verarbeitet. Die Baugruppe hatte einen neueren Revisionsstand als die vom ersten Ereignis betroffene Baugruppe. Dies bedeutet u. a., dass unterschiedliche Versionen der Firmware der Anschaltbaugruppe im Kernkraftwerk Verwendung finden. Grundsätzlich sind beide Revisionsstände laut Hersteller miteinander kompatibel. Die Baugruppe wurde vorerst nicht getauscht, um dem Hersteller die Klärung der Störungsursache vor Ort zu ermöglichen.

Die Funktionsstörungen der betroffenen Baugruppen konnten zunächst nicht unmittelbar reproduziert werden. Detailliertere Untersuchungen des Herstellers (praktische Tests und Code-Screening) zeigten, dass die Störungen auf Fehler in der Firmware der Baugruppe zurückzuführen sind.

Mit diesen Erkenntnissen konnte das Verhalten der Brandmeldeanlage dann auch auf Laborsystemen reproduziert werden. Die Baugruppe kann in einen fehlerhaften Zustand gelangen, werden während der Initialisierung der Baugruppe in einer bestimmten Phase per Handbefehl Meldeleitungen an- und abgeschaltet. In diesem fehlerhaften Zustand sind Meldeleitungen ungewollt abgeschaltet, ohne dass dies an der Anzeigeeinrichtung ersichtlich wird. Bei der älteren eingesetzten Version der Firmware kann dies nach Erreichen des fehlerhaften Zustandes auch Meldeleitungen betreffen, die nicht während der kritischen Phase der Initialisierung, sondern erst danach, per Hand geschaltet wurden. Es besteht technisch keine Notwendigkeit, während des Initialisierungsprozesses Meldeleitungen an- oder abzuschalten.

Die genaue Abfolge der zum Erreichen des fehlerhaften Zustandes notwendigen Handbefehle ist abhängig von der Version der Firmware. Generell sind aber ein bis zwei Handbefehle bei allen Firmwareversionen ausreichend, um einen fehlerhaften Zustand zu erzeugen. Die Dauer des Zeitfensters, innerhalb dessen diese Handbefehle gegeben werden müssen, beträgt, abhängig von der Firmwareversion und der Anzahl der angeschlossenen Melder, zwischen wenigen Sekunden und über eine Minute. Bei der neuesten Version der Firmware ist außerdem eine Mindestanzahl an angeschlossenen Meldern notwendig, um den Übergang in den fehlerhaften Zustand zu ermöglichen. Grundsätzlich sind aber alle Versionen der Firmware von dem Fehler betroffen.

Der fehlerhafte Zustand kann durch eine erneute, korrekte Durchführung der Initialisierung behoben werden.

Der Betreiber hat nach dem ersten Ereignis zunächst die betroffene Anschaltbaugruppe getauscht. Später wurden alle Baugruppen mit dem älteren Revisionsstand der Firmware gegen Baugruppen mit dem neuen Revisionsstand getauscht. Nach dem zweiten Ereignis wurden Sonderprüfungen aller Brandmeldeanlagen mit der entsprechenden Technik durchgeführt. Zwischenzeitlich wurden zusätzliche Begehungen als absichernde Maßnahme durchgeführt.

Der Hersteller hat als Abhilfemaßnahme eine Service-Anweisung an alle Unternehmen, die die betroffene Gefahrenmeldeanlagen selbst warten oder zur Wartung berechtigt sind, verschickt, in der auf die Problematik hingewiesen wird.

In der Anweisung wird empfohlen, während der Initialisierung der Gefahrenmeldeanlage keine Handeingaben vorzunehmen und in Betrieb befindliche Gefahrenmeldeanlagen, bei denen nicht ausgeschlossen werden kann, dass es bei der letzten Initialisierung zu Handeingaben kam, neu zu initialisieren.

Kommentar:

Bei diesem Ereignis führten die zunächst in einer Anschaltbaugruppe in einer Brandmeldeanlage vorhandenen latenten Fehler (Fehlerhafte Berücksichtigung der An- und Abschaltung von Meldeleitungen per Handbefehl während der Initialisierung der Baugruppe in der Firmware, Mängel in der Dokumentation der Anschaltbaugruppe) nach ihrer Aktivierung (Anschaltung von Brandmeldeleitungen per Handbefehl im Rahmen einer WKP) zum Ausfall mehrerer Brandmeldeleitungen und der daran angeschlossenen Brandmelder. Zudem belegt die fehlende Fehlermeldung bei abgeschalteten Meldeleitungen ein nicht ausgereiftes Design des Systems und mangelnde Robustheit. Aufgrund der dezentralen Anordnung der Brandmeldeanlage begrenzte sich die Auswirkung des Fehlers zunächst auf die an die betroffene Anschaltbaugruppe angeschlossenen Brandmelder. Da jedoch alle Anschaltbaugruppen vom gleichen Typ waren und demzufolge von dem Firmwarefehler betroffen waren, ist in diesem Fall von einer potenziellen Ausbreitung des Firmwarefehlers nach dem CCF-Modell gemäß Abschnitt 2.1 auszugehen.

2.5.3 Vorübergehende Unverfügbarkeit der Bedienstationen eines rechnerbasierten Informations- und Steuerungssystems

Block-1 der betroffenen Anlage befand sich im Volllastbetrieb. Während der planmäßigen Durchführung der Funktionsprüfung eines Parameters eines rechnerbasierten Informations- und Steuerungssystems wurde festgestellt, dass nach Anwählen des Symbols zum Setzen des zu prüfenden Parameters keine Bedienstation des rechnerbasierten Informations- und Steuerungssystems mehr verfügbar war. Daraufhin führte ein Schichtmitarbeiter die bei Unverfügbarkeit der Bedienstation vorgesehenen Prozeduren zur Umstellung von den Bedienstationen des betroffenen rechnerbasierten Steuerungssystems auf das zugehörige Backup-System erfolgreich durch, um darüber den Fehler zu beheben. Nach etwa 10 Minuten waren die Bedienstationen wieder in Funktion.

Direkter Auslöser des Ereignisses war eine Fehlkonfiguration des zum geprüften Parameter gehörenden Objekts des betroffenen rechnerbasierten Steuerungssystems.

Diese Fehlkonfiguration führte dazu, dass beim Anwählen des dem zu prüfenden Parameter zugeordneten Symbols, das Signal eines anderen Parameters an das Steuerungssystem gesendet wurde. Das Steuerungssystem erkannte den möglicherweise zu falschen Handlungen führenden falschen Befehl während der aktuellen Anwendung und leitete auslegungsgemäß einen automatischen Stopp eines dem System vorgelagerten Servers ein, wodurch die Bedienstationen unverfügbar wurden. Der Gerätehersteller hatte diesen Mechanismus nicht in den zum Server mitgelieferten Dokumenten beschrieben.

Der Entscheidungsmechanismus, der zum Stoppen des Servers führte, soll mit dem Hersteller in Bezug auf seine Verhältnismäßigkeit diskutiert werden. Der fehlerhaft konfigurierte Parameter wurde vorübergehend blockiert und während Reparaturarbeiten rekonfiguriert. Zudem soll die Ereignisursache mithilfe von Simulationen in Block-2 verifiziert werden.

Kommentar:

Bei diesem Ereignis führten die zunächst latenten Fehler (fehlerhafte Zuordnung eines Befehls zu einem Bedienelement, Mängel in der Dokumentation) in einem rechnerbasierten Informations- und Steuerungssystem nach ihrer Aktivierung (Durchführung einer Funktionsprüfung eines Parameters) zum Ausfall von Bedienstationen des zugehörigen rechnerbasierten Informations- und Steuerungssystems. Ausgehend von der Bedienstation, an der der fehlerhafte Befehl getätigt wurde, breitete sich der Fehler in diesem Fall entsprechend dem Kaskadenmodell (Abschnitt 2.1) im rechnerbasierten Informations- und Steuerungssystem aus und führte zur vorübergehenden Unverfügbarkeit aller Bedienstationen des Systems.

3 Entwicklung eines Konzeptes zur Untersuchung von Softwarefehlern in softwarebasierter Leittechnik

3.1 Einleitung

In diesem Kapitel wird das im Rahmen des Arbeitspaketes 2 entwickelte Konzept zur Untersuchung von Ausfällen aufgrund von Softwarefehlern in softwarebasierten Leittechniksystemen dargestellt.

Das Ziel des entwickelten Konzeptes ist es, potenzielle Ausfälle aufgrund latenter Softwarefehler in einem softwarebasierten Leittechniksystem zu identifizieren und deren Auswirkungen auf die Funktion des Leittechniksystems (Ausfall auf Anforderung, Fehlbetätigung usw.) zu untersuchen. Bei dieser Untersuchung werden die Ausbreitungspfade der identifizierten Ausfallarten analysiert.

Hierfür wurde zunächst ausgehend von einem generischen Modell eines Leittechniksystems, der Untersuchungsrahmen (siehe Abschnitt 3.3) für das Konzept festgelegt. Anschließend wurde das Konzept zur Untersuchung der Ausbreitung von Softwarefehlern und deren Auswirkungen in einem softwarebasierten Leittechniksystem spezifiziert. Hierbei wurden u. a. die zu betrachtende Software und die anzuwendende Zuverlässigkeitsanalysemethode festgelegt. Die erzielten Ergebnisse dieser Arbeitsschritte sind in den Abschnitten 3.2 bis 3.4 dargestellt.

Im Abschnitt 3.5 sind einige Anwendungsbeispiele des entwickelten Analysekonzeptes dargestellt.

3.2 Generisches Modell/Generische Beschreibung eines softwarebasierten Leittechniksystems

Das im Rahmen dieses Arbeitspaketes entwickelte Konzept baut auf dem in /MBO 16/, /GRS 17/ dargestellten generischen Modell eines Leittechniksystems auf (s. Abb. 3.1).

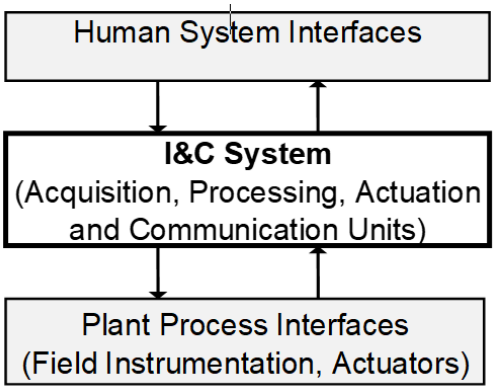

Abb. 3.1 Vereinfachtes Modell eines Leittechniksystems mit Mensch-Maschinen- und Prozessschnittstellen

Ein Leittechniksystem besteht aus aufeinander abgestimmten Komponenten, die eine Reihe von spezifizierten Leittechnikfunktionen in Zusammenhang mit dem zu steuernden Prozess bzw. dem Anlagenprozess, sogenannte Anwendungsfunktionen, wie z. B. Überwachung, Steuerung und/oder Schutz, ausführen.

Darüber hinaus führt ein Leittechniksystem auch Funktionen aus, die mit dem Betrieb des Leittechniksystems selbst zusammenhängen, sogenannte Systemfunktionen, wie z. B. interne Diagnose.

Das Zusammenwirken eines Leittechniksystems mit dem Anlagenprozess bzw. mit dem zu steuernden Prozess erfolgt über Eingangssignale von der Feldinstrumentierung und über Ausgangssignale an Aktuatoren (Prozess- oder Feldschnittstelle) und/oder über Signale von/zu den Operateuren (Mensch-Maschine-Schnittstelle), wie in Abb. 3.1 dargestellt.

Entsprechend den genannten Aufgaben (Ausführung von Anwendungs- und Systemfunktionen) lässt sich die Funktion eines Leittechniksystems prinzipiell anhand von drei Funktionsebenen beschreiben: **Eingabeebene, Verarbeitungsebene** und **Ausgabeebene**.

Die **Eingabeebene** schließt alle Komponenten ein, die für die Generierung von Eingabedaten aus dem zu steuernden Prozess für das Leittechniksystem verwendet werden. Dazu zählen Geräte zur Erfassung von Messgrößen (z. B. Sensoren/Detektoren, Messumformer) und zur Eingabe von Daten (Bedienelemente wie z. B. Taster, Schalter).

Auf der **Verarbeitungsebene** werden die Daten aller Komponenten verarbeitet, die aus der Eingabeebene des Leittechniksystems zur Verfügung gestellt werden. Die Verarbeitung der Daten erfolgt nach einer entsprechend der auszuführenden leittechnischen Funktion festgelegten Verarbeitungsvorschrift. Dabei kann es sich beispielsweise um die Grenzwertbildung, logische Operationen, den Signalvergleich oder die Berechnung der Ausgangssignale für die Ausgabeebene handeln.

Der **Ausgabeebene** des generischen Leittechniksystems werden alle Komponenten zugeordnet, die für die Umformung der Signale aus der Verarbeitungsebene und für die Ansteuerung der verfahrenstechnischen Komponenten (Aktuatoren), wie beispielsweise Ventile, Schieber, Pumpen, Motoren und Schalter, zur Beeinflussung der Anlagenparameter genutzt werden. Baugruppen für die Prioritätssteuerung der Signale aus der Verarbeitungsebene werden hier ebenso betrachtet. Die Komponenten der Ausgabeebene können funktionell in Steuerungseinrichtungen und in Anzeige- und Meldeeinrichtungen unterteilt werden. Sie bilden die Schnittstelle zwischen dem Leittechniksystem und dem zu steuernden Prozess bzw. dem Bedienpersonal.

Den zuvor genannten Funktionsebenen werden in der Regel die folgenden Funktionseinheiten zugeordnet: Signalerfassungseinheit (Eingabeebene), Signalverarbeitungseinheit (Verarbeitungsebene), Betätigungseinheit (Ausgabeebene). Zum Datenaustausch zwischen den Funktionseinheiten eines Leittechniksystems werden Kommunikationseinheiten eingesetzt.

Bei einem softwarebasierten Leittechniksystem bestehen die Funktionseinheiten aus digitalen Modulen wie Eingangsmodul, Verarbeitungsmodul, Ausgangsmodul und Kommunikationsmodul und ihrer zugehörigen Software. Die Grundkomponenten der Module sind in der Regel Analog-Digital-Wandler (A/D-Wandler), Digital- Analog-Wandler (D/A-Wandler), Multiplexer, Mikroprozessoren, Mikrocontroller und programmierbare elektronische Geräte.

Dementsprechend werden in einem softwarebasierten Leittechniksystem die Leittechnikfunktionen (System- und Anwendungsfunktionen) mit digitalen Modulen und ihrer zugehörigen Software ausgeführt.

Die Systemsoftware ist hierbei der Teil der Software des softwarebasierten Leittechniksystems, der die Systemfunktionen implementiert. Die Systemsoftware besteht unter anderem aus dem Betriebssystem der zugrundeliegenden leittechnischen Plattform, der Kommunikationssoftware und der Firmware der digitalen Module. Die Anwendungssoftware hingegen ist der Teil der Software, in der die Anwendungsfunktionen des softwarebasierten Leittechniksystems in Bezug auf den zu steuernden Prozess implementiert sind. Darüber hinaus wird sogenannte Engineering-Software für das Design, die Wartung und die Modifikationen des softwarebasierten Leittechniksystems verwendet.

3.3 Untersuchungsrahmen für das entwickelte Konzept

Die Systemsoftware wird in der Regel in den entsprechenden Modulen des Leittechniksystems implementiert und bleibt während des Lebenszyklus des Leittechniksystems nahezu unverändert.

Die Anwendungssoftware resultiert aus verschiedenen funktionalen Anforderungsspezifikationen, die u. a. auf der sicherheitstechnischen Bedeutung und der Zuverlässigkeit der zu implementierenden Leittechnikfunktionen (LEFU) beruhen. Die Auswertung der Betriebserfahrung mit softwarebasierten Leittechniksystemen hat bereits Ereignisse im Zusammenhang mit Fehlern in funktionalen Anforderungsspezifikationen und bei der Umsetzung der Anforderungsspezifikationen in Anwendungssoftware im Rahmen der Systementwicklung aufgezeigt. Diese Fehler können sowohl in digitalen als auch in analogen Systemen auftreten. Ein weiteres Fehlerpotential ergibt sich aus der Tatsache, dass während des Betriebs des Leittechniksystems Parameter wie Sollwerte/Grenzwerte modifiziert werden können. Zudem können während des Betriebs leittechnische Funktionspläne geändert werden. Es ist daher davon auszugehen, dass die Anwendungssoftware eines softwarebasierten Leittechniksystems potenziell häufiger von Fehlern betroffen sein wird als die Systemsoftware. Aus diesem Grund wurde, im Rahmen der Entwicklung des Konzeptes zur Untersuchung von potenziellen Ausfällen aufgrund latenter Softwarefehler in einem softwarebasierten Leittechniksystem, die Anwendungssoftware betrachtet.

3.4 Spezifikation der Anwendungssoftware softwarebasierter Leittechniksysteme

Die LEFUs moderner softwarebasierter Leittechniksysteme werden anhand von FBD /BEL 03/ entworfen. Darauf aufbauend wird der Code generiert, der auf dem Leittechniksystem ausgeführt werden kann. In einem FBD wird die LEFU zwischen den entsprechenden Eingangs- und Ausgangssignalen des Leittechniksystems als miteinander verbundene elementare Funktionsblöcke (EFB) grafisch dargestellt, wobei jeder EFB eine definierte Funktion zwischen Eingangs- und Ausgangssignalen beschreibt.

Jeder elementare Funktionsblock hat eine bestimmte Anzahl von Eingangs- und Ausgangssignalen.

Außerdem werden für jeden elementaren Funktionsblock der Algorithmus für die zugrundeliegende Funktion, die Art und der Bereich der Eingangs- und Ausgangssignale, die für die Berechnung der Funktion notwendigen Parameter und der Name des elementaren Funktionsblocks angegeben.

Abb. 3.2 zeigt eine grafische Darstellung eines EFB mit *n* Eingangssignalen $I_1, I_2, I_3 \ldots I_n$, *m* Ausgangssignalen $O_1, O_2, O_3 \ldots O_m$ und *p* Parameter $p_1, p_2, \ldots p_p$. Der EFB ist auch durch seinen Namen <*EFB_name*> gekennzeichnet, der sich auf die implementierte Funktion bezieht, z. B. AND (logische UND-Verknüpfung) oder OR (logische ODER-Verknüpfung). Ein weiteres wichtiges Merkmal von EFBs ist die Berücksichtigung von Signalzuständen der Eingangssignale bei der Berechnung der Ausgangssignale. Der Signalstatus ist ein boolescher Wert, der angibt, ob der Signalwert eines Eingangssignals für die Berechnung der Ausgangssignale (potenziell) fehlerbehaftet ist.

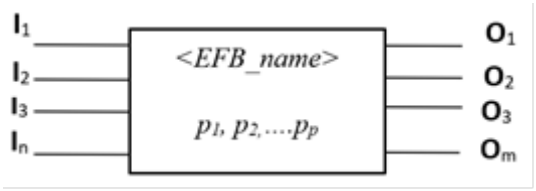

Abb. 3.2 Grafische Darstellung eines elementaren Funktionsblocks

Abgebildet hier mit *n* Eingangssignalen $I_1, I_2, I_3 \ldots I_n$ und *m* Ausgangssignalen $O_1, O_2, O_3 \ldots O_m$ und *p* Parametern $p_1, p_2, \ldots p_p$

Die EFBs sind Teil der Engineering-Software der Leittechniksystemplattform und werden als einzelne Softwaremodule einer Softwarebibliothek implementiert, aus der die Leittechnikfunktionen spezifiziert werden. Auch die Berechnungsalgorithmen der EFBs sind in der Leittechnikdokumentation spezifiziert.

Abb. 3.3 zeigt ein Beispiel für ein Funktionsblockdiagramm einer Leittechnikfunktion unter Verwendung typischer elementarer Funktionsblöcke. In der Leittechnikfunktion in Abb. 3.3 werden dreifach redundante Füllstandsmessungen an einem Behälter auf MAX-Überschreitung und MIN-Unterschreitung überwacht und nach unterschiedlichen Zeitverzögerungen ein Signal ausgelöst. Diese Funktion ist der Bildung des Signals „Durchdringungsabschluss am Sicherheitsbehälter“ (DDA) eines SWR nachempfunden.

Wie Abb. 3.3 zu entnehmen ist, sind Eingangs- und Ausgangssignale eines Funktionsblockdiagramms mit den elementaren Funktionsblöcken durch Verbindungsleitungen verbunden, die den Signalfluss in der Leittechnikfunktion darstellen. Ein Ausgang eines elementaren Funktionsblocks kann auch mit einem Eingang eines anderen elementaren Funktionsblocks verbunden sein.

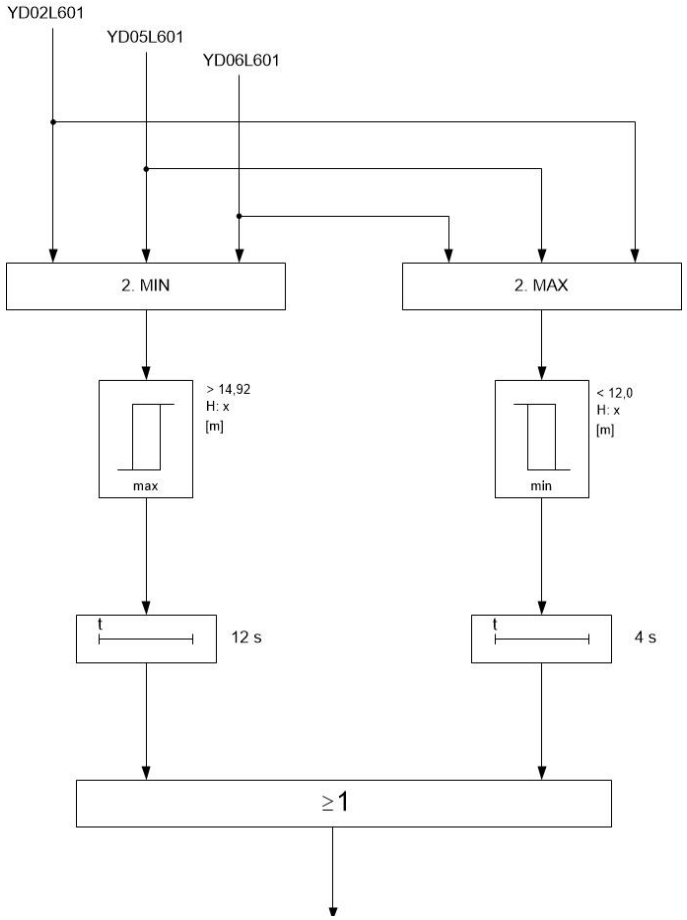

Abb. 3.3 Beispiel eines Funktionsblockdiagramms einer Leittechnikfunktion

Die Funktionsblockdiagramme der Leittechnikfunktionen werden in der Regel mit der Engineeringsoftware des Leittechniksystems implementiert, die Bestandteil der Leittechnikplattform ist.

Ausgehend von der grafischen Darstellung der Leittechnikfunktionen wird dann der ausführbare Code der Anwendungssoftware generiert.

3.5 Konzept zur Untersuchung von potenziellen Ausfällen in softwarebasierter Leittechnik

3.5.1 Überblick über das Analysekonzept

Das entwickelte Konzept zur Untersuchung von potenziellen Ausfällen in einem softwarebasierten Leittechniksystem aufgrund latenter Softwarefehler basiert auf dem im Abschnitt 2.4 beschriebenen FMEA-Verfahren. Im Rahmen des entwickelten Konzeptes wird die FMEA auf die Anwendungssoftware eines softwarebasierten Leittechniksystems angewandt.

Bei der Durchführung einer FMEA an einem Untersuchungsgegenstand ist zunächst die zu berücksichtigende Abstraktionsebene festzulegen, d. h. die Ebene, auf der die FMEA durchgeführt werden soll.

Zu diesem Zweck wurden im Rahmen der Konzeptentwicklung die folgenden Annahmen getroffen:

- Da die Leittechnikfunktionen softwarebasierter Leittechniksysteme aus ihrer Blockdiagrammdarstellung unter Verwendung von EFBs, wie in Abschnitt 3.4 erwähnt, generiert werden, betrachten wir in Analogie zu einer Hardware-FMEA die EFBs als die grundlegenden Elemente, aus denen die Leittechnikfunktionen generiert werden. Aus diesem Grund werden die EFBs als die niedrigste Abstraktionsebene für die Durchführung der FMEA betrachtet.
- Das Softwaremodul jedes einzelnen EFB funktioniert wie spezifiziert. Daher werden potenzielle interne Fehler der EFBs von der weiteren Betrachtung ausgeschlossen. Der EFB wird daher als eine Black-Box betrachtet.

Unter Berücksichtigung dieser Annahmen wird die FMEA der EFBs auf der Funktionsebene in Bezug auf die Eingangs- und Ausgangssignale und die Parameter der EFBs durchgeführt. Die Ausfallarten der Ausgangssignale der EFBs sind mit den Ausfallarten der Eingangssignale und der Parameter der EFBs verknüpft. Sobald die Ausfallarten der entsprechenden Signale und Parameter der EFBs identifiziert sind, können die Auswirkungen dieser Ausfallarten auf die Leittechnikfunktion in einem zweiten Schritt (z. B. mit der Methode der Fehlerbaumanalyse oder der dynamischen Simulation) analysiert werden. Auch mögliche Fehlerausbreitungspfade in der Leittechnikfunktion können evaluiert werden.

In Abb. 3.4 ist ein Überblick des entwickelten Konzeptes zur Bewertung von potenziellem Fehlverhalten der Anwendungssoftware softwarebasierter Leittechniksysteme, kurz SFMEA-Konzept genannt, dargestellt.

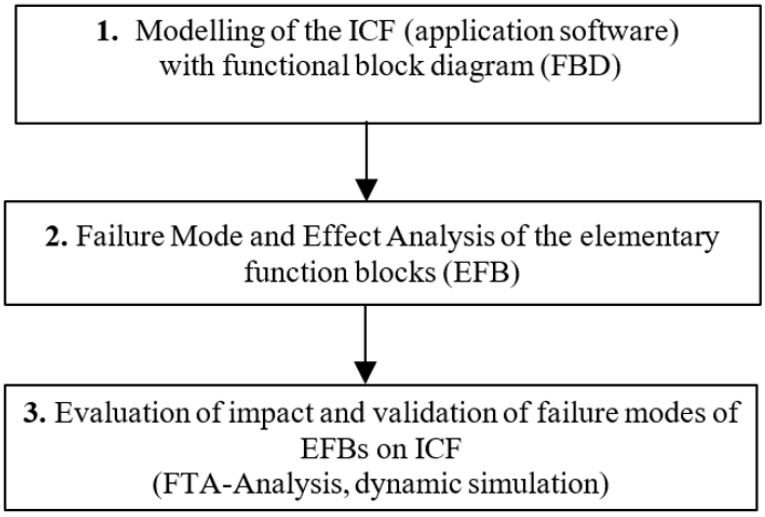

Abb. 3.4 Überblick über das entwickelte Analysekonzept nach /MBO 19/

Zur Anwendung des Konzepts wird zunächst die zu untersuchende LEFU (Anwendungssoftware) entsprechend den Ausführungen in Abschnitt 3.4 modelliert.

Hierzu kann die Engineering-Software des softwarebasierten Leittechniksystems eingesetzt werden. Es besteht auch grundsätzlich die Möglichkeit, andere Software-Tools zur Modellierung der LEFU einzusetzen.

Anschließend wird eine FMEA auf der Ebene der elementaren Funktionsblöcke durchgeführt. Dies dient dazu, die Auswirkungen relevanter Ausfallarten der Eingangssignale und Parameter der elementaren Funktionsblöcke auf die Ausgangssignale für den jeweiligen elementaren Funktionsblock zu identifizieren.

Die Evaluierung der Auswirkungen der identifizierten Ausfallarten der Eingangssignale und Parameter der elementaren Funktionsblöcke auf die zu untersuchende Leittechnikfunktion wird anschließend mittels dynamischer Simulation oder Fehlerbaumanalyse durchgeführt.

3.5.2 FMEA auf Ebene der elementaren Funktionsblöcke

Ziel ist es die Auswirkungen potenzieller Ausfallarten der Eingangssignale und der Parameter der EFBs auf die Ausgangssignale der EFBs zu analysieren.

3.5.2.1 FMEA der Eingangssignale der EFBs

In der Anwendungssoftware typischer softwarebasierter Leittechniksysteme haben verarbeitete Signale im Allgemeinen zwei Attribute: Einen Signalzustand/Signalstatus (STAT) und einen Signalwert (VAL). Die FMEA der Eingangssignale ist unter Berücksichtigung der Annahmen aus Abschnitt 3.5.1 daher hinsichtlich dieser beiden Attribute durchzuführen. Wie in Abschnitt 3.4 erwähnt, ist der Signalstatus (STAT) ein boolescher Wert, der angibt, ob der Signalwert (VAL) für die Berechnung der LEFU gültig (boolescher Wert "0" für STAT) oder nicht gültig (boolescher Wert "1" für STAT) ist. Daraus folgt, dass der Signalstatus STAT die Berechnung der EFB-Ausgangssignale wesentlich beeinflussen kann und daher für die Identifizierung spezifischer Ausfallarten der Eingangssignale der EFBs von Bedeutung ist.

Der Signalzustand STAT wird den Eingangssignalen über die Spezifikation der Leittechnikfunktion während des Software-Engineering-Prozesses im Rahmen von fehlertoleranten Maßnahmen und/oder der Fehlerbehandlung zugeordnet. Einem Signal wird beispielsweise bei erkannter fehlerhafter Kommunikation zwischen den betreffenden Leittechnikbaugruppen ein fehlerhafter Status zugewiesen.

Befindet sich die dem Signal vorgeschaltete Verarbeitungseinheit im Testmodus, wird auch dem betreffenden Signal ein Fehlerstatus zugewiesen. Ergibt sich das Signal aus einer mathematisch ungültigen Berechnung, wird dem Signal ebenfalls ein Fehlerstatus zugeordnet.

Während Signale mit dem Status "0" ohne jede Einschränkung für die Berechnung der Ausgangssignale der EFBs verarbeitet werden, werden Signale mit dem Status "1" je nach EFB-Typ unterschiedlich behandelt. Für die Verarbeitung von Signalen mit dem Signalstatus "1" in softwarebasierten Leittechniksystemen sind die folgenden Methoden üblich:

- Der Status "1" eines Signals wird passiv an das Ausgangssignal des EFB und damit an die direkt nachfolgenden elementaren Funktionsblock weitergeleitet.
- Der Signalwert eines Signals mit dem Status "1" wird vor der Verarbeitung des Ausgangssignals der EFB durch einen Ersatzwert ersetzt.
- Das Signal mit dem Status "1" wird "ausgefiltert", d. h. das Signal mit dem Status "1" wird bei der Berechnung der Ausgangssignale der EFB nicht berücksichtigt.

Um den Einfluss potenzieller fehlerhafter Signalzustände STAT eines Eingangssignals auf die Ausgangssignale einer EFB zu bewerten, wird zusätzlich das Attribut VAL-STAT als booleschen Wert innerhalb der SFMEA-Methode eingeführt. Hierbei liegt ein fehlerhafter Signalzustand eines Eingangssignals vor, wenn der dem Eingangssignal vom Leittechniksystem zugewiesene boolesche Wert für STAT fehlerhaft ist. Das Attribut VAL-STAT zeigt den Status des Signalwertes VAL unter Berücksichtigung eines "echten" unvoreingenommenen Anlagenprozesses unabhängig vom Attribut STAT an. VAL-STAT ist wie folgt definiert:

- Das Attribut VAL-STAT wird dem booleschen Wert "0" zugeordnet, wenn der Signalwert VAL dem korrekten Wert entspricht (korrekter Messwert bei analogen Eingangssignalen, korrekter "Schaltzustand" bei binären Eingangssignalen).
- Das Attribut VAL-STAT wird dem booleschen Wert "1" zugeordnet, wenn der Signalwert VAL nicht dem richtigen Wert entspricht (falscher Messwert bei analogen Eingangssignalen, falscher "Schaltzustand" bei binären Eingangssignalen).

Für die FMEA der Eingangssignale der elementaren Funktionsblöcke wird der resultierende Signalzustand dann aus Kombinationen des Attributs STAT (aus dem Leittechniksystem) und des eingeführten Attributs VAL-STAT wie folgt bewertet: Ein Signalzustand ist korrekt, wenn STAT und VAL-STAT den gleichen booleschen Wert haben. Ein Signalzustand ist fehlerhaft, wenn STAT und VAL-STAT verschieden sind.

In Tab. 3.1sind alle möglichen Kombinationen der Signalattribute STAT und VAL-STAT und die daraus resultierenden ausgewerteten Signalzustände aufgelistet. Die "0"- und "1"-Einträge in Tab. 3.1 entsprechen den booleschen Werten der Attribute STAT und VAL-STAT, wie in den vorhergehenden Absätzen definiert.

Tab. 3.1 Bewertung der Signalzustände für die FMEA der EFB-Eingangssignale in Abhängigkeit vom Signalwert VAL

VAL	STAT	VAL-STAT	Ausgewerteter Signalstatus
a) richtig	0	0	Korrekter „0“-Status
b) falsch	**0**	**1**	**Fehlerhafter „0“-Status**
c) richtig	**1**	**0**	**Fehlerhafter „1“-Status**
d) falsch	1	1	Korrekter „1“-Status

Der fehlerhafte "0"-Signalstatus in Tab. 3.1 kennzeichnet den Fall, dass ein fehlerhafter Signalwert (VAL-STAT "1") für die Verarbeitung des EFB-Ausgangs gültig ist (STAT "0"). Der fehlerhafte "1"-Signalstatus in Tab. 3.1 kennzeichnet den Fall, dass ein korrekter Signalwert (VAL-STAT “0") für die Berechnung des EFB-Ausgangs nicht gültig ist (STAT “1"). Diese fehlerhaften Signalzustände stellen Ausfallmodi der Eingangssignale des EFB dar.

Im vorgestellten SFMEA-Ansatz wird angenommen, dass diese fehlerhaften Signalzustände zu fehlerhaften Ausgangssignalen der EFBs führen können. Alle Kombinationen von Signalzuständen der Eingangssignale eines EFB werden dann bestimmt und ihre Auswirkungen auf die Ausgangssignale des EFB bewertet.

Der Signalwert VAL entspricht bei analogen Eingangssignalen (Temperatur, Druck, Füllstand) dem quantisierten gemessenen Wert, und bei binären Eingangssignalen dem booleschen Wert, beispielsweise für einen Schaltzustand (EIN/AUS) oder eine Komponentenstellung (AUF/ZU). Der Signalwert VAL ist auf den zu steuernden Anlagenprozess bezogen. Abweichungen der Signalwerte vom wahren Prozesswert können als Ausfallmodi der Signalwerte betrachtet werden. Diese Ausfallmodi können z. B. aus Sensorausfällen und/oder Kalibrierungsfehlern der Sensoren oder aus Ausfällen von Messwerterfassungsbaugruppen resultieren. Beispielsweise kann aufgrund eines Ausfalls eines Sensors oder einer Messwerterfassungsbaugruppe der Signalwert nach oben oder nach unten ausfallen, d. h. das Ausgangssignal aus der Messwerterfassung nimmt den oberen bzw. unteren Wert des Messbereichs an. Je nach Berechnungsalgorithmus der zugrundeliegenden Funktion des EFB können diese Ausfallmodi der Eingangssignalwerte VAL, falls nicht erkannt bzw. im Rahmen der Projektierung nicht berücksichtigt, zu fehlerhaften Ausgangssignalen des EFB führen.

3.5.2.2 FMEA der Parameter der EFBs

Die Parameter der EFBs, wie z. B. oberer/unterer Grenzwert, Sollwerte, hängen mit dem von der Leittechnik gesteuerten Anlagenprozess zusammen und können veränderbar sein. Veränderbare Parameter ermöglichen z. B. Änderungen der Einstellwerte von LEFU während des Betriebs der Anlage (z. B. Anfahrbedingungen). Die Parameter der EFBs in den LEFU können beispielsweise während des Engineering-Prozesses des Leittechniksystems im Rahmen der Spezifikation der Anwendungssoftware festgelegt werden und während des Betriebs entsprechend der Funktionsspezifikation eingestellt werden. Ausfallmodi von Parametern können sich aus latenten/unerkannten Fehlern während des Engineering-Prozesses des Leittechniksystems (Spezifikation und/oder Modifikation der LEFUs) ergeben. Beispielsweise können bei den EFBs zur Überwachung auf eine Grenzwertunterschreitung bzw. Grenzwertüberschreitung die entsprechenden Grenzwerte im Rahmen des Engineering-Prozesses als Parameter der EFBs spezifiziert werden und im Rahmen des Betriebs eingestellt werden. Ein fehlerhafter Grenzwert kann in diesem Zusammenhang zum Funktionsausfall des EFBs führen und somit zu den Ausfallmodi des EFBs beitragen.

3.5.3 Anwendungsbeispiele

Die entwickelte SFMEA-Methode wurde angewandt, um die Ausfallmodi verschiedener Typen von EFB zu bestimmen. Exemplarisch werden in diesem Abschnitt die erhaltenen Ergebnisse für einen AND-EFB, COMP-EFB, 2. MIN-EFB und 2. MAX-EFB gezeigt und diskutiert. Die Ergebnisse für einen AND-EFB wurden in /MBO 19/ veröffentlicht.

3.5.3.1 AND-EFB

Der hier betrachtete AND-EFB hat zwei Eingangssignale I_1 und I_2 und ein Ausgangssignal O_1, wie in Abb. 3.5 dargestellt. Der AND-EFB besitzt keine Parameter.

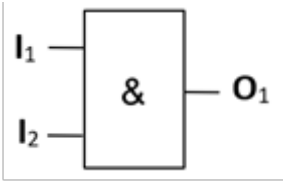

Abb. 3.5 AND-EFB mit zwei Eingangssignalen I_1 und I_2 und einem Ausgangssignal O_1

Der AND-EFB berechnet die logische UND-Verknüpfung der beiden binären Eingangssignale I_1 und I_2 und gibt das Ergebnis als binäres Ausgangssignal O_1 aus. Hinsichtlich der Behandlung von fehlerhaften Zuständen der Eingangssignale leitet der AND-EFB die fehlerhaften Signalzustände der Eingangssignale I_1 und/oder I_2 passiv an sein Ausgangssignal O_1 weiter. Der resultierende Signalzustand von O_1 wird als logische ODER-Verknüpfung der Signalzustände der Eingangssignale I_1 und I_2 ausgewertet. D. h., dass der Signalstatus des Ausgangssignals auf “1“ gesetzt wird, sobald der Status mindestens eines der Eingangssignale I_1 und I_2 auf “1“ gesetzt ist. Daraus folgt, dass mögliche Fehlerzustände des Ausgangssignals O_1 ("fehlerhafter “1"-Zustand oder fehlerhafter “0"-Zustand) den fehlerhaften Signalzuständen der Eingangssignale I_1 und I_2 entsprechen.

Basierend auf den Definitionen in Tab. 3.1 wurden alle möglichen Kombinationen der Signalzustände der Eingangssignale I_1 und I_2 bestimmt und dann entsprechend verknüpft, um ihre Beiträge zu potenziellen Ausfallmodi des Ausgangssignals O_1 zu bewerten. Diese Bewertung berücksichtigt die Behandlung fehlerhafter Zustände der Eingangssignale des AND-EFB.

Die resultierende FMEA-Tabelle für den AND-EFB ist in Tab. 3.2 aufgeführt. Hierbei stellt I1.STAT den Signalstatus von I_1 dar, wie er durch das Leittechniksystem über die Spezifikation der LEFU zugewiesen wird.

I1.VAL-STAT entspricht dem prozessbezogenen Signalstatus-Attribut von I_1, wie in Abschnitt 3.5.2.1 (Tab. 3.1) definiert. Die Attribute STAT und VAL-STAT für das Eingangssignal I_2 sind analog zu I_1 definiert. Aus der Analyse wurden sechs Kombinationen der Signalstatus von I_1 und I_2 identifiziert (in Tab. 3.1 grau hinterlegt), die zum fehlerhaften “1"-Zustand oder zum fehlerhaften “0"-Zustand des Ausgangssignals O_1 führen können.

Diese Kombinationen entsprechen Fällen, bei denen nicht erkannte Ausfälle von Eingangssignalen bzw. als fehlerhaft identifizierte aber korrekte Eingangssignale vorliegen. Dies verdeutlicht, dass insbesondere fehlerhafte Zustände der Eingangssignale die Ausfallmodi der Ausgangssignale der elementaren Funktionsblöcke wesentlich bestimmen. Die Auswirkungen dieser Ausfallkombinationen auf die in Frage stehende LEFU können dann in einem nächsten Schritt mit zusätzlichen Werkzeugen (z. B. Fehlerbaumanalyse statische und dynamische Simulation) evaluiert werden, wobei der Schwerpunkt auf die Analyse der potenziellen Ausbreitungspfade dieser Ausfallmodi in der LEFU gelegt wird.

Tab. 3.2 FMEA-Tabelle eines AND-EFB mit zwei Eingangssignalen I_1 und I_2 und einem Ausgangssignal O_1 (Vgl. Abb. 3.5) /MBO 19/

Fälle	I_1.STAT	I_1.VAL-STAT	Ausgewert. Signalzustand von I_1	I_2.STAT	I_2.VAL-STAT	Ausgewert. Signalzustand von I_2	Ausgewert. Signalzustand v. O_1	Fehlermodeerkennung
1	0	0	korrekter “0”	0	0	korrekter “0”	korrekter “0”	entfällt+
2	0	0	korrekter “0”	0	1	fehlerhafter “0”	fehlerhafter “0”	möglich*
3	0	0	korrekter “0”	1	0	fehlerhafter “1”	fehlerhafter “1”	möglich*
4	0	0	korrekter “0”	1	1	korrekter ”1”	korrekter ”1”	entfällt+
5	0	1	fehlerhafter “0”	0	0	korrekter “0”	fehlerhafter “0”	möglich*
6	0	1	fehlerhafter “0”	0	1	fehlerhafter “0”	fehlerhafter “0”	möglich*
7	0	1	fehlerhafter “0”	1	0	fehlerhafter “1”	korrekter “1”	entfällt+
8	0	1	fehlerhafter “0”	1	1	korrekter ”1”	korrekter ”1”	entfällt+
9	1	0	fehlerhafter “1”	0	0	korrekter “0”	fehlerhafter “1”	möglich*
10	1	0	fehlerhafter “1”	0	1	fehlerhafter “0”	korrekter “1”	entfällt+
11	1	0	fehlerhafter “1”	1	0	fehlerhafter “1”	fehlerhafter “1”	möglich*
12	1	0	fehlerhafter “1”	1	1	korrekter ”1”	korrekter “1”	entfällt+
13	1	1	korrekter “1”	0	0	korrekter “0”	korrekter ”1”	entfällt+
14	1	1	korrekter “1”	0	1	fehlerhafter “0”	korrekter “1”	entfällt+
15	1	1	korrekter “1”	1	0	fehlerhafter “1”	korrekter “1”	entfällt+
16	1	1	korrekter “1”	1	1	korrekter ”1”	korrekter ”1”	entfällt+

Die booleschen Werte I_1.STAT (Status des Eingangssignals I_1) und I_2.STAT (Status des Eingangssignals I_2) sind die vom Leittechniksystem zugewiesenen Signalstatus an die Signale I_1 und I_2. Sie geben an, ob die Signale I_1 und I_2 für die Berechnung des Ausgangssignals O_1 gültig (“0”) oder nicht gültig (“1”) sind.
Die booleschen Werte I_1.VAL-STAT und I_2.VAL-STAT sind die prozessbezogenen Signalstatus der Eingangssignale I_1 und I_2 (siehe Abschnitt 3.5.2.1/Tabelle 3.1). Sie geben an, ob die Signalwerte der Eingangssignale I_1 und I_2 den korrekten Prozesswerten entsprechen („0“) oder nicht entsprechen („1“).
+Fehlermodeerkennung ist nicht erforderlich, denn der Signalstatus des Ausgangssignals O_1 ist korrekt.
*Fehlermodeerkennung ist abhängig von nachfolgenden EFB.

3.5.3.2 COMP-EFB

Der COMP-EFB berechnet die Summe von bis zu drei analogen Eingangssignalen AI1, AI2 und AI3 (sum (AI1, AI2, AI3)) und vergleicht das Ergebnis mit dem Parameter <Lim val>. Abhängig vom Parameter <Type>, der den Typ der Grenzwertüberwachung festlegt, stellt <Lim val> entweder einen oberen Grenzwert oder einen unteren Grenzwert dar.

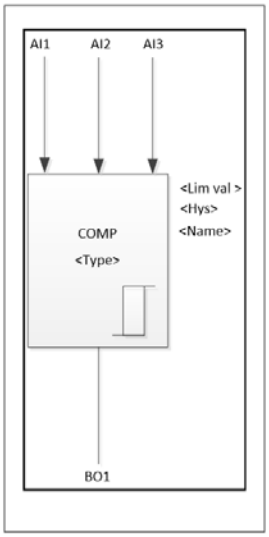

Abb. 3.6 COMP-EFB mit drei analogen Eingangssignalen und einem binären Ausgangsignal

AI1, AI2 und AI3 sind die analogen Eingangssignale und BO1 das binäre Ausgangssignal; <Lim val>, <Hys> und <Type> sind Parameter des COMP-EFB

Überschreitet bzw. unterschreitet (abhängig von <Type>) die berechnete Summe der drei Eingangssignale den eingestellten Grenzwert <Lim val>, setzt der COMP-EFB das binäre Ausgangssignal BO1 auf den booleschen Wert „1“. Das Ausgangssignal BO1 wird auf den booleschen Wert „0“ zurückgesetzt, sobald der Differenzbetrag |sum (AI1, AI2, AI3) - <Lim val>| den Parameter <Hys> überschreitet.

Die optional änderbaren Parameter <Lim val>, <Type>und <Hys> werden im Rahmen der Spezifikation der LEFU festgelegt.

Hinsichtlich der Behandlung von fehlerhaften Zuständen der Eingangssignale leitet der COMP-EFB ähnlich wie der AND-EFB die fehlerhaften Signalzustände der Eingangssignale AI1, AI2 und AI3 passiv an das Ausgangssignal BO1 weiter. Der resultierende Signalstatus von BO1 wird demnach als logische ODER-Verknüpfung der Status der Eingangssignale AI1, AI2, AI3 ausgewertet. Daraus folgt, dass mögliche Fehlerzustände des Ausgangssignals BO1 ("fehlerhafter "1"-Zustand oder fehlerhafter “0"-Zustand) den fehlerhaften Signalzuständen der Eingangssignale AI1, AI2 und AI3 entsprechen.

Bei Vorliegen eines fehlerhaften Signalzustandes wird der Signalwert des Ausgangssignals BO1 nicht mehr aktualisiert und behält seinen letztgültigen Wert.

Die FMEA des COMP-EFB hinsichtlich der Signalzustände der Eingangssignale entspricht im Wesentlichen der FMEA des AND-EFB. Die Anzahl der zu betrachtenden Kombinationen der Eingangssignale zur Bestimmung der Ausfallmodi des COMP-EFB erhöht sich in diesem Fall jedoch auf 64, da der COMP-EFB drei Eingangssignale besitzt. Die FMEA-Tabelle des COM-EFB wurde aus diesem Grund nach vorheriger Validierung der FMEA-Tabelle des AND-EFB mittels dynamischer Simulation (siehe Abschnitt 4.3.2) ermittelt, nachdem diese Simulation am Beispiel des AND-EFB validiert wurde.

Aus der Analyse des COMP-EFB wurden vierzehn Kombinationen von fehlerhaften Signalzuständen der Eingangssignale AI1, AI2 und AI3 identifiziert, die je zu einem fehlerhaften Signalzustand des Ausgangssignals BO1 beitragen können (s. Tab. A.2 im Anhang). Liegt nur ein Signal am Eingang des COMP-EFB an, werden den übrigen Eingängen der Signalwert “0“ zugewiesen.

In Bezug auf mögliche Ausfallmodi des COMP-EFB aufgrund fehlerhafter Parameter <Type>, <Lim val> und <Hys> gelten die Ausführungen im Abschnitt 3.5.2.2 entsprechend. Die genannten Parameter werden während des Engineering-Prozesses des Leittechniksystems im Rahmen der Spezifikation der LEFU festgelegt.

Ausfallmodi der Funktion aufgrund dieser Parameter können sich aus Fehlern während der Spezifikation und/oder Modifikation der Parameter ergeben. Nachfolgend sind einige Beispiele möglicher Ausfallmodi aufgelistet:

- Der Parameter <Type> kann fehlerhaft auf eine Unterschreitung bzw. Überschreitung gesetzt werden.
- Der Grenzwert <Lim val> kann fehlerhaft sein.
- Der Parameter <Hys> kann fehlerhaft sehr hoch gesetzt werden, so dass ein Zurücksetzten des binären Ausgangssignals BO1 unterbleibt.

3.5.3.3 2. MIN-EFB

Der elementare Funktionsblock 2. MIN wählt das zweite Minimum AO1 von bis zu vier analogen Eingangssignalwerten AI1, AI2, AI3 und AI4 aus. Das zweite Minimum wird berechnet, indem die Eingangswerte in aufsteigender Reihenfolge von links nach rechts sortiert werden und dann der zweite Wert von links genommen wird.

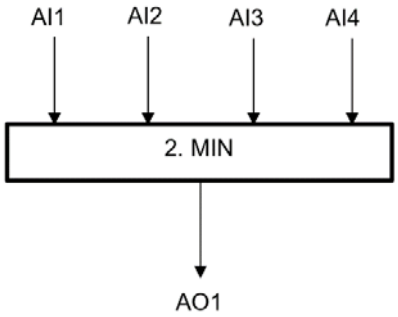

Abb. 3.7 2.MIN-EFB mit vier Eingangssignalen AI1, AI2, AI3 und AI4 und einem Ausgangsignal AO1

Im Gegensatz zu dem AND-EFB und dem COMP-EFB passt der 2. MIN-EFB seine Funktionsweise an die Anzahl der Eingangssignale mit einem fehlerhaften Signalzustand an. Der Funktionsblock 2. MIN filtert analoge Eingangssignale mit einem fehlerhaften Signalzustand heraus. Signale mit einem fehlerhaften Signalzustand werden bei der Berechnung des zweiten Minimums nicht berücksichtigt. Im Falle fehlerhafter Signalzustände aller Eingangssignale gibt der 2. MIN-EFB an seinem Ausgang den Signalwert VAL = 0 und den Signalstatus STAT="1" ERROR zurück. Daraus folgt, dass die Berechnung des Ausgangssignals des 2. MIN-EFB durch den Fehlerstatus der Eingangssignale bestimmt ist.

Die Berechnung des zweiten Minimums ausgehend von Eingangssignalen mit fehlerhaften Signalzuständen gemäß der Definition aus 3.5.2.1 (Tab. 3.1) kann abhängig vom betrachteten Szenario zu einem fehlerhaften Ausgangssignalwert des 2. MIN-EFB führen. Dementsprechend kann die Ausführung der zugrundeliegenden leittechnischen Funktion durch Ausbreitung des fehlerhaften Signalwertes nachfolgende Funktionsblöcke beeinflussen. In der Folge kann es zum Ausfall der LEFU bei Anforderung bzw. zum fehlerhaften Ansprechen der LEFU kommen.

Diese Wirkungsweise des 2. MIN-EFB lässt sich am geeignetsten anhand einer spezifischen leittechnischen Funktion aufzeigen. Aus diesem Grund wurden dynamische Simulationen der in Abb. 3.3 dargestellten Leittechnikfunktion für verschiedene Szenarien durchgeführt. Die hierbei erzielten Ergebnisse (siehe Abschnitt 4.3) belegen, dass es zur fehlerhaften Aktivierung bzw. zum Ausfall der Aktivierung des Ausgangssignals der Leittechnikfunktion kommen kann, und zwar dann, wenn alle 3 Messwerte fehlerhafte Signalzustände haben.

3.5.3.4 2. MAX-EFB

Der elementare Funktionsblock 2. MAX wählt das zweite Maximum AO1 von bis zu vier analogen Eingangssignalwerten AI1, AI2, AI3 und AI4 aus. Das zweite Maximum wird berechnet, indem die Eingangswerte in aufsteigender Reihenfolge von links nach rechts sortiert werden und dann der zweite Wert von rechts genommen wird.

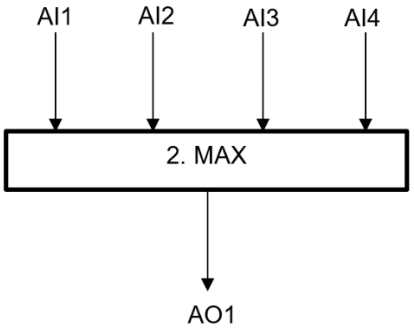

Abb. 3.8 2. MAX-EFB mit vier Eingangssignalen AI1, AI2, AI3 und AI4 und einem Ausgangsignal AO1

Das Verhalten des 2. MAX-EFB bei Vorliegen fehlerhafter Signalzustände am Eingang entspricht dem des 2. MIN-EFB. Die Auswirkung fehlerhafter Signalzustände am Eingang des 2. MAX-EFB wird anhand der in Abb. 3.3 dargestellten Leittechnikfunktion gezeigt (siehe Abschnitt 4.3).

4 Entwicklung eines Konzepts zur Validierung der Analyseergebnisse zu den Auswirkungen von Softwarefehlern in softwarebasierten Leittechniksystemen

4.1 Einleitung

In diesem Kapitel werden die im Rahmen der Bearbeitung des Arbeitspaketes 3 erzielten Ergebnisse vorgestellt.

Ziel war es, ein Konzept zur Validierung der Analyseergebnisse einer durchgeführten Software-FMEA zu den Auswirkungen von latenten Softwarefehlern in softwarebasierten Leittechniksystemen mittels Testanordnungen zu entwickeln.

Das Validierungskonzept baut auf der Anwendung der Software-Fault-Injection-Methode auf. Die Auswirkungen latenter Softwarefehler werden anhand dynamischer Simulationen untersucht. Hierbei werden die Signale durch Eingabe von Fehlern gezielt beeinflusst bzw. fehlerhafte Signalzustände als “Bitflips“ modelliert.

Zur Umsetzung dieses Validierungskonzeptes wurde als Testanordnung zur Nachbildung der Leittechnikfunktionen eines softwarebasierten Leittechniksystems das Analyse und Testsystem (AnTeS) der GRS eingesetzt, welches im Rahmen der Bearbeitung des Arbeitspaketes 3 dieses Vorhabens weiterentwickelt wurde.

Im Abschnitt 4.2 wird das AnTeS der GRS beschrieben. Ergebnisse durchgeführter dynamischer Simulationen zur Validierung des SFMEA-Analysekonzeptes sind im Abschnitt 4.3 dargestellt.

4.2 Das Analyse- und Testsystem der GRS

Das Analyse- und Testsystem der GRS ist eine modular aufgebaute Plattform zur Untersuchung verschiedener Fragestellungen hinsichtlich der Zuverlässigkeit digitaler Leittechnik. Das AnTeS besteht im derzeitigen Entwicklungsstand aus drei Modulen (vgl. Abb. 4.1). Die wichtigsten Eigenschaften der drei Module sind im Einzelnen:

- **Modul 1 - Reales Leittechniksystem:**
  - Basiert auf Komponenten des rechnerbasierten Leittechniksystems Teleperm-XS (TXS) des Herstellers Framatome (ehemals AREVA).
  - Hardware und zugehörige Engineering-Software zur Generierung von Anwendungssoftware
  - Validierung von Anwendungssoftware mit der Simulationssoftware SIVAT (Simulation and Validation Tool).
  - Unterschiedliche Leittechnik-Architekturen realisierbar.
  - Unterschiedliche Netzwerktopologien realisierbar.
  - Typische leittechnische Funktionen der Prozessautomatisierung und für Sicherheitsaufgaben umsetzbar.
  - Beeinflussung von Signalen und Eingabe von Fehlern möglich (Fault-Injection).
  - Online-Überwachung sämtlicher Signale innerhalb der Leittechnik möglich.

- **Modul 2 - Simuliertes Leittechniksystem**:
  - Nachbildung der jeweils mit dem realen Leittechniksystem umgesetzten Leittechnikfunktionen mit der Software MATLAB/Simulink[18] oder durch eigenständige Leittechniksystem-Simulationen.
  - Beeinflussung von Signalen und Eingabe von Fehlern möglich (Fault-Injection).
  - Automatisierte FMEA für alle Eingangssignalkombinationen und zuvor festgelegte Ausfallarten innerhalb des Leittechniksystems möglich.
  - Durchführung von Monte-Carlo-Simulationen (statistische Fehlereinspeisung und Ergebnisaufzeichnung für lange simulierte Zeiträume) möglich.
- **Modul 3 - Generierung und Überwachung von Ein- und Ausgangssignalen:**
  - Für die Module 1 und 2 über zusätzliche Hardware (Raspberry Pi, Arduino Due).
  - Software entwickelt mit Python (flexibel änderbar).
  - Manuelle oder automatische (per Skript-Steuerung) Vorgabe von Eingangssignalen und Aufzeichnung der durch die Leittechnik-Systeme (Modul 1 und 2) generierten Ausgangssignale.
  - Simulation von verfahrenstechnischen Systemen möglich (z. B. generisches Brennelementlagerbecken).

Mit dem realen Leittechniksystem (Modul 1) können unterschiedliche Leittechnik-Architekturen und Netzwerktopologien redundant aufgebauter softwarebasierter Leittechniksysteme verwirklicht werden. Derzeit verfügt das Modul 1 über vier redundante Verarbeitungsrechner in zwei Leittechnikschränken (vgl. Abb. 4.2). Die vier Verarbeitungsrechner umfassen jeweils Ein- und Ausgabebaugruppen für digitale und analoge Signale sowie Prozessormodule.

[18] Simulink ist eine blockorientierte Programmier- und Entwicklungsumgebung der Software MATLAB zur Simulation dynamischer Systeme. In Simulink werden Systemmodelle mittels grafischer Funktionsblöcke erstellt.

Die Verarbeitungsrechner sind untereinander über Netzwerkverbindungen für den Austausch zwischen den vier Redundanten vernetzt. Zusätzlich ist über eine Ethernetverbindung ein Service-Rechner („Service Unit“) für die Programmierung und Überwachung der Leittechnikfunktionen mittels Engineering-Software angebunden.

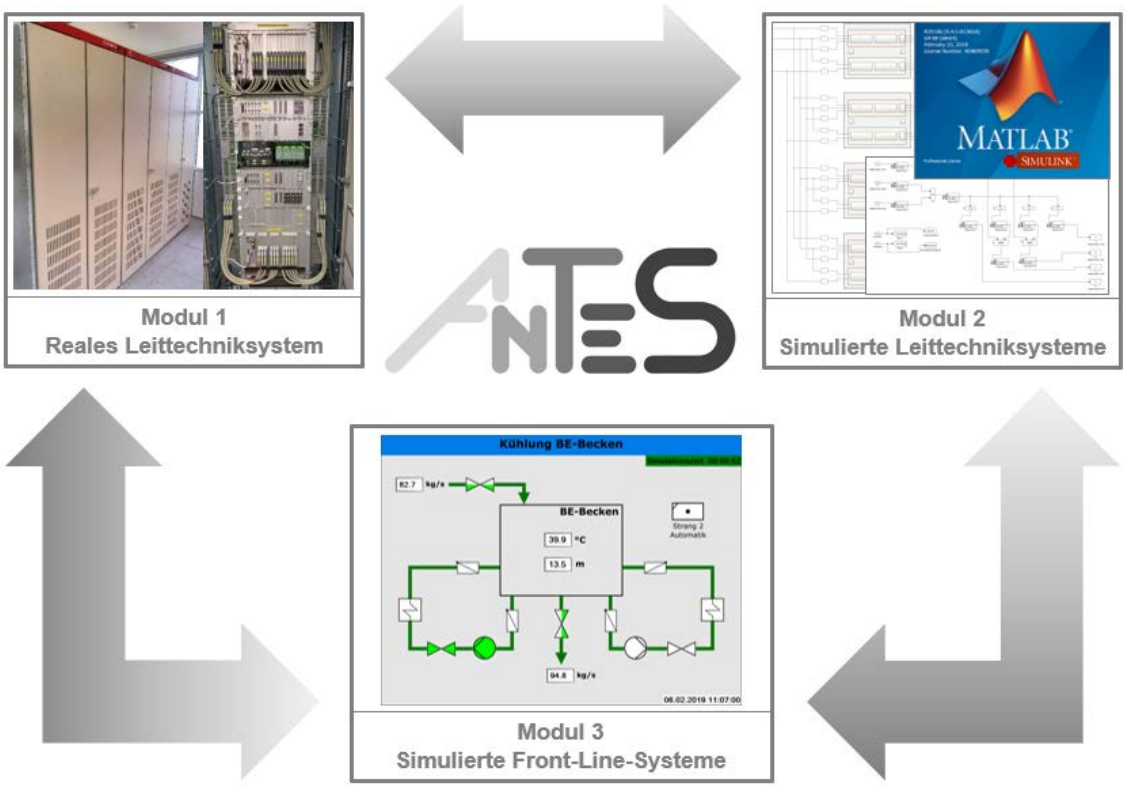

Abb. 4.1 Modular aufgebautes Analyse- und Testsystem der GRS zur Untersuchung digitaler Leittechniksysteme

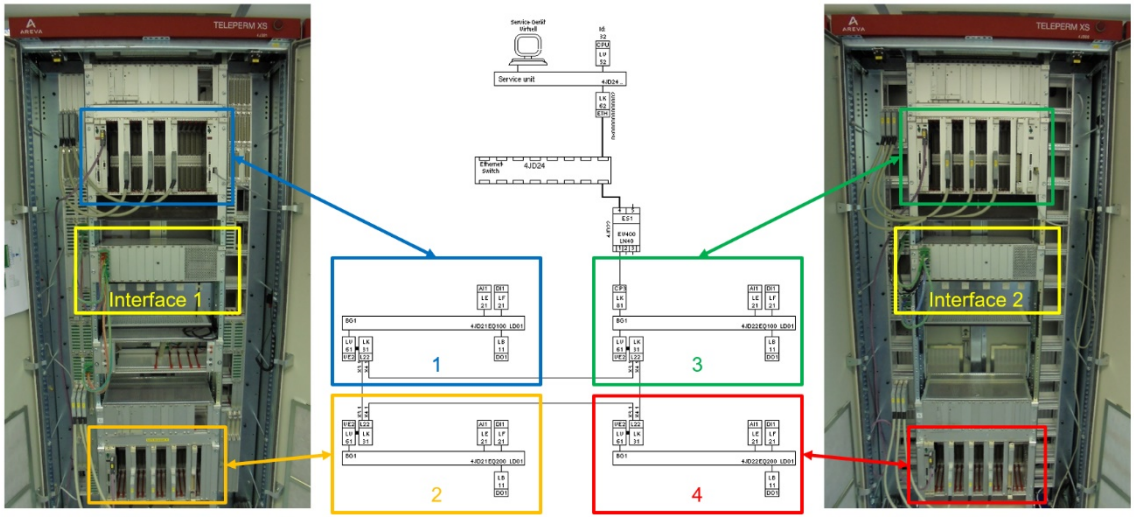

Abb. 4.2 Grundsätzlicher Aufbau des Moduls 1 – TXS-Netzwerkplan (Mitte) und Frontansicht der TXS-Schränke (links, rechts)

Das Modul 2 von AnTeS erlaubt es, Architekturen und Topologien in einer simulierten Umgebung zu analysieren. Für diesen Zweck sind die einzelnen elementaren Funktionsblockbausteine des TXS-Systems nachgebildet worden, diese können in der simulierten Umgebung in derselben Weise wie im realen System (Modul 1) zu Leittechnikfunktionen kombiniert werden.

Sowohl in der realen als auch in der simulierten Leittechnik können beliebige Leittechnikfunktionen erstellt werden. Im TXS-System steht hierfür die entsprechende Engineering-Umgebung von Framatome (SPACE – Specification And Coding Environment) zur Verfügung. Die Erstellung von leittechnischen Funktionen in der simulierten Umgebung erfolgt unmittelbar mit der Software Matlab/Simulink. In beiden Modulen (Modul 1 und Modul 2) können gezielt Fehler injiziert und deren Auswirkungen analysiert werden (Fault-Injection). Diese Möglichkeit, Signale gezielt durch Fault-Injection zu beeinflussen, wurde im Rahmen der Validierung des entwickelten SFMEA-Konzeptes entwickelt und verwendet.

In der simulierten Umgebung ist diese Möglichkeit jedoch umfangreicher, da hier Signale uneingeschränkt beeinflusst und auch Hardware-Fehler berücksichtigt werden können, die im realen Leittechniksystem nicht ohne Beschädigung des Systems möglich wären.

Mit Hilfe des Moduls 3 können für Analysen u. a. die Eingangssignale vorgegeben und die entsprechenden Ausgangssignale aufgezeichnet werden. Die entsprechende Software ist eine Eigenentwicklung der GRS. Da die Software im Wesentlichen mittels der Programmiersprache Python entwickelt wurde, sind hier Veränderungen oder Anpassungen flexibel möglich.

Neben der bloßen Möglichkeit gezielt Eingangssignale (in die Leittechnik) vorzugeben und die Ausgangssignale aufzuzeichnen, umfasst das Modul 3 des AnTeS auch noch Simulationen verfahrenstechnischer Systeme (wie z. B. eine Simulation eines generischen Brennelementlagerbeckens), welche mit dem softwarebasierten Leittechniksystem gekoppelt werden können.

Das Modul 3 kann sowohl mit der realen Leittechnik (Modul 1) als auch dem simulierten Leittechniksystem (Modul 2) gekoppelt werden. Für die Anbindung an die reale Leittechnik stehen hierfür zwei von der GRS entwickelte Interfaces (s. Abb. 4.2) zur Verfügung.

4.3 Dynamische Simulationen zur Validierung des SFMEA-Analysekonzeptes

4.3.1 Prinzip der dynamischen Simulationen

Mit der im Rahmen der Bearbeitung des Arbeitspaketes 2 entwickelten SFMEA-Methode (siehe Abschnitt 3.5) können die Ausfallmodi elementarer Funktionsblöcke zur Generierung der Anwendungssoftware softwarebasierter Leitechniksysteme systematisch ermittelt werden. Für elementare Funktionsblöcke mit einer großen Anzahl von Eingangssignalen wird die Anwendung der SFMEA-Methode per Hand sehr zeitaufwändig bis unmöglich. Daher liegt es nahe, die entwickelte SFMEA-Methode zu automatisieren.

Das Prinzip der Automatisierung besteht darin, die Fehlerstatus der Eingangssignale von elementaren Funktionsblöcken mittels Fault-Injection gezielt zu modifizieren. Alle möglichen Kombinationen der Fehlerstatus der Eingangssignale werden ermittelt und deren Auswirkung auf den Fehlerstatus der zugehörigen Ausgangssignale des elementaren Funktionsblockes ausgewertet. Hierfür wurden dynamische Simulationen mit dem Modul 1 (TXS/SIVAT) und dem Modul 2 (Matlab/Simulink) des AnTeS der GRS durchgeführt.

4.3.2 Automatisierte SFMEA eines AND-EFB

Am Beispiel eines AND-Funktionsblocks mit zwei Eingangssignalen und einem Ausgangssignal wird die Vorgehensweise zur Automatisierung der entwickelten SFMEA-Methode erläutert.

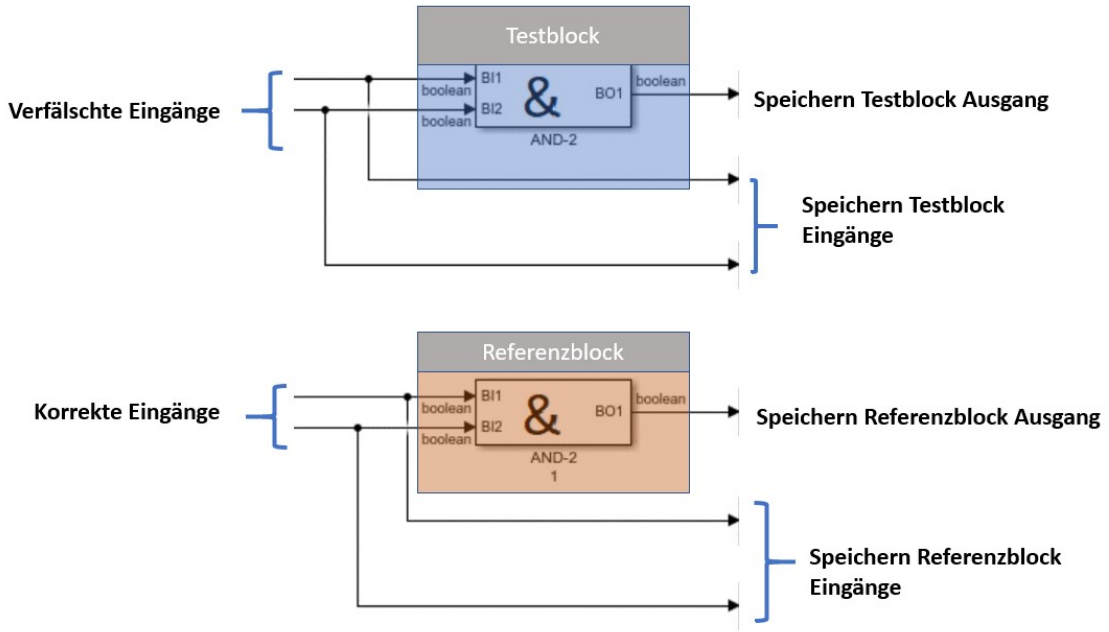

Abb. 4.3 Matlab/Simulink-Modell zur Bestimmung der Ausfallmodi des Ausgangssignals eines AND-EFB

Abb. 4.3 zeigt einen Ausschnitt des zugrunde gelegten Modells zur dynamischen Simulation des AND-EFB in Matlab/Simulink (Modul 2 des AnTeS). Hierzu werden zwei AND-EFBs, deren Funktion, die eines AND-EFB aus dem Leittechniksystem TXS nachbilden, angesteuert. Der obere Block („Testblock“) stellt den zu untersuchenden AND-EFB dar, dessen Ausfallmodi ermittelt werden sollen. Der untere Block („Referenzblock“) dient

zum Vergleich der Ergebnisse des Testblocks mit denen bei korrekten/“echten“ Eingangssignalen am Referenzblock gemäß der Definition aus dem Abschnitt 3.5.2.1. Um eine vollständige Abdeckung der zu betrachtenden Fälle zu gewährleisten, werden nun alle möglichen Eingangskombinationen der insgesamt vier Fehlerstatus (Referenz- und Testblock) angelegt, d. h. die Berechnung des Ausgangssignals des AND-EFB wird sowohl für jede mögliche Eingangskombination der Fehlerstatus der Eingangssignale als auch für jede mögliche Verfälschung der Fehlerstatus der Eingangssignale durchgeführt. Die Eingänge des Referenzblocks sind hierbei gemäß der Definition aus dem Abschnitt 3.5.2.1 die “korrekten“/„echten“ Eingangssignale. Die Eingänge des Testblocks werden als potenziell aufgrund latenter Fehler verfälschte Eingangssignale betrachtet. Die Fehlerstatus der Ein- und Ausgangssignale von Referenz- und Testblock werden nun mittels eines hierfür entwickelten Sortieralgorithmus miteinander verglichen und ihnen darauf basierend der entsprechende Fehlerstatus gemäß der Definition aus dem Abschnitt 3.5.2.1 („korrekter 1“, “fehlerhafter 0“, “fehlerhafter 1“, “korrekter 0“) zugewiesen.

Die Ergebnisse der durchgeführten Simulationen sind gemäß der Bezeichnung aus Abschnitt 3.5.3.1 in Tabelle 4.1 dargestellt. I1_STAT bezeichnet hierbei den Fehlerstatus des ersten Eingangssignals des Testblockes, I1_VAL_STAT den entsprechenden Fehlerstatus am Eingang des Referenzblockes. Wie aus Tab. 4.1 zu entnehmen ist, stimmen die Ergebnisse mit denen aus der Evaluierung der Ausfallmodi des AND-EFB per Hand (siehe Abschnitt 3.5.3.1) überein.

Tab. 4.1 Ergebnisse der automatischen Evaluierung der Ausfallmodi eines AND-EFB mit dem Modul 2 von AnTeS (Matlab/Simulink)

I1_STAT	I1_VAL_STAT	Ausgewerteter Signal-status von I1	I2_STAT	I2_VAL_STAT	Ausgewerteter Signal-status von I2	Ausgewerteter Signal-status von O1	Fehlermode-erkennung
0	0	korrekter 0	0	0	korrekter 0	korrekter 0	entfällt
0	0	korrekter 0	0	1	fehlerhafter 0	fehlerhafter 0	möglich
0	0	korrekter 0	1	0	fehlerhafter 1	fehlerhafter 1	möglich
0	0	korrekter 0	1	1	korrekter 1	korrekter1	entfällt
0	1	fehlerhafter 0	0	0	korrekter 0	fehlerhafter 0	möglich
0	1	fehlerhafter 0	0	1	fehlerhafter 0	fehlerhafter 0	möglich
0	1	fehlerhafter 0	1	0	fehlerhafter 1	korrekter 1	entfällt
0	1	fehlerhafter 0	1	1	korrekter 1	korrekter 1	entfällt
1	0	fehlerhafter 1	0	0	korrekter 0	fehlerhafter 1	möglich
1	0	fehlerhafter 1	0	1	fehlerhafter 0	korrekter 1	entfällt
1	0	fehlerhafter 1	1	0	fehlerhafter 1	fehlerhafter 1	möglich
1	0	fehlerhafter 1	1	1	korrekter 1	korrekter 1	entfällt
1	1	korrekter 1	0	0	korrekter 0	korrekter 1	entfällt
1	1	korrekter 1	0	1	fehlerhafter 0	korrekter 1	entfällt
1	1	korrekter 1	1	0	fehlerhafter 1	korrekter 1	entfällt
1	1	korrekter 1	1	1	korrekter 1	korrekter 1	entfällt

Neben der Nachbildung in MATLAB/Simulink wurde die automatische Bestimmung der Ausfallmodi des AND-2 Blockes ebenfalls unter Nutzung der Engineering-Tools SPACE und SIVAT des Leittechniksystems TXS (Moduls 1 von ANTES) durchgeführt.

SPACE ist eine Entwicklungsumgebung, in der mittels einer blockbasierten grafischen Oberfläche leittechnische Funktionspläne für das Leittechniksystems TXS entworfen werden können. Ausgehend von der grafischen Darstellung der Leittechnikfunktionen wird dann der ausführbare Code der Anwendungssoftware generiert, der anschließend an die Verarbeitungsmodule des TXS Systems übertragen wird.

SIVAT ist ein Simulationsprogramm, mit dem in der SPACE-Entwicklungsumgebung generierte leittechnische Funktionspläne getestet und ausgeführt werden können, ohne diese zuvor auf das TXS System zu übertragen. Dies ermöglicht eine Überprüfung der implementierten leittechnischen Funktionen vor der Übertragung des ausführbaren Codes der Anwendungssoftware an das TXS System.

In der nachfolgenden Abb. 4.4 ist ein Ausschnitt des mit SPACE/SIVAT modellierten Funktionsblockdiagramms zur automatischen Bestimmung der Ausfallmodi des AND-EFB enthalten in dem zwei AND-EFB (Referenzblock und Testblock wie im Matlab/Simulink-Modell) sowie die anliegenden Signale dargestellt sind.

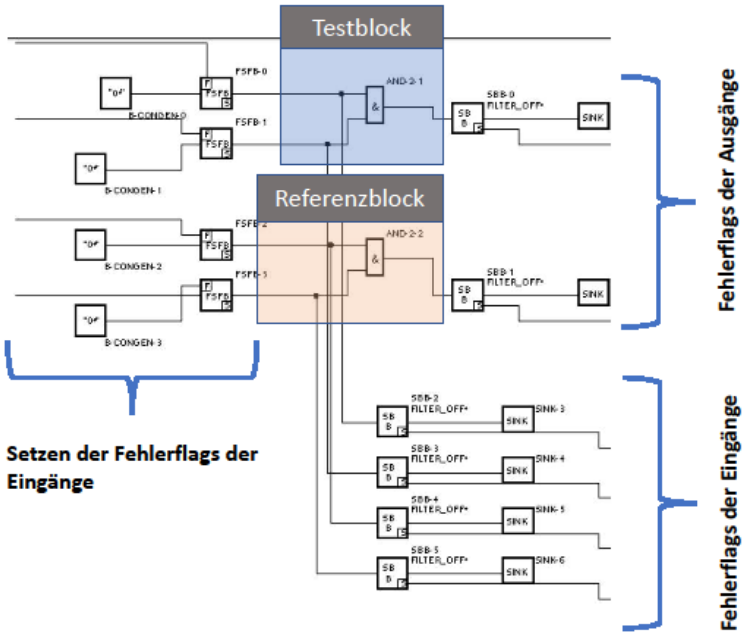

Abb. 4.4 SPACE/SIVAT- Modell zur automatischen Evaluierung der Ausfallmodi des Ausgangssignals eines AND-EFB (Funktionsplanausschnitt)

Zur Generierung der Eingangssignale der zwei AND-EFB (Referenz- und Testblock) wurden die B-CONGEN und FSFB-Funktionsblöcke in SPACE verwendet (Abb. 4.4).

Die FSFB-Funktionsblöcke dienen dem Setzen/Modifizieren der Fehlerflags der Eingangssignale. Mit den B-CONGEN-Funktionsblöcken werden die Signalwerte der Eingangssignale erzeugt. Im vorliegenden Modell wurden Eingangssignale mit dem booleschen Wert “0“ generiert, da der Signalwert für die Verarbeitung der Fehlerflags im AND-EFB nicht relevant ist.

Zur Weiterleitung der Fehlerflags für die anschließende Auswertung der Simulationsergebnisse wurden SB-Funktionsblöcke in SPACE eingesetzt, die es ermöglichen Signale in Signalwert und zugehöriges Fehlerflag aufzuteilen.

Die Simulation erfolgte entsprechend der Simulation mit dem MATLAB/Simulink-Modell, in dem die Berechnung des Ausgangssignals der AND-EFB für alle möglichen Kombinationen der insgesamt vier Fehlerstatus (Referenz- und Testblock) der Eingangssignale durchgeführt wurde.

Die Ergebnisse dieser mit SPACE/SIVAT durchgeführten Simulationen zur Bestimmung der Ausfallmodi des AND-EFB stimmen, wie Tab 4.2 zu entnehmen ist, mit denen aus Matlab/Simulink überein.

Tab. 4.2 Ergebnisse der automatischen Evaluierung der Ausfallmodi eines AND-EFB mit dem Modul 1 von ANTES (SPACE/SIVAT)

I1_STAT	I1_VAL_STAT	Ausgewerteter Signalstatus von I1	I2_STAT	I2_VAL_STAT	Ausgewerteter Signalstatus von I2	Ausgewerteter Signalstatus von O1	Fehlermode-erkennung
0	0	korrekter 0	0	0	korrekter 0	korrekter 0	entfällt
0	0	korrekter 0	0	1	fehlerhafter 0	fehlerhafter 0	möglich
0	0	korrekter 0	1	0	fehlerhafter 1	fehlerhafter 1	möglich
0	0	korrekter 0	1	1	korrekter 1	korrekter 1	entfällt
0	1	fehlerhafter 0	0	0	korrekter 0	fehlerhafter 0	möglich
0	1	fehlerhafter 0	0	1	fehlerhafter 0	fehlerhafter 0	möglich
0	1	fehlerhafter 0	1	0	fehlerhafter 1	korrekter 1	entfällt
0	1	fehlerhafter 0	1	1	korrekter 1	korrekter 1	entfällt
1	0	fehlerhafter 1	0	0	korrekter 0	fehlerhafter 1	möglich
1	0	fehlerhafter 1	0	1	fehlerhafter 0	korrekter 1	entfällt
1	0	fehlerhafter 1	1	0	fehlerhafter 1	fehlerhafter 1	möglich
1	0	fehlerhafter 1	1	1	korrekter 1	korrekter 1	entfällt
1	1	korrekter 1	0	0	korrekter 0	korrekter 1	entfällt
1	1	korrekter 1	0	1	fehlerhafter 0	korrekter 1	entfällt
1	1	korrekter 1	1	0	fehlerhafter 1	korrekter 1	entfällt
1	1	korrekter 1	1	1	korrekter 1	korrekter 1	entfällt

Der Modellierungsaufwand mit den Engineering-Tools SPACE/SIVAT ist jedoch erheblich höher im Vergleich zum Einsatz von Matlab/Simulink.

Im Anhang A finden sich Ergebnisse durchgeführter Simulationen zur automatischen Bestimmung der Ausfallmodi der Eingangssignale eines EFB und deren Auswirkungen auf seinen Ausgang für zwei weitere Bausteine: Ein OR-EFB mit zwei Eingangssignalen und einem Ausgangssignal sowie ein LIMITER-EFB mit drei Eingangssignalen und einem Ausgangssignal.

4.3.3 SFMEA einer Leittechnikfunktion bestehend aus mehreren EFB

In diesem Abschnitt wird ein Beispiel zur automatischen Evaluierung des Verhaltens einer Leittechnikfunktion bei Vorliegen fehlerhafter Signalzustände mittels dynamischer Simulationen gezeigt.

4.3.3.1 Modellierung der Leittechnikfunktion

Für die Untersuchungen wurde eine Leittechnikfunktion zugrunde gelegt, die der Logik zur Auslösung des Durchdringungsabschlusses (DDA) der Frischdampfleitungen in einer SWR-Anlage entlehnt ist. Diese Leittechnikfunktion wurde ausgewählt, da zur Realisierung ihrer Auslöselogik EFBs mit den typischen Behandlungsarten fehlerhafter Signalzustände (siehe Abschnitt 3.5.2) verwendet werden. Hierdurch wird ermöglicht, die Auswirkungen verschiedener Behandlungsarten fehlerhafter Signalzustände auf die Ausbreitung von Fehlern in einer Leittechnikfunktion zu analysieren.

Das entsprechende Funktionsblockdiagramm der DDA-Funktion ist in Abb. 4.5 dargestellt.

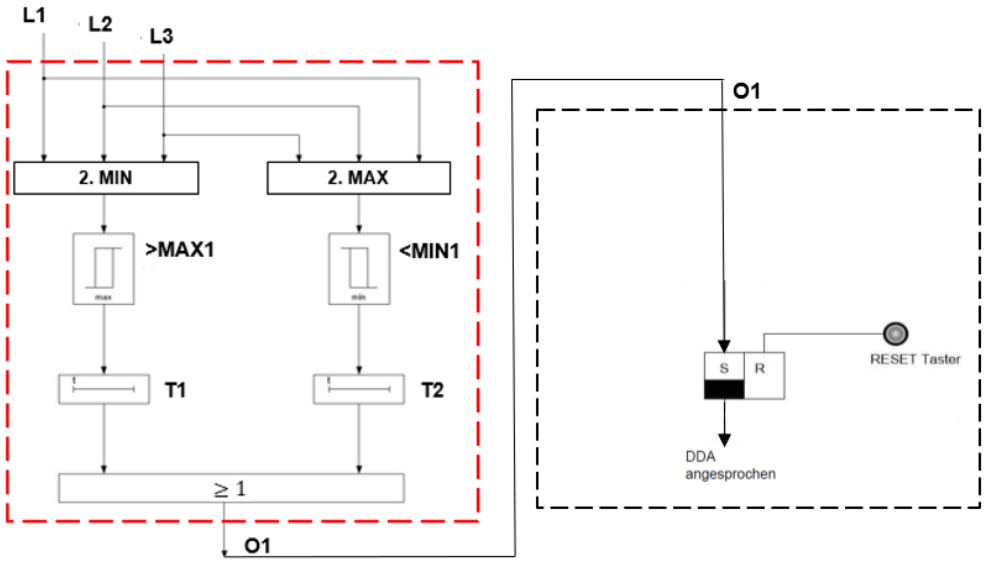

Abb. 4.5 Darstellung der DDA-Funktion: Berechnung (rote Umrandung) und Speicherung (schwarze Umrandung) des DDA-Auslösesignals (O1)

Zur Auslösung des Durchdringungsabschlusses werden gemäß der dargestellten Logik in Abb. 4.5 drei redundante Füllstandmesswerte des Reaktordruckbehälters (L1, L2, L3) in zwei Strängen verarbeitet. In einem Strang wird die Füllstandsmessung des Reaktordruckbehälters auf ein Überschreiten, im anderen Strang auf ein Unterschreiten jeweils eines definierten Grenzwertes überprüft. Dafür wird jeweils das 2. Minimum (2. MIN) bzw. das 2. Maximum (2. MAX) der drei Füllstandmesswerte gebildet und anschließend ein Grenzwertvergleich mit Hysterese (COMP-EFB „>MAX1“, COMP-EFB „<MIN1“) durchgeführt.

Falls einer der beiden Grenzwerte (oberer Grenzwert/unterer Grenzwert) für ein vorgegebenes Zeitintervall über- bzw. unterschritten wird, wird über eine ODER-Verknüpfung (OR-EFB) das DDA-Signal O1 ausgelöst. Die Überwachung der Zeitintervalle für die definierten Grenzwerte wird hierbei mit Einschaltverzögerungsbausteinen (VZ-EFB „T1“, VZ-EFB „T2“) realisiert. Das DDA-Auslösesignal O1 wird in einem weiteren Schritt nach seiner Auslösung einem Speicherbaustein übergeben und dort gespeichert. Ein Rücksetzen des gespeicherten DDA-Auslösesignals ist erst möglich, wenn der Reaktorfüllstand im Reaktordruckbehälter die DDA-Ansprechgrenzwerte unterschritten bzw. überschritten hat. Das Rücksetzen des DDA-Speichersignals erfolgt über einen Taster (RESET-Taster).

Die bei der Modellierung der DDA-Funktion zugrunde gelegten Werte für den Reaktordruckbehälterfüllstand, Ansprechgrenzwerte und Verzögerungszeiten zur Auslösung des DDA sind in aufgeführt.

Tab. 4.3 Verwendete Werte und Parameter in der Simulation der DDA-Funktion

Werte für den Reaktordruckbehälterfüllstand, Ansprechgrenzwerte und Verzögerungszeiten zur Auslösung des DDA[19]

Normalfüllstand L-SOLL Messbereich	13 m 9-16 m
Oberer Grenzwert L-MAX des RDB-Füllstandes	14,92 m
Unterer Grenzwert L-MIN des RDB-Füllstandes	12 m
Hysterese der COMP-EFB	0,1 m
Einschaltverzögerung d. VZ-EFB bei Überschreiten d. oberen Grenzwertes L-MAX	12 s
Einschaltverzögerung d. VZ-EFB bei Unterschreiten d. unteren Grenzwertes L-MIN	4 s

4.3.3.2 Festlegungen für die Simulationen

Mit der Simulation der DDA-Funktion sollen insbesondere zwei weitere Aspekte demonstriert werden. Zum einen die Untersuchung eines zeitabhängigen Verlaufs (v. a. durch die Einschaltverzögerungen) und zum anderen die Untersuchung einer Leittechnikfunktion, bei der die Fehlerzustände der Eingangssignale (hier L1, L2, L3) die Berechnung

[19] Die Wasserstände entsprechen hier nicht zwingend realistischen Werten irgendeiner Beispielanwendung, da für die Auswertung hier lediglich eine Abweichung nach oben/unten notwendig ist.

des Ausgangssignals (hier O1) der Leittechnikfunktion beeinflussen. Diese Beeinflussung geschieht in der vorliegenden DDA-Funktion dadurch, dass bei der Berechnung des 2. Minimums (2. MIN-EFB) und des 2. Maximums (2. MAX-EFB) Signalwerte mit gesetztem Fehlerzustand für die Berechnung nicht berücksichtigt werden, solange noch mindestens ein valides/gültiges Signal vorliegt.

Liegt nur ein valides Signal am Eingang des 2. MIN-EFB bzw. des 2. MAX-EFB wird dieses ausgegeben.

Um diesen Einfluss des 2. MIN-EFB und des 2. MAX-EFB auf das Verhalten einer Leittechnikfunktion aufzuzeigen, wurden für die Untersuchung des Verhaltens der DDA-Funktion bei Vorliegen von fehlerhaften Signalzuständen zwei Szenarien mit den gemäß Abb. 4.6 und Abb. 4.7 vordefinierten Verläufen für den Füllstand im Reaktordruckbehälter betrachtet. Prinzipiell wäre es auch möglich, den Füllstandverlauf im RDB durch eine verfahrenstechnische Simulation unter Verwendung des Moduls 3 von AnTeS nachzubilden und anschließend durch Koppelung derart simulierter Füllstandverläufe mit dem Modul 2 von AnTeS das Verhalten der LEFU bei Vorliegen von fehlerhaften Signalzuständen zu untersuchen.

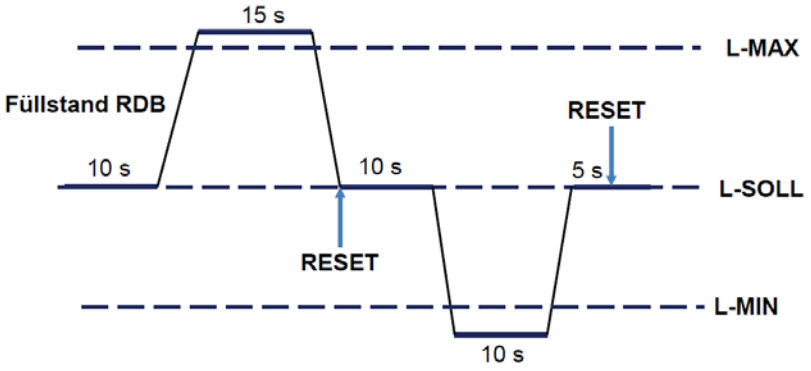

Abb. 4.6 Szenario A: Simulierter Verlauf des Füllstandes im RDB mit Angabe der Rücksetzpunkte (RESET) des DDA-Speichersignals

L-SOLL kennzeichnet den RDB-Sollfüllstand. L-MAX bezeichnet den oberen Füllstandgrenzwert, L-MIN den unteren Füllstandgrenzwert zur Auslösung des DDA-Signals (s. Abb. 4.5)

Beim ersten Szenario (Szenario A) wird der Füllstand im RDB ausgehend vom Sollfüllstand L-SOLL zunächst bis oberhalb des oberen Füllstandgrenzwertes L-MAX zur Auslösung des DDA-Signals erhöht. Nach Ablauf eines Zeitintervalls von 15 s wird der RDB-Füllstand auf den Sollwert L-SOLL zurückgeführt. Nach Erreichen des Sollfüllstandes wird der DDA-Speicher zurückgesetzt. Darauf folgt ein Absenken des RDB-Füllstandes

bis unterhalb des unteren Füllstandgrenzwertes L-MIN zur Auslösung des DDA-Signals. Nach Ablauf eines Zeitintervalls von 10 s wird der RDB-Füllstand erneut auf den Sollwert L-SOLL zurückgeführt und anschließend der DDA-Speicher zurückgesetzt. Die Verweildauer des RDB-Füllstandes oberhalb des oberen Füllstandgrenzwertes bzw. unterhalb des unteren Füllstandgrenzwertes wurde entsprechend gewählt, um die Funktion der Einschaltverzögerungsbausteine im Rahmen der Untersuchungen zu überprüfen. Durch das Überschreiten des oberen Grenzwertes und anschließende Unterschreiten des unteren Grenzwertes wird ermöglicht, beide Auslösestränge der DDA-Funktion zu überprüfen.

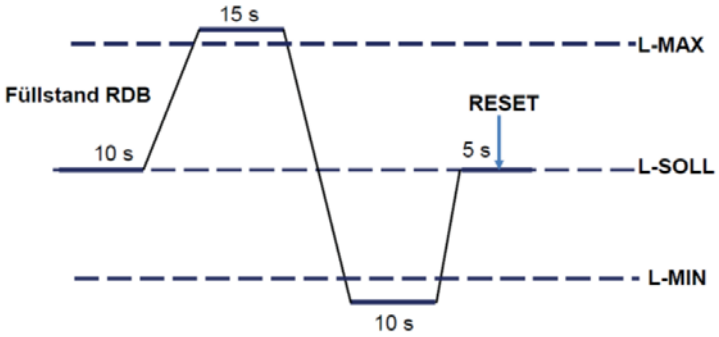

Abb. 4.7 Szenario B: Simulierter Verlauf des RDB-Füllstandes mit Kennzeichnung der Rücksetzpunkte (RESET) des DDA-Speichersignals

L-SOLL kennzeichnet den RDB-Sollfüllstand. L-MAX bezeichnet den oberen Füllstandgrenzwert, L-MIN den unteren Füllstandgrenzwert zur Auslösung des DDA-Signals (s. Abb. 4.5)

Der simulierte Füllstandverlauf im zweiten Szenario (Szenario B) entspricht im Wesentlichen dem simulierten Verlauf im ersten Szenario (Szenario A). Der Unterschied zwischen den beiden simulierten Verläufen besteht darin, dass beim zweiten Szenario der Übergang zwischen Überschreiten und Unterschreiten der DDA-Füllstandgrenzwerte zur Auslösung des DDA-Signals unmittelbar geschieht, ohne zwischenzeitliches Rücksetzen des DDA-Speichersignals. Somit kann überprüft werden, ob der Speicherbaustein in seiner Vorzugslage (gesetzt) verharrt.

Der Einfluss der Fehlerzustände der Eingangssignale L1, L2 und L3 auf die Ausführung der DDA-Funktion wird für folgende Fälle untersucht:

- Ausfall der Signale L1, L2, L3 nach Messbereichsende OBEN (HIGH) (Fall 1) und
- Ausfall der Signale L1, L2, L3 nach Messbereichsende UNTEN (LOW) (Fall 2).

Hierbei werden folgende Variationen betrachtet:

- Ausfall eines der drei Signale nach OBEN (Fall 1a): z. B. L1 fällt auf > L-MAX; L2 und L3 gemäß Verlauf in Abb. 4.6 und 4.7.
- Ausfall von zwei Signalen nach OBEN (Fall 1b): z. B. L1 fällt auf > L-MAX;
- L2 fällt auf > L-MAX; L3 gemäß Verlauf in Abb. 4.6 und 4.7.
- Ausfall aller drei Signale nach OBEN (Fall 1c): L1, L2, L3 fallen auf > L-MAX.
- Ausfall eines der drei Signale nach UNTEN (Fall 2a): z. B. L1 fällt auf < L-MIN; L2 und L3 gemäß Verlauf in Abb. 4.6 und 4.7.
- Ausfall von zwei Signalen nach UNTEN (Fall 2b): z. B. L1 fällt auf < L-MIN;
- L2 fällt auf < L-MIN; L3 gemäß Verlauf in Abb. 4.6 und 4.7
- Ausfall aller drei Signale nach UNTEN (Fall 2c): L1, L2, L3 fallen auf < L-MIN.

Hinsichtlich der Fehlerzustände der Eingangssignale L1, L2 und L3 werden im Rahmen der Untersuchungen folgende Fälle betrachtet:

- Fälle, bei denen der Ausfall eines Signals auf das obere bzw. untere Messbereichsende vom Leittechniksystem erkannt wird. Der Fehlerstatus des betroffenen Eingangssignals wird entsprechend gesetzt. Das betroffene Signal wird für die Berechnung der DDA-Funktion nicht berücksichtigt.
- Fälle, bei denen der Ausfall eines Signals auf das obere bzw. untere Messbereichsende vom Leittechniksystem nicht erkannt wird. Der Fehlerstatus des betroffenen Signals wird nicht gesetzt. Folglich bleibt das betroffene Signal für die Berechnung der DDA-Funktion fehlerhaft gültig.
- Fälle, bei denen ein korrektes Signal fälschlicherweise einem Fehlerstatus zugewiesen wird. Das betroffene Signal wird für die Berechnung der DDA-Funktion fehlerhaft nicht berücksichtigt.
- Fälle, bei denen ein korrektes Signal richtigerweise keinem Fehlerstatus zugewiesen wird. Das betroffene Signal wird für die Berechnung der DDA-Funktion berücksichtigt.

Ausgefallene Signale werden im Rahmen der Simulationen mit der Signalkennung „F“, korrekte Signale mit der Signalkennung „OK“ gekennzeichnet. Der Fehlerstatus eines

Signals wird durch die Fehlerzustandskennung EF gekennzeichnet. Die Fehlerzustandskennung EF wird dem booleschen Wert “1“ für ein fehlerhaftes bzw. “0“ für ein valides/gültiges Signal zugewiesen. Die entsprechende Zuordnung der Fehlerzustandskennung EF zu den Signalkennungen “F“ und “OK“ ist in Tab. 4.4 angegeben.

Tab. 4.4 Zuordnung der Fehlerzustandskennungen zu den Signalkennungen im Rahmen der Simulationen

	Signalkennung „F“	**Signalkennung „OK“**
Fehlerzustandskennung EF= „0“	Ausgefallenes Signal nicht erkannt	Korrektes Signal erkannt
Fehlerzustandskennung EF= „1“	Ausgefallenes Signal erkannt	Korrektes Signal nicht erkannt

Die Fehlerzustandskennungen der Signale L1, L2 und L3 werden im Rahmen der Simulationen gemäß obigen Festlegungen variiert. Fehlerhafte Signalzustände der Eingangssignale L1, L2 und L3 liegen dann vor, wenn mindestens ein ausgefallenes Signal nicht erkannt bzw. mindestens ein korrektes Signal nicht erkannt wird.

Es wird für die Simulationen zudem postuliert, dass mindestens ein Signal ausfällt und dass die Signale bei mehrfachem Ausfall in die gleiche Richtung ausfallen, d. h., dass alle ausgefallenen Signale auf das obere bzw. untere Messbereichsende ausfallen.

4.3.3.3 Ergebnisse

Die DDA-Funktion wurde zunächst zur Funktionsvalidierung der entwickelten Testanordnung AnTeS (Modul 1 und Modul 3) verwendet. Hierzu wurde die DDA-Funktion mit dem Engineering-Tool SPACE (Modul 1 von AnTeS) modelliert und der erzeugte Code für die Anwendungssoftware in das TXS-System überspielt. Die Eingangssignale L1, L2 und L3 wurden entsprechend den vordefinierten Verläufen (vgl. Abb. 4.6 und Abb. 4.7) mit dem Modul 3 von AnTeS generiert und die DDA-Funktion angesteuert. Die Auslösung des DDA-Signals wurde über das Modul 3 überwacht.

Abb. 4.8 zeigt die aufgezeichneten Verläufe der Eingangssignale L1, L2, L3 und des zugehörigen DDA-Speichersignals. Die beiden Szenarien A und B wurden hierbei direkt nacheinander durchlaufen.

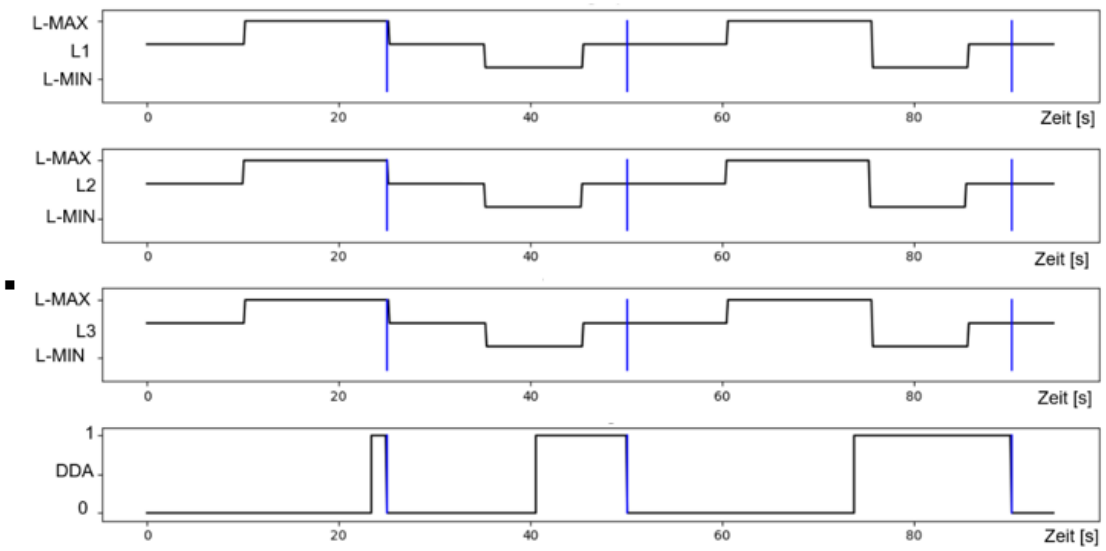

Abb. 4.8 Mit dem Modul 3 von AnTeS aufgezeichnete Verläufe der Eingangssignale L1, L2 und L3 und des DDA-Seichersignals mit Kennzeichnung der Rücksetzpunkte des DDA-Speichersignals.

Der Reset-Taster des Speichers wurde an den blau gekennzeichneten Zeitpunkten betätigt.

Wie Abb. 4.8 zu entnehmen ist, entspricht der Verlauf des DDA-Ausgangssignals dem erwarteten Verlauf aus Abb. 4.6. Ausgehend vom Sollfüllstand wird das DDA-Speichersignal 12 s nach Überschreiten der oberen Grenze L-MAX für den RDB-Füllstand gesetzt und unmittelbar zurückgesetzt, sobald der RDB-Füllstand auf seinen Sollwert zurückgeführt wurde. Im weiteren Verlauf wird das DDA-Speichersignal erwartungsgemäß 4 s nach Unterschreiten der unteren Grenze L-MIN für den RDB-Füllstand erneut gesetzt. Entsprechend den Vorgaben aus Abb. 4.6 erfolgt dann das Rücksetzen des DDA-Speichersignals 5 s nachdem der RDB-Füllstand ausgehend von der Grenze L-MIN auf seinen Sollfüllstand zurückgeführt wurde. Nach erneutem Überschreiten der Grenze L-MAX wird der Speicher entsprechend der Logik der DDA-Funktion nach 12 s gesetzt. Der RDB-Füllstand wird dann anschließend ausgehend von der Grenze L-MAX entsprechend Abb. 4.7 unmittelbar auf die Grenze L-MIN überführt. Hierbei bleibt der DDA-Speicher erwartungsgemäß gesetzt.

Für die Analyse des Verhaltens der DDA-Funktion bei Vorliegen fehlerhafter Signalzustände wurden dynamische Simulationen in Matlab/Simulink (Modul 2 von AnTeS) gemäß den Vorgaben aus Abschnitt 4.3.3.2 durchgeführt. Hierfür wurde die DDA-Funktion in Matlab/Simulink implementiert. Da der Schwerpunkt der Analysen auf der Betrachtung von Auswirkungen fehlerhafter Signalzustände des Ausgangssignals O1 der DDA-Funktion lag, wurde der Speicherbaustein in MALTAB/Simulink nicht modelliert.

Unter Berücksichtigung der Festlegungen aus Abschnitt 4.3.3.2 ergeben sich insgesamt 96 zu betrachtende Testfälle (je 48 Testfälle pro Szenario) für die Analyse der Auswirkungen fehlerhafter Signalzustände auf die Ausführung der DDA-Funktion. Jeder Testfall wird hierbei spezifiziert durch

- den vordefinierten RDB-Füllstandverlauf (Szenario A oder Szenario B),
- die Ausfallrichtung der Signale (HIGH oder LOW),
- die den Signalen zugeordneten Kennungen zur Kennzeichnung von deren Ausfall (F) oder deren Gültigkeit (OK),
- die den Signalen zugeordneten Fehlerzustandskennungen zur Angabe, ob das Signal für die Berechnung der DDA-Funktion berücksichtigt (EF=0) oder nicht berücksichtigt wird (EF=1).

Die erzielten Simulationsergebnisse wurden entsprechend gekennzeichnet.

Es wurde zunächst für jedes betrachtete Szenario (Szenario A oder Szenario B) das Verhalten der DDA-Funktion für gültige/valide Signale L1, L2, L3 ermittelt (Referenzfall), d. h., dass die Füllstandmesssignale L1, L2 und L3 den vordefinierten Verläufen für den RDB-Füllstand im jeweiligen Szenario entsprechen und vom Leittechniksystem als korrekte Signale erkannt wurden. Anschließend wurde die DDA-Funktion für jeden Testfall gemäß den Festlegungen aus Abschnitt 4.3.3.2 simuliert und die erhaltenen Ergebnisse mit denen des Referenzfalls verglichen.

Wie bereits im Abschnitt 4.3.3.2 erwähnt, bestimmt die Anzahl gültiger Signale das Verhalten des 2. MIN- bzw. 2.MAX-EFB. Die Simulationsergebnisse wurden daher in Abhängigkeit von der Anzahl nicht erkannter ausgefallener Signale bzw. nicht erkannter korrekter Signale gemäß den Definitionen aus Tab. 4.4 ausgewertet.

Im Folgenden sind die dabei gewonnenen wesentlichen Ergebnisse aufgeführt:

- Die Ausführung der DDA-Funktion im betrachteten Testfall entspricht der Ausführung der DDA-Funktion im Referenzfall.
- Das DDA-Ausgangssignal wird unabhängig vom Füllstandverlauf im RDB einsträngig über den oberen Grenzwert L-MAX bzw. über den unteren Grenzwert L-MIN ausgelöst.
- Die DDA-Funktion wird nicht ausgeführt.

Es zeigte sich eine Äquivalenz der Ergebnisse für die betrachteten Testfälle. Aus diesem Grund werden nachfolgend Simulationsergebnisse für je einen repräsentativen Testfall für, hinsichtlich der gewonnenen Erkenntnisse, äquivalente Testfälle gezeigt.

In Tab. 4.5 sind die Ergebnisse repräsentativer Testfälle zusammengestellt. Die Testfälle sind entsprechend obiger Festlegungen gekennzeichnet.

Tab. 4.5 Zusammenstellung der Ergebnisse repräsentativer Testfälle

Lfd.Nr.	Kennzeichnung	Bedeutung des Testfalls	Auswirkung auf die DDA-Funktion
1	**L1: OK; L2: OK; L3: F HIGH/LOW EF1:0; EF2:0; EF3:0**	Ausfall eines Signals nach oben bzw. nach unten Der Ausfall des Signals wurde nicht erkannt.	Die Ausführung der DDA-Funktion entspricht der Ausführung im Referenzfall.
2	**L1: OK; L2: F; L3: F HIGH/LOW EF1:0; EF2:0; EF3:1**	Ausfall zweier Signale nach oben bzw. nach unten Der Ausfall eines der beiden Signale wurde nicht erkannt.	Das DDA-Ausgangssignal wird fehlerhaft einsträngig über den oberen Grenzwert L-MAX bzw. über den unteren Grenzwert L-MIN ausgelöst.
3	**L1: F; L2: F; L3: F HIGH/LOW EF1:0; EF2:1; EF3:1**	Ausfall aller drei Signale nach oben bzw. nach unten Der Ausfall eines der drei Signale wurde nicht erkannt.	Das DDA-Ausgangssignal wird fehlerhaft einsträngig über den oberen Grenzwert L-MAX bzw. über den unteren Grenzwert L-MIN ausgelöst.
4	**L1: OK; L2: F; L3: F HIGH/LOW EF1:0; EF2:0; EF3:0**	Ausfall zweier Signale nach oben bzw. nach unten Der Ausfall beider Signale wurde nicht erkannt.	Das DDA-Ausgangssignal wird fehlerhaft einsträngig über den oberen Grenzwert L-MAX bzw. über den unteren Grenzwert L-MIN ausgelöst.
5	**L1: F; L2: F; L3: F HIGH/LOW EF1:0; EF2:0; EF3:1**	Ausfall aller drei Signale nach oben bzw. nach unten Der Ausfall von zwei der drei Signale wurde nicht erkannt.	Das DDA-Ausgangssignal wird fehlerhaft einsträngig über den oberen Grenzwert L-MAX bzw. über den unteren Grenzwert L-MIN ausgelöst.
6	**L1: F; L2: F; L3: F HIGH/LOW EF1:0; EF2:0; EF3:0**	Ausfall aller drei Signale nach oben bzw. nach unten Der Ausfall aller drei Signale wurde nicht erkannt.	Das DDA-Ausgangssignal wird fehlerhaft einsträngig über den oberen Grenzwert L-MAX bzw. über den unteren Grenzwert L-MIN ausgelöst.
7	**L1: OK; L2: OK; L3: F HIGH/LOW EF1:0; EF2:1; EF3:1**	Ausfall eines der drei Signale wurde erkannt. Eines von zwei korrekten Signalen wurde nicht erkannt.	Die Ausführung der DDA-Funktion entspricht der Ausführung im Referenzfall.
8	**L1: OK; L2: OK; L3: F HIGH/LOW EF1:1; EF2:1; EF3:1**	Ausfall eines der drei Signale wurde erkannt. Zwei korrekte Signale wurden nicht erkannt.	Keine Ausführung der DDA-Funktion, da alle drei Eingangssignale fehlerhaft als ungültig bewertet werden.
9	**L1: OK; L2: F; L3: F HIGH/LOW EF1:1; EF2:1; EF3:1**	Ausfall zwei von drei Signalen wurde erkannt. Ein korrektes Signal wurde nicht erkannt.	Keine Ausführung der DDA-Funktion, da alle drei Eingangssignale fehlerhaft als ungültig bewertet werden.

Zur Veranschaulichung der in Tab. 4.5 zusammengefassten Erkenntnisse werden nachfolgend für einige Testfälle exemplarisch der Verlauf des DDA-Ausgangssignals für den RDB-Füllstandverlauf gemäß Szenario A (s. Abb. 4.6) dargestellt.

Ähnliche Verläufe des DDA-Ausgangsignals wurden ebenfalls für den RDB-Füllstandverlauf entsprechend Szenario B (s. Abb. 4.7) ermittelt und werden aufgrund der Übersichtlichkeit nicht gezeigt.

In Abb. 4.9 ist der Verlauf des DDA-Ausgangssignals dargestellt für den Fall, dass alle drei Eingangssignale der DDA-Funktion zum Simulationsstart (Zeitpunkt 0 s) nach oben ausgefallen sind und dass die Ausfälle der Signale nicht erkannt wurden (Testfall mit laufender Nr. 6). Der Verlauf des DDA-Ausgangssignals im Referenzfall, d. h. für drei gültige/valide Eingangssignale ist ebenfalls in Abb. 4.9 dargestellt.

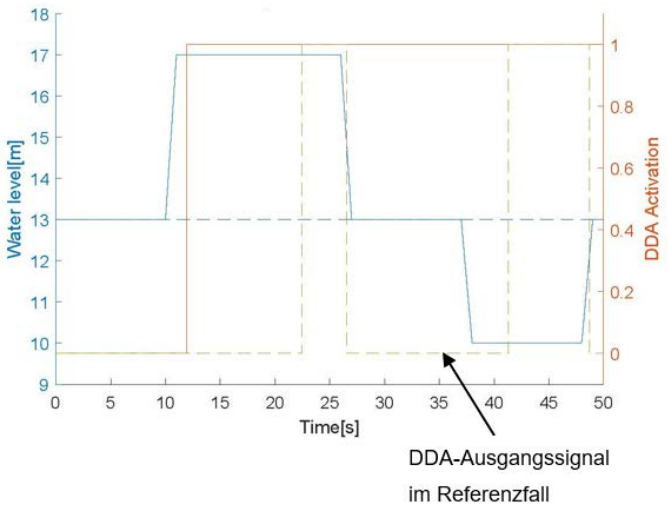

Abb. 4.9 Verlauf des DDA-Ausgangssignals bei nicht erkanntem Ausfall aller drei Eingangssignale

Verlauf des DDA-Ausgangssignals (rot) bei einem nicht erkannten Ausfall aller drei Eingangssignale nach oben: RDB-Füllstandverlauf (blau) gemäß Szenario A;

Wie Abb. 4.9 zu entnehmen ist, wird das DDA-Ausgangssignal nach Ablauf von 12 s nach Simulationsstart, unabhängig vom Verlauf des RDB-Füllstandes, unmittelbar aktiviert und bleibt für die restliche Simulationszeit aktiv, da durch den nicht erkannten Ausfall aller drei Eingangssignale nach oben, der obere Grenzwert L-MAX dauerhaft überschritten wird. Der Verlauf des DDA-Ausgangssignals im Referenzfall folgt dem vorgegebenen RDB-Füllstandverlauf. Die Auslösung des DDA-Signals über den oberen Grenzwert L-MAX erfolgt im Referenzfall erwartungsgemäß 22 s nach Simulationsstart bzw. 12 s nachdem der RDB-Füllstand den oberen Grenzwert überschritten hat. Im wei-

teren Verlauf folgen im Referenzfall ein Rücksetzen des DDA-Ausgangssignals und erneutes Setzen des DDA-Ausgangssignals 4 s nachdem der RDB-Füllstand den unteren Grenzwert unterschritten hat.

Als weiteres Beispiel ist in der Abb. 4.10 der Verlauf des DDA-Ausgangssignals für den Fall dargestellt, dass zwei der drei Eingangssignale zum Simulationsstart nach unten ausgefallen sind und dass die Ausfälle nicht erkannt wurden. Das dritte Eingangssignal ist hierbei gültig. Dies entspricht dem Testfall mit der laufenden Nr. 4 in Tab. 4.5. Der Verlauf des DDA-Ausgangssignals im Referenzfall ist ebenfalls zum Vergleich in Abb. 4.10 dargestellt. Wie Abb. 4.10 zu entnehmen ist, wird das DDA-Ausgangssignal 4 s nach Simulationsstart, unabhängig vom Verlauf des RDB-Füllstandes, aktiviert und bleibt für die restliche Simulationszeit aktiv. Dies liegt daran, dass in diesem Fall durch den nicht erkannten Ausfall von zwei Signalen nach unten der untere Grenzwert L-MIN zur Auslösung des DDA-Signals für die ganze Simulationszeit unabhängig vom RDB-Füllstandverlauf unterschritten wird, wodurch es zur Auslösung des DDA-Signals kommt.

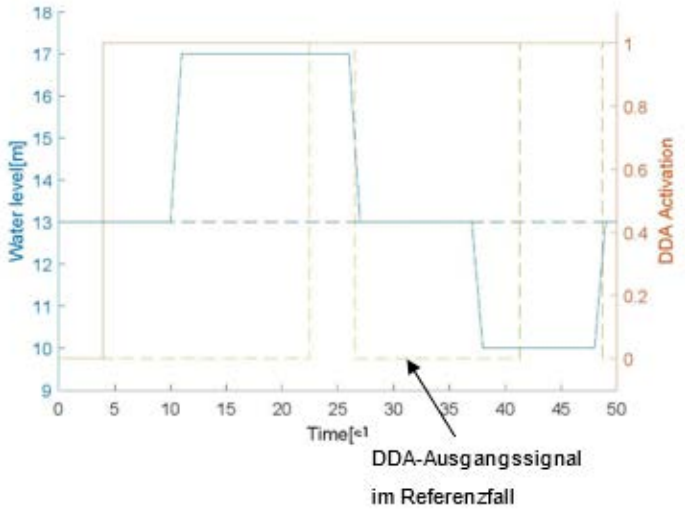

Abb. 4.10 Verlauf des DDA-Ausgangssignals bei nicht erkanntem Ausfall von zwei der drei Eingangssignale nach unten

Verlauf des DDA-Ausgangssignals (rot) bei einem Ausfall nach unten; Drittes Eingangssignal gültig; RDB-Füllstandverlauf gemäß Szenario A (blau)

Als drittes Beispiel ist in Abb. 4.11 der Verlauf des DDA-Ausgangssignals für den Fall dargestellt, dass eines der drei Signale zum Simulationsstart nach oben bzw. nach unten ausgefallen ist und dass der Ausfall dieses Signals nicht erkannt wurde (Testfall mit lau-

fender Nr. 1). In diesem Fall entspricht der Verlauf des DDA-Ausgangssignals dem Verlauf im Referenzfall (s. Abb. 4.11). Daraus folgt, dass der nicht erkannte Ausfall eines der drei Eingangssignale keine Auswirkung auf die Ausführung der DDA-Funktion im Vergleich zum Referenzfall hat.

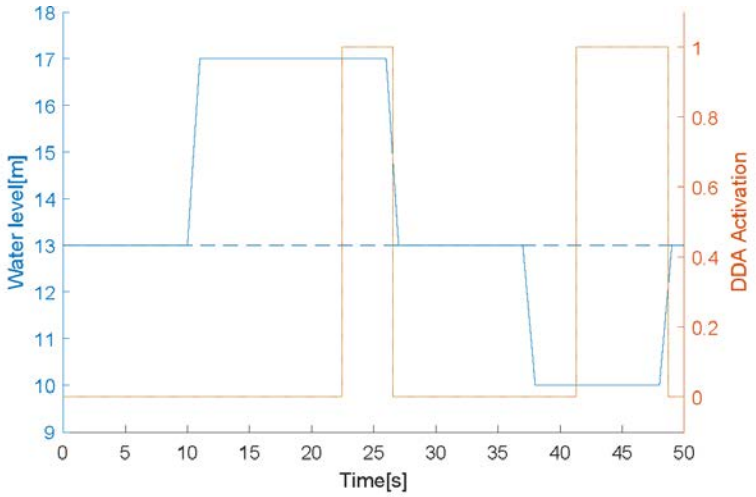

Abb. 4.11 Verlauf des DDA-Ausgangssignals bei nicht erkanntem Ausfall von einem der drei Eingangssignale nach unten/oben

Verlauf des DDA-Ausgangssignals (rot) bei einem nicht erkannten Ausfall von einem der drei Eingangssignale nach unten/oben; RDB-Füllstandverlauf gemäß Szenario A (blau)

Bei nur einem gültigen Eingangssignal (Testfall mit laufender Nr. 7) verhält sich ebenfalls das DDA-Ausgangssignal entsprechend der Abb. 4.11.

In Abb. 4.12 ist der Verlauf des DDA-Ausgangssignals für den Fall dargestellt, dass zwei der drei Eingangssignale zum Simulationsstart nach oben bzw. nach unten ausgefallen sind und dass die Ausfälle vom Leittechniksystem erkannt wurden. Das dritte korrekte Eingangssignal wurde hierbei vom Leittechniksystem nicht erkannt bzw. als fehlerhaft gekennzeichnet. Wie auf Abb. 4.12 zu sehen ist, erfolgt in diesem Fall kein Auslösen des DDA-Ausgangssignals, da alle drei Eingangssignale vom Leittechniksystem als fehlerhaft angesehen und entsprechend gekennzeichnet werden. Diese Fehlerzustände werden entsprechend der Spezifikationen der in der DDA-Funktion verwendeten elementaren Funktionsbausteine (2. MIN, 2. MAX, COMP, OR) bis zum Ausgang der DDA-Funktion weitergeleitet. Der Ausgang der DDA-Funktion wird unabhängig vom Verlauf des RDB-Füllstandes für die ganze Simulationszeit auf den Wert 0 mit der Fehlerzustandskennung EF=1 gesetzt.

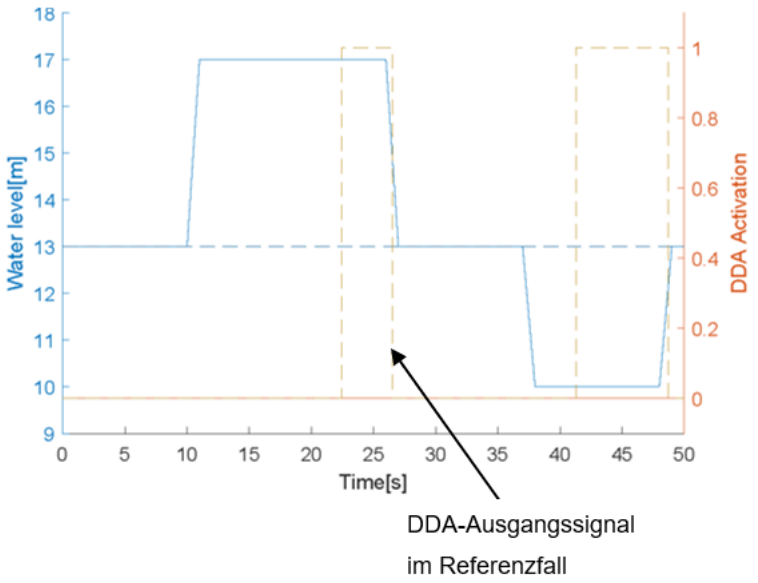

Abb. 4.12 Verlauf des DDA-Ausgangssignals bei erkanntem Ausfall von zwei von drei Signalen

Verlauf des DDA-Ausgangssignals (rot) beim erkannten Ausfall zwei von drei Signalen. Ein korrektes Signal wurde nicht erkannt. RDB-Füllstandverlauf gemäß Szenario A (blau)

Anhand der durchgeführten Simulationen am Beispiel der DDA-Funktion konnte gezeigt werden, dass nicht erkannte Ausfälle von Eingangssignalen bzw. fehlerhaft als ausgefallen gekennzeichnete korrekte Eingangssignale zur fehlerhaften Ausführung einer Leittechnikfunktion bzw. zur Nichtausführung einer Leittechnikfunktion führen können.

4.3.4 Ergänzende Aspekte zur Automatisierung des SFMEA-Konzepts

Die im Rahmen dieses Arbeitspaketes 3 entwickelte und implementierte Methode zur automatisierten Bestimmung der Auswirkungen fehlerhafter Signalzustände auf die Ausgangssignale elementarer Funktionsblöcke bzw. auf Leittechnikfunktionen baut auf der automatischen Bestimmung aller möglichen Eingangskombinationen fehlerhafter Signalzustände und der Ermittlung von deren Auswirkungen auf die Berechnung des Ausgangsignals bzw. der Ausgangsignale der elementarer Funktionsblöcke bzw. Leittechnikfunktionen auf. Sie kann als sogenannte „Brute-Force-Methode“ betrachtet werden.

Als „Brute-Force-Methode“, die alle Eingangskombinationen überprüft und damit einen exponentiell mit der Anzahl der Eingangssignale steigenden Ressourcenbedarf hat, ist die automatische Bestimmung der Ausfallmodi elementarer Funktionsblöcke bzw. Leit-

technikfunktionen in der Komplexität der zu untersuchenden Funktionsblöcke bzw. Leittechnikfunktionen begrenzt. Bei der Durchführung der automatisierten FMEA hinsichtlich der Signalzustände der Eingangssignale der elementaren Funktionsblöcke liegt die Anzahl der zu betrachtenden Kombinationen bei 2^{2n}, wobei *n* die Anzahl der Eingangssignale des zu testenden Funktionsblockes ist.

Beachtet werden sollte auch, dass bei Funktionsblöcken oder leittechnischen Funktionen, deren Verhalten nicht nur von den Signalzuständen der Eingangssignale, sondern auch von den Werten der Eingangssignale abhängt (z. B. 2. MIN-EFB oder 2. MAX-EFB), schnell eine größere Anzahl von Eingangskombinationen auftritt, da in diesen Fällen die Signalwerte auch variiert werden müssen. In solchen Fällen sind daher die relevanten zu untersuchenden Fällen zu ermitteln, um die Anzahl der Eingangskombinationen zu reduzieren.

4.4 Zusammenfassung

In diesem Kapitel wurden die im Rahmen der Bearbeitung des Arbeitspaketes 3 erzielten Ergebnisse vorgestellt. Ziel war es, ein Konzept zur Validierung der Analyseergebnisse einer durchgeführten Software-FMEA zu den Auswirkungen von latenten Softwarefehlern in softwarebasierten Leittechniksystemen mittels Testanordnungen zu entwickeln.

Das entwickelte Validierungskonzept, welches die automatisierte Durchführung einer SFMEA einschließt, baut auf der Anwendung der Software-Fault-Injection-Methode auf. Hierbei werden die Signale durch Eingabe von Fehlern gezielt beeinflusst bzw. fehlerhafte Signalzustände als „Bitflips“ modelliert.

Die Auswirkungen latenter Softwarefehler wurden anhand dynamischer Simulationen untersucht.

Das Prinzip der Simulationen besteht darin, alle möglichen Kombinationen der Fehlerstatus der Eingangssignale zu ermitteln und deren Auswirkung auf den Fehlerstatus der zugehörigen Ausgangssignale des elementaren Funktionsblockes auszuwerten.

Zur Umsetzung dieses Validierungskonzeptes wurde als Testanordnung zur Nachbildung der Leittechnikfunktionen eines softwarebasierten Leittechniksystems das Analyse und Testsystem (AnTeS) der GRS eingesetzt, welches im Rahmen der Bearbeitung des Arbeitspaketes 3 dieses Vorhabens weiterentwickelt wurde.

Ergebnisse durchgeführter dynamischer Simulationen mit dem Analyse- und Testsystem (AnTeS) der GRS zur Validierung des SFMEA-Analysekonzeptes belegen übereinstimmende Ergebnisse zwischen der automatisch durchgeführten und der per Hand durchgeführten SFMEA elementarer Funktionsbausteine. Weiterhin wurde am Beispiel einer aus mehreren elementaren Funktionsbausteinen bestehenden Leittechnikfunktion die Eignung des SFMEA-Konzeptes zur Bestimmung des Ausfallverhaltens von Leittechnikfunktionen bei Vorliegen fehlerhafter Eingangssignale (Status und Wert) erprobt. Hierbei konnte anhand durchgeführter Simulationen gezeigt werden, dass nicht erkannte Ausfälle von Eingangssignalen bzw. fehlerhaft als ausgefallen identifizierte korrekte Eingangssignale zur fehlerhaften Ausführung der Leittechnikfunktion bzw. zur Nichtausführung der Leittechnikfunktion führen können.

5 Zusammenfassung und Schlussfolgerung

Der Schwerpunkt dieses Vorhabens lag darin, die Auswirkungen von Softwarefehlern innerhalb von softwarebasierten Leittechniksystemen systematisch zu untersuchen. Ein wesentlicher Aspekt war hierbei die Untersuchung von potenziellen Ausbreitungspfaden von Fehlern in softwarebasierten Leittechniksystemen.

Dazu wurden zunächst auf Grundlage der Auswertung des relevanten Standes von Wissenschaft und Technik, Modelle zur Untersuchung der Ausbreitung von Fehlern in softwarebasierten Leittechniksystemen erarbeitet. Hierfür wurde ein Fehler als ein latent anzusehender Mangel in einer Hardware- oder Softwarekomponente eines softwarebasierten Leittechniksystems definiert, der bei entsprechender Aktivierung zum Versagen/Ausfall der betroffenen Komponente führen kann. Hierunter fallen auch Mängel in der Dokumentation des Leittechniksystems. In der Folge kann es zur Nichtausführung oder zur fehlerhaften Ausführung der betroffenen leittechnischen Funktion kommen. Ist ein Fehler aktiviert, kann er sich durch Folgeausfälle von Komponenten (Kaskadenmodell) oder durch Ausfall aufgrund gemeinsamer Ursache (GVA-Modell) im softwarebasierten Leittechniksystem ausbreiten. Zu den möglichen Ursachen für die Aktivierung latenter Fehler zählen beispielsweise Fehler bei Instandhaltung und Modifikation des Systems. Zur Veranschaulichung der Fehlerausbreitungsmodelle in softwarebasierten Leittechniksystemen wurden exemplarisch in der Betriebserfahrung aufgetretene Ereignisse mit Softwarefehlern vorgestellt, bei denen es zur Ausbreitung von Softwarefehlern innerhalb eines Leittechniksystems kam.

Anschließend wurden für das Vorhaben relevante Regelwerksanforderungen an softwarebasierte Leittechniksysteme identifiziert und in Bezug auf die Vorsorge gegen die Ausbreitung von Softwarefehlern in softwarebasierten Leittechniksystemen ausgewertet. Hierfür wurden DIN-Normen, IAEA-Standards sowie das deutsche kerntechnische Regelwerk (Sicherheitsanforderungen für Kernkraftwerke, KTA-Regelwerk) betrachtet. Es zeigte sich, dass in diesen Anforderungen Aspekte enthalten sind, die der Vermeidung und Ausbreitung von Softwarefehlern in softwarebasierter Leittechnik dienen. Hierzu zählen u. a. die Entwicklung der Software nach einem Phasenmodell in verifizierbaren Schritten, die Gestaltung der Softwarearchitektur als eigenständige aufgabenspezifische Softwareeinheiten, der Nachweis der korrekten Arbeitsweise der Software sowie die Verifizierung der Ergebnisse der einzelnen Phasen der Softwareentwicklung unter Anwendung systematischer Analysen sowie der Einsatz von fehlervermeidenden und

fehlerbeherrschenden Maßnahmen zwecks Erzielung einer fehlertoleranten Auslegung des Leittechniksystems.

In einem weiteren Arbeitsschritt wurden in softwarebasierten Leittechniksystemen eingesetzte Fehlertoleranzverfahren (Fehlererkennungs- und -behandlungsmethoden) recherchiert und die hieraus gewonnenen Erkenntnisse übersichtlich dargestellt. Zur Erzielung eines fehlertoleranten Systemverhaltens sind grundsätzlich zwei Aufgaben zu erfüllen: Die Fehlererkennung und die Fehlerbehandlung. In softwarebasierten Leittechniksystemen werden zur Fehlererkennung verschiedene Techniken eingesetzt wie z. B. die Selbstüberwachung und die Überwachung durch externe Mittel wie z. B. der sogenannte Watchdog-Timer. Die Fehlerbehandlung erfolgt in der Regel unter Ausnutzung des Redundanzprinzips. Sie dient dem Wiederherstellen eines fehlerfreien Systemzustands nach dem Auftreten/Erkennen eines Fehlers. Das Redundanzprinzip kann hierbei auf die Hardware-, Software- und Informationsebene angewendet werden.

Weiterhin wurde der aktuelle Stand von Wissenschaft und Technik zu Zuverlässigkeitsanalysemethoden für softwarebasierte Leittechniksysteme dargestellt. Die hierbei ermittelten Methoden wurden hinsichtlich deren Eignung zur Analyse der Auswirkungen und insbesondere der Ausbreitung von Softwarefehlern in softwarebasierten Leittechniksystemen ausgewertet. Es zeigte sich, dass mit den ermittelten Zuverlässigkeitsanalysemethoden die Auswirkungen von Fehlern/Softwarefehlern in softwarebasierten Leittechniksystemen grundsätzlich analysiert werden können. In Bezug auf die Identifizierung von potenziellen Ausbreitungspfaden von latenten Softwarefehlern, wie z. B. in Leittechnikfunktionen, erweist sich die Methode der Fehlerzustands- und -auswirkungsanalyse (FMEA) aufgrund ihrer induktiven und systematischen Vorgehensweise als besonders geeignet.

In einem weiteren Arbeitsschritt wurde ein Konzept zur Durchführung einer Software-Fehlerart- und Auswirkungsanalyse, kurz SFMEA, für ein softwarebasiertes Leittechniksystem entwickelt. Das Ziel des entwickelten SFMEA-Konzeptes ist es, potenzielle Ausfälle aufgrund latenter Softwarefehler in einem softwarebasierten Leittechniksystem zu identifizieren und deren Auswirkungen auf die Leittechnikfunktionen (Ausfall auf Anforderung, Fehlbetätigung usw.) zu untersuchen. Hierbei werden die Ausbreitungspfade der identifizierten Ausfallarten analysiert. In einem ersten Schritt werden zunächst die mit dem Anlagenprozess verknüpften Leittechnikfunktionen modelliert. Da Leittechnikfunktionen typischerweise anhand von miteinander verknüpften elementaren Funktions-

bausteinen spezifiziert werden, werden in diesem Projekt die elementaren Funktionsbausteinen als die niedrigste Abstraktionsebene zur Durchführung einer FMEA definiert. Folglich wird bei dem entwickelten SFMEA-Konzept die Analyse auf der Ebene der elementaren Funktionsblöcke durchgeführt. Dies dient dazu, die relevanten Ausfallmodi der Eingangssignale der elementaren Funktionsblöcke systematisch zu identifizieren. Die Evaluierung der Auswirkungen der so identifizierten Ausfallarten der Eingangssignale auf die Ausgangssignale elementarer Funktionsblöcke bzw. auf die zu untersuchenden Leittechnikfunktionen kann anschließend mittels dynamischer Simulation oder Fehlerbaumanalyse durchgeführt werden. Diese Evaluierung schließt auch die Untersuchung der Auswirkungen fehlerhaft gesetzter Parameter von elementaren Funktionsblöcken mit ein. Anhand einiger ausgewählter Anwendungsbeispiele des entwickelten SFMEA-Konzeptes konnte im Rahmen der durchgeführten Arbeiten gezeigt werden, dass insbesondere fehlerhafte Zustände der Eingangssignale (nicht erkannte Ausfälle von Eingangssignalen bzw. als fehlerhaft identifizierte aber korrekte Eingangssignale) die identifizierten Ausfallmodi der Ausgangssignale von elementaren Funktionsblöcken wesentlich bestimmen. Deshalb wurde die Ausbreitung der fehlerhaften Zustände der Eingangssignale in einem Leittechniksystem und deren Auswirkungen auf die korrekte Funktion des Leittechniksystems vertieft untersucht.

Bei den Untersuchungen zeigte sich, dass für elementare Funktionsblöcke mit einer großen Anzahl von Eingangssignalen und Parametern die Anwendung der SFMEA-Methode per Hand sehr zeitaufwändig bis unmöglich wird. Daher wurde eine Automatisierung der entwickelten SFMEA-Methode mittels dynamischer Simulationen erarbeitet.

Das Prinzip der Automatisierung besteht darin, alle möglichen Kombinationen der Fehlerstatus der Eingangssignale zu ermitteln und deren Auswirkungen auf den Fehlerstatus der zugehörigen Ausgangssignale des elementaren Funktionsblockes anhand dynamischer Simulationen zu untersuchen und auszuwerten.

In einem weiteren Arbeitsschritt wurde ein Konzept zur Validierung der Analyseergebnisse einer durchgeführten SFMEA an einem softwarebasierten Leittechniksystem anhand von Testanordnungen entwickelt, welches die automatisierte Durchführung einer SFMEA einschließt. Das entwickelte Validierungskonzept baut auf der Anwendung der Software-Fault-Injection-Methode auf. Hierbei werden die Signale durch Eingabe von Fehlern gezielt beeinflusst bzw. fehlerhafte Signalzustände als „Bitflips“ modelliert.

Zur Umsetzung dieses Validierungskonzeptes wurde als Testanordnung das Analyse- und Testsystem (AnTeS) der GRS eingesetzt, welches im Rahmen dieses Vorhabens weiterentwickelt wurde. Das AnTeS besteht im derzeitigem Entwicklungsstand aus drei Modulen: Einem realen softwarebasierten Leittechniksystem und einem simulierten softwarebasierten Leittechniksystem, mit denen Leittechnikfunktionen eines softwarebasierten Leittechniksystems modelliert werden können, und einem Modul zur Generierung und Überwachung von Ein- und Ausgangssignalen der Leittechniksysteme. Im realen und im simulierten Leittechniksystem können gezielt Fehler injiziert und deren Auswirkungen analysiert werden (Fault-Injection). Diese Möglichkeit, Signale gezielt durch Fault-Injection zu beeinflussen, wurde im Rahmen der Validierung des entwickelten SFMEA-Konzeptes in AnTeS implementiert und in dynamischen Simulationen verwendet.

Durchgeführte dynamische Simulationen mit AnTeS haben übereinstimmende Ergebnisse zwischen der automatisch durchgeführten und der per Hand durchgeführten SFMEA elementarer Funktionsbausteine belegt. Weiterhin wurde am Beispiel einer aus mehreren elementaren Funktionsbausteinen bestehenden Leittechnikfunktion die Eignung des SFMEA-Konzeptes zur Bestimmung des Ausfallverhaltens von Leittechnikfunktionen bei Vorliegen fehlerhafter Eingangssignale (Status und Wert) erprobt. Hierbei konnte anhand durchgeführter Simulationen gezeigt werden, dass nicht erkannte Ausfälle von Eingangssignalen bzw. fehlerhaft als ausgefallen identifizierte korrekte Eingangssignale zur fehlerhaften Ausführung der Leittechnikfunktion bzw. zur Nichtausführung der Leittechnikfunktion führen können.

Die im Rahmen dieses Vorhabens entwickelte und implementierte SFMEA-Methode kann für die Untersuchung komplexerer Leittechnikfunktionen weiterentwickelt werden. Da diese Methode auf der automatischen Bestimmung aller möglichen Eingangskombinationen fehlerhafter Signalzustände und der Ermittlung von deren Auswirkungen auf die Berechnung des Ausgangsignals bzw. der Ausgangsignale der elementaren Funktionsblöcke bzw. Leittechnikfunktionen aufbaut, geht dies mit einem exponentiell mit der Anzahl der Eingangssignale steigenden Ressourcenbedarf einher, welcher die automatische Bestimmung der Ausfallmodi elementarer Funktionsblöcke bzw. des Ausfallverhaltens der Leittechnikfunktionen in der Komplexität der zu untersuchenden Funktionsblöcke bzw. Leittechnikfunktionen begrenzt.

Beachtet werden sollte auch, dass bei Funktionsblöcken oder leittechnischen Funktionen, deren Verhalten nicht nur von binären Signalzuständen der Eingangssignale, sondern auch von den quantisierten Werten analoger Eingangssignale abhängt (z. B. 2. MIN-EFB oder 2. MAX-EFB), eine noch größere Anzahl von Eingangskombinationen auftritt, da in diesen Fällen die Signalwerte ebenfalls variiert werden müssen. Eine Optimierung der im Rahmen dieses Vorhabens entwickelten Methode hinsichtlich dieser Aspekte stellt einen möglichen Schwerpunkt für künftige Arbeiten dar, um das Ausfallverhalten komplexerer Leittechnikfunktionen im Rahmen der Zuverlässigkeitsanalyse softwarebasierter Leittechniksysteme effizient untersuchen zu können.

Literatur

/ABU 08/ Chris Price, Neal Snooke, An Automated Software FMEA, Department of Computer Science, Aberystwyth University, 2008

/AIA 08/ AIAG, Automotive Industry Action Group: Potential Failure Mode and Effect Analysis (FMEA), Version 4, 2008

/AVE 14/ Uncertainty in Risk Assessment, Chapter 8: Uncertainty representation and propagation in event tree analysis, T. Aven, P. Baraldi, R. Flage, E. Zio, Online ISBN:9781118763032, John Wiley & Sons, Ltd, 2014

/BAI 05/ Software failure prediction based on a Markov Bayesian network model, C.G. Bai, Q.P. Hu, M. Xie, S.H. Ng, Journal of Systems and Software Volume 74, Issue 3, 1 February 2005, Pages 275-282

/BEH 09a/ Software FMEA und Fehlerbaumanalyse, P. Behrmann, Hochschule für Angewandte Wissenschaften Hamburg - Fachbereich Informatik, WS 08/09

/BEH 09/ Software Failure Modes and Effects Analysis & Failure Tree Analysis, P. Behrmann, Hochschule für Angewandte Wissenschaften Hamburg - Fachbgebiet Informatik, Januar 2009

/BEL 03/ Bela G. Liptak, (2003). Instrument Engineers' Handbook, Volume Two: Process Control and Optimization. CRC Press

/BHU 15/ K. Umadevi, S. Brintha Rajakumari, A Review on Software Fault Injection Methods and Tools, International Journal of Innovative Research in Computer and Communication Engineering, 2015

/BMU 15/ Bundesministerium für Umwelt, Naturschutz, Bau und Reaktorsicherheit, Bekanntmachung der „Sicherheitsanforderungen an Kernkraftwerke“ vom 3. März 2015

/BMU 15b/ Bundesministerium für Umwelt, Naturschutz, Bau und Reaktorsicherheit, Interpretationen zu den Sicherheitsanforderungen an Kernkraftwerke, geändert am 3. März 2015

/BOY 98/ An Introduction to Markov Modeling: Concepts and Uses, M.A. Boyd, Reliability and Maintainability Symposium, Januar 1998, Anaheim, CA

/CLA 16/ Clarke, P., CHAZOP: Assessing the risks of control system failure, 2016, xSeriCon, abrufbar auf www.xsericon.com https://www.xsericon.com/res/CHAZOP%20paper%20(lite)%20r0.1.pdf

/DIN 02/ Deutsches Institut für Normung (DIN): Kernkraftwerke - Leittechnik für Systeme mit sicherheitstechnischer Bedeutung - Allgemeine Systemanforderungen, DIN EN 61513:2002, März 2002.

/DIN 05/ Deutsches Institut für Normung (DIN): Kernkraftwerke – Leittechnische Systeme mit sicherheitstechnischer Bedeutung – Kategorisierung leittechnischer Funktionen, DIN IEC 61226:2005, April 2005.

/DIN 10/ Deutsches Institut für Normung (DIN): Leittechnik für Systeme mit sicherheitstechnischer Bedeutung - Softwareaspekte für rechnerbasierte Systeme zur Realisierung von Funktionen der Kategorie A, DIN EN 60880:2009, März 2010.

/DIN 12/ Analysemethoden für Zuverlässigkeit- Petrinetze, Deutsche Fassung EN 62551:2012, August 2013

/DIN 19a/ Deutsches Institut für Normung (DIN): Leittechnische Systeme mit sicherheitstechnischer Bedeutung- Datenkommunikation in Systemen, die Funktionen der Kategorie A ausführen (IEC 61500:2018); Deutsche Fassung EN IEC 61500:2019.

/DIN 19b/ Deutsches Institut für Normung (DIN): Kernkraftwerke – Leittechnische Systeme mit sicherheitstechnischer Bedeutung – Physikalische und elektrische Trennung (IEC 60709:2018); Deutsche Fassung EN IEC 60709:2019

/DIN 81/ DIN 25424 Teil 1: Fehlerbaumanalyse, Methoden und Bildzeichen, 1981

/DIN 90/ DIN 25424 Teil 2: Fehlerbaumanalyse, Handrechenverfahren zur Auswertung eines Fehlerbaums, 1990

/DIN 07/ DIN EN 61025: Fehlzustandsbaumanalyse, 2007

/DOD 80/ Military Standard, Procedures for Performing a Failure Mode, Effects and Criticality Analysis. MIL-STD-1629A, U.S. Department of Defense, Washington, D.C. (Cancelled 4.8.1998).

/DUB 13/ Fault Tolerant Design, ISBN 978-1-4614-2113-9 Springer Verlag:2013

/GÄR 01/ Gärtner, F, Formale Grundlagen der Fehlertoleranz in verteilten Systemen, Darmstadt, Technische Universität, Dissertation 2001.

/GRS 15/ Dr. Gottschall, M., Dr. Lindner, F., Piljugin, E., Vogt, P.: Entwicklung eines Ansatzes zur Analyse der Netzwerktechnologien in sicherheitsrelevanten Leittechniksystemen hinsichtlich Verarbeitung und Auswirkung postulierter Fehler, GRS-377, 163 S, ISBN , Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) mbH: Köln, 2015.

/GRS 16/ Jopen, M., Quester, C., Römer, S., Sommer, D., Stiller, J., Ulrich, B.: Zuverlässigkeitsbewertung unter neuen Anforderungen an Sicherheitsleittechnik in Kernkraftwerken: Analyse der Anwendungspraxis, GRS-441, 286 S., ISBN 978-3-946607-23-6, Gesellschaft für Anlagen- und Reaktorsicherheit (GRS)gGmbH: 2016.

/GRS 17/ Jopen, M., Mbonjo, H., Sommer, D., Ulrich, B.: Auswirkungsbereiche von Softwarefehlern in sicherheitstechnisch wichtigen Einrichtungen von Kernkraftwerken, GRS-456, 224 S., ISBN 978-3-946607-39-7, Gesellschaft für Anlagen- und Reaktorsicherheit (GRS)gGmbH: 2017.

/GRS 17b/ Arians, R., Arnold, S., Lindner, F., Mbonjo, H., Quester, C., Sommer, D.: Aufstellung von Kriterien und Kenngrößen zur deterministischen Prüfung der Eignung von Redesign-Komponenten für den Einsatz in der Sicherheitsleittechnik von Kernkraftwerken, GRS-395, 168 S., ISBN 978-3-944161-76-1, Gesellschaft für Anlagen- und Reaktorsicherheit (GRS)gGmbH: 2017.

/GRS 18/ Müller, C., Peschke J., Piljugin E.: Entwicklung und Erprobung eines Werkzeugs zur Sensitivitätsanalyse der Fehlerauswirkungen in der sicherheitsrelevanten digitalen Leittechnik, GRS-494, 128 S, ISBN 978-3-946607-79-3, Gesellschaft für Anlagen- und Reaktorsicherheit (GRS)gGmbH:2018

/GRU 05/ Petri-Netze - eine informelle Einführung, U. Grude, Beuth Hochschule für Technik Berlin - Fachbereich Informatik und Medien, 2003

/IAE 09/ Protecting against common cause failures in digital i&c systems of nuclear power plants, IAEA NUCLEAR ENERGY SERIES No. NP-T-1.5

/IEC 06/ IEC 60812: Analysetechnik für die Funktionsfähigkeit von Systemen – Verfahren für die Fehlerzustandsart- und –auswirkungsanalyse (FMEA), 2006.

/IEC 07/ IEC 62340: Nuclear power plants - Instrumentation and control systems important to safety - Requirements for coping with common cause failure (CCF), 2007.

/IEC 09/ IEC 60709: Nuclear power plants – Instrumentation and control systems important to safety – Separation, 2009.

/IVS 20/ IVSS Sektion Chemie, Risikobeurteilung in der Anlagensicherheit, Das PAAG-/HAZOP-Verfahren und weitere praxisbewärte Methoden, März 2020

/INS 09/ Arbeitspapier Modellierung, , Institut für Qualität und Wirtschaftlichkeit im Gesundheitswesen (IQWiG), Köln, Version 1.0 vom 12.10.2009

/JOC 15/ Jockenhövel-Barttfeld, M., Bäckström,O., Holmberg, J.-E., Porthin, M., Taurines, A., Tyrväinen, T. (2015, 3 March) Modelling Software Failures of Digital I&C in Probabilistic Safety Analyses based on the TELEPERM XS Operating Experience. In atw. Vol. 60.

/JOH 20/ https://www.johner-institut.de/blog/iso-14971-risikomanagement/fault-tree-analysis-fta/, abgerufen am 02.05.2020

/KAU 15/ Markovian Reliability Analysis for Software using Error Generation and Imperfect Debugging, M. Kaushik, G. Kumar, Proceedings of the International Multi Conference of Engineers and Computer Scientists 2015 Vol I, IMECS 2015, March 18 - 20, 2015, Hong Kong

/KOR 01/ Software Safety Analysis Techniques for Developing Safety Critical Software in the Digital Protection System of the LMR, Korean Atomic Energy Research Institute, KAERI/AR-591/2001

/KRU 12/ Modellierung und Analyse eingebetteter und verteilter Systeme, Vorlesung am Lehrstuhl für praktische Informatik, Technische Universität Dortmund, H. Krumm, WS 11/12

/KTA 15/ KTA 3501 „Reaktorschutzsystem und Überwachungseinrichtungen des Sicherheitssystems“, Fassung 2015-11

/LAW 93/ Lawrence, J.D. Software Reliability and Safety in Nuclear Reactor Protection Systems, Lawrence Livermore National Laboratory, June 1993

/LAW 95/ Lawrence, J.D.: Software Safety Hazard Analysis, UCRL-ID-122514, Lawrence Livermore National Laboratory (1995)

/LEU 91/ Leu, S.-W., Fernandez E. B., Khoshgoftaa, T., Fault-tolerant software reliability modeling using Petri nets, Microelectronics Reliability, Vol. 31, Issue 4, 1991

/LEV 85/ N. G. Levenson, J. L. Stolzy., Analyzing safety nd fault tolerance using time petri nets, 1985, abrufbar unter https://link.springer.com/content/pdf/10.1007/3-540-15199-0_22.pdf

/LUT 98/ R. Lutz, G. Helmer Safety analysis of requirements for a product family, Third International Conference on Requirements Engineering 1998

/MBO 16/ H. Mbonjo, M. Jopen, B. Ulrich, D. Sommer: “Approach for the Evaluation of the Impact of Potential Software Failures In Software-Based Instrumentation And Control (I&C) Equipment in Nuclear Power Plants (NPP)”, Proceedings of the 2016 24th International Conference on Nuclear Engineering (ICONE), June 26-30, 2016, Charlotte North Carolina

/MBO 19/ Mbonjo, H., and Piljugin, J. (2019). Approach for evaluation of software failure modes in software-based I&C systems in nuclear power plants. Proceedings of the 29th European Safety and Reliability Conference (ESREL 2019), Hannover, Germany.

/NEA 15/ NEA/CSNI: Failure Modes Taxonomy for Reliability Assessment of Digital Instrumentation and Control Systems for Probabilistic Risk Analysis, Nuclear Safety. NEA/CSNI/R(2014)16, February 2015.

/NEU 05/ An Introduction to Software Failure Modes Effects Analysis (SFMEA), A. M. Neufelder, SoftRel, LLC, 2005

/NUR 81/ Fault Tree Handbook, Systems and Reliability Research Office of Nuclear Regulatory Research, U.S. Nuclear Regulatory Research, NUREG-0492, 1981

/NUR 97/ NUREG-0800, Standard Review Plan: BTP HICB–14, Guidance on Software Reviews for Digital Computer-Based Instrumentation and Control Systems, U.S. Nuclear Regulatory Commission (1997)

/PAR 07/ Safety Analysis of Safety-Critical Software for Nuclear Digital Protection System, G.-Y. Park, J.-S. Lee, S.-W. Cheon, K.-C. Kwon, E. Jee, K. Y. Koh, SAFECOMP, 2007

/PAR 08/ Fault Tree Analysis of KNICS RPS Software, G.-Y. Park, K. Y. Koh, E. Jee, P. Seong, K.-C. Kwon, D. H. Lee, Nuclear Engineering and Technology, Vol. 40, No. 5, Aug. 2008

/POT 12/ Paolo Prinetto, Fault Injection Techniques and Tools, Lecture, 2012

/PUM 99/ The Principled Design of Computer System Safety Analyses, Dissertation, D.J. Pumfrey, Universität New York, Sept. 1999

/SAE 00/ Potential Failure Mode and EffectsAnalysis in Design (Design FMEA) and Potential Failure Mode and Effects Analysis in Manufacturing and Assembly Processes (Process FMEA) Reference Manual. SAE J-1739.

/SAL 10/ A Survey of Online Failure Prediction Methods, F. Salfner, M. Lenk, M. Malek, ACM Computing Surveys, Article no. 10, March 2010

/SAN 08/ Modellierung und Analyse, Kapitel 4: Markovketten, W. Sandmann, Universität Bamberg - Fakultät für Wirtschaftsinformatik und Angewandte Informatik, WS 07/08

/SEI 07/ Phasenmodelle und Investmentstilanalyse von Hedge- und Investmentfonds, Dissertation, Universität Bremen, K. Seiler, Peter Lang Internationaler Verlag der Wissenschaften, 2007

/SGS 20/ https://www.sgs-tuev-saar.com/de/funktionale-sicherheit /sicherheitsanalytik/markov-analyse.html, abgerufen am 03.05.2020

/SPE 13/ Terminology Proposal: Redundancy for Fault Tolerance; Abrufbar unter http://www.ieee802.org/1/files/public/docs2013/new-tsn-specht-redundancy-terminology-20130115-v01.pdf

/STO 06/ Ein Markov-Modell zur Ermittlung der Kosten-Nutzen-Relation von Risikoscores zur Prognose der koronaren Herzkrankheit, Dissertation, Universität Dortmund & Universität zu Köln, B. Stollenwerk, Juni 2006

/STU 02/ Haapanen Pentti, Helminen Atte: Failure Mode and Effects Analysis of Software-based Automation Systems, STUK-YTO-TR 190, August 2002

/TRI 04/ S Butler, Ricky & Tribble, Alan & Miller, Steven & Lempia, David. (2004). Software Safety Analysis of a Flight Guidance System.

/TUD 12/ Michael Jugovac: Ausarbeitung zum Seminar “Softwarebasierte Fehlertoleranz”, TU Dortmund, 2012

/TÜV 20/ https://www.sgs-tuev-saar.com/de/funktionale-sicherheit/sicherheitsanalytik/eta.html, SGS TÜV SAAR; abgerufen am 26.04.2020

/TÜV 20a/ https://www.sgs-tuev-saar.com/de/funktionale-sicherheit/sicherheitsanalytik/markov-analyse.html, SGS TÜV SAAR; abgerufen am 26.04.2020

/UMA 07/ University of Massachusetts Amherst, Electrical and Computer Engineering, Fault Tolerant Systems, abrufbar unter http://euler.ecs.umass.edu/ece655/pdf/Part14-ch5-swft1.pdf

/UPO 16/ Universität Potsdam, Vorlesung „Zuverlässigkeit und Fehlertoleranz“, Design- und Testmethodik https://www.unipotsdam.de/fileadmin/projects/desn/lehre/ws16/zft/ZFT_01_Fehlertoleranz.pdf

/URO 15/ Universität Roma, Fault-tolerant design techniques, abrufbar unter https://www.dis.uniroma1.it/~disanzo/CP2015-Slides/II%20-%20Fault-Tolerant-techniques.pdf

/UER 12/ Fabian Scheler, Verlässliche Echtzeitsysteme, Friedrich-Alexander-Universität Erlangen-Nürnberg, Lehrstuhl Informatik 4, April 2012

/UER 14/ Peter Matthias Ulbrich, Ganzheitliche Fehlertoleranz in eingebetteten Softwaresystemen, 2014

/UIL 97/ Mei-Chen Hsueh, Timothy K. Tsai, Ravishankar K. Iyer, Fault Injection Techniques and Tools, University of Illinois at Urbana-Champaign, 1997

/UNI 20/ http://www.informatik.uni-hamburg.de/TGI/PetriNets/faq/, abgerufen am 27.04.2020

/UNI 20a/ https://www.hs-augsburg.de/informatik/projekte/mebib/emiel/entw_inf/lernprogramme/petrinetze/B_E_Netz_Uebersicht.html, abgerufen am 27.04.2020

/UNI 20b/ https://www.wiwi.uni-kl.de/bisor-orwiki/Petrinetze, abgerufen am 27.04.2020

/UNI 20c/ https://www.hs-augsburg.de/informatik/projekte/mebib/emiel/entw_inf/lernprogramme/petrinetze/NebenlaeufigkeitHTML.html, abgerufen am 27.04.2020

/UNI 20d/ https://www.hs-augsburg.de/informatik/projekte/mebib/emiel/entw_inf/lernprogramme/petrinetze/VerzweigungHTML.html, abgerufen am 27.04.2020

/UNI 20e/ https://uol.de/f/2/dept/informatik/ag/lks/download/abteilung/ sicherheits-analyse-tutorial.pdf, abgerufen am 26.07.2020

/UNI 20f/ Universität Duisburg Essen, Skript zur Vorlesung Modellierungsmethoden der Informatik, H.U. Hoppe, https://www.is.inf.uni-due.de/courses/mod_ws02/skript/kapitel3.pdf, abgerufen am 27.07.2020

Abbildungsverzeichnis

Tabellenverzeichnis

A Anhang: Ergebnisse von Simulationen weiterer Funktionsbausteine

A.1 OR-EFB

Das Fehlerverhalten des OR-EFB entspricht dem des AND-EFB. Daher entsprechen sich die Ergebnisse.

Tab. A. 1 FMEA-Tabelle eines OR-EFB mit zwei Eingangssignalen und einem Ausgangssignal

I1_STAT	I1_VAL_STAT	Ausgewerteter Signalstatus von I1	I2_STAT	I2_VAL_STAT	Ausgewertert Signalstatus von I2	Ausgewerteter Signalstatus von O1	Fehlermode-erkennung
0	0	korrekter 0	0	0	korrekter 0	korrekter 0	entfällt
0	0	korrekter 0	0	1	fehlerhafter 0	fehlerhafter 0	möglich
0	0	korrekter 0	1	0	fehlerhafter 1	fehlerhafter 1	möglich
0	0	korrekter 0	1	1	korrekter 1	korrekter 1	entfällt
0	1	fehlerhafter 0	0	0	korrekter 0	fehlerhafter 0	möglich
0	1	fehlerhafter 0	0	1	fehlerhafter 0	fehlerhafter 0	möglich
0	1	fehlerhafter 0	1	0	fehlerhafter 1	korrekter 1	entfällt
0	1	fehlerhafter 0	1	1	korrekter 1	korrekter 1	entfällt
1	0	fehlerhafter 1	0	0	korrekter 0	fehlerhafter 1	möglich
1	0	fehlerhafter 1	0	1	fehlerhafter 0	korrekter 1	entfällt
1	0	fehlerhafter 1	1	0	fehlerhafter 1	fehlerhafter 1	möglich
1	0	fehlerhafter 1	1	1	korrekter 1	korrekter 1	entfällt
1	1	korrekter 1	0	0	korrekter 0	korrekter 1	entfällt
1	1	korrekter 1	0	1	fehlerhafter 0	korrekter 1	entfällt
1	1	korrekter 1	1	0	fehlerhafter 1	korrekter 1	entfällt
1	1	korrekter 1	1	1	korrekter 1	korrekter 1	entfällt

A.2 Limiter-EFB

Ein Begrenzer (LIMITER-EFB) besitzt ähnlich wie ein Vergleicher (COMP-EFB) drei Eingänge, was die Anzahl der zu betrachtenden Fälle auf 64 erhöht. Der Begrenzer-Block leitet ähnlich wie der COMP-EFB fehlerhafte Signalzustände der Eingangssignale an seinen Ausgang weiter. Das Ausgangsignal erhält den Signalstatus „fehlerhaft“, wenn eines der der Eingangssignale den Signalstatus „fehlerhaft“ hat. Die resultierende FMEA-Tabelle des LIMITER-EFB - bzw. COMP-EFB ist nachfolgend aufgelistet.

Tab. A. 2 FMEA-Tabelle des Limiter/COM-EFB mit drei Eingangssignalen und einem Ausgangssignal

I1_STAT	I1_VAL_STAT	Ausgewerteter Signalstatus von I1	I2_STAT	I2_VAL_STAT	Ausgewerteter Signalstatus von I2	I3_STAT	I3_VAL_STAT	Ausgewerteter Signalstatus von I3	Ausgewerteter Signalstatus von O1	Fehlermode-erkennung
0	0	korrekter 0	0	0	korrekter 0	0	0	korrekter 0	korrekter 0	entfällt
0	0	korrekter 0	0	0	korrekter 0	0	1	fehlerhafter 0	fehlerhafter 0	möglich
0	0	korrekter 0	0	0	korrekter 0	1	0	fehlerhafter 1	fehlerhafter 1	möglich
0	0	korrekter 0	0	0	korrekter 0	1	1	korrekter 1	korrekter 1	entfällt
0	0	korrekter 0	0	1	fehlerhafter 0	0	0	korrekter 0	fehlerhafter 0	möglich
0	0	korrekter 0	0	1	fehlerhafter 0	0	1	fehlerhafter 0	fehlerhafter 0	möglich
0	0	korrekter 0	0	1	fehlerhafter 0	1	0	fehlerhafter 1	korrekter 1	entfällt
0	0	korrekter 0	0	1	fehlerhafter 0	1	1	korrekter 1	korrekter 1	entfällt
0	0	korrekter 0	1	0	fehlerhafter 1	0	0	korrekter 0	fehlerhafter 1	möglich
0	0	korrekter 0	1	0	fehlerhafter 1	0	1	fehlerhafter 0	korrekter 1	entfällt
0	0	korrekter 0	1	0	fehlerhafter 1	1	0	fehlerhafter 1	fehlerhafter 1	möglich
0	0	korrekter 0	1	0	fehlerhafter 1	1	1	korrekter 1	korrekter 1	entfällt
0	0	korrekter 0	1	1	korrekter 1	0	0	korrekter 0	korrekter 1	entfällt
0	0	korrekter 0	1	1	korrekter 1	0	1	fehlerhafter 0	korrekter 1	entfällt
0	0	korrekter 0	1	1	korrekter 1	1	0	fehlerhafter 1	korrekter 1	entfällt
0	0	korrekter 0	1	1	korrekter 1	1	1	korrekter 1	korrekter 1	entfällt
0	1	fehlerhafter 0	0	0	korrekter 0	0	0	korrekter 0	fehlerhafter 0	möglich
0	1	fehlerhafter 0	0	0	korrekter 0	0	1	fehlerhafter 0	fehlerhafter 0	möglich
0	1	fehlerhafter 0	0	0	korrekter 0	1	0	fehlerhafter 1	korrekter 1	entfällt
0	1	fehlerhafter 0	0	0	korrekter 0	1	1	korrekter 1	korrekter 1	entfällt
0	1	fehlerhafter 0	0	1	fehlerhafter 0	0	0	korrekter 0	fehlerhafter 0	möglich
0	1	fehlerhafter 0	0	1	fehlerhafter 0	0	1	fehlerhafter 0	fehlerhafter 0	möglich
0	1	fehlerhafter 0	0	1	fehlerhafter 0	1	0	fehlerhafter 1	korrekter 1	entfällt
0	1	fehlerhafter 0	0	1	fehlerhafter 0	1	1	korrekter 1	korrekter 1	entfällt
0	1	fehlerhafter 0	1	0	fehlerhafter 1	0	0	korrekter 0	korrekter 1	entfällt
0	1	fehlerhafter 0	1	0	fehlerhafter 1	0	1	fehlerhafter 0	korrekter 1	entfällt
0	1	fehlerhafter 0	1	0	fehlerhafter 1	1	0	fehlerhafter 1	korrekter 1	entfällt
0	1	fehlerhafter 0	1	0	fehlerhafter 1	1	1	korrekter 1	korrekter 1	entfällt
0	1	fehlerhafter 0	1	1	korrekter 1	0	0	korrekter 0	korrekter 1	entfällt
0	1	fehlerhafter 0	1	1	korrekter 1	0	1	fehlerhafter 0	korrekter 1	entfällt
0	1	fehlerhafter 0	1	1	korrekter 1	1	0	fehlerhafter 1	korrekter 1	entfällt
0	1	fehlerhafter 0	1	1	korrekter 1	1	1	korrekter 1	korrekter 1	entfällt
1	0	fehlerhafter 1	0	0	korrekter 0	0	0	korrekter 0	fehlerhafter 1	möglich
1	0	fehlerhafter 1	0	0	korrekter 0	0	1	fehlerhafter 0	korrekter 1	entfällt
1	0	fehlerhafter 1	0	0	korrekter 0	1	0	fehlerhafter 1	fehlerhafter 1	möglich
1	0	fehlerhafter 1	0	0	korrekter 0	1	1	korrekter 1	korrekter 1	entfällt
1	0	fehlerhafter 1	0	1	fehlerhafter 0	0	0	korrekter 0	korrekter 1	entfällt
1	0	fehlerhafter 1	0	1	fehlerhafter 0	0	1	fehlerhafter 0	korrekter 1	entfällt
1	0	fehlerhafter 1	0	1	fehlerhafter 0	1	0	fehlerhafter 1	korrekter 1	entfällt
1	0	fehlerhafter 1	0	1	fehlerhafter 0	1	1	korrekter 1	korrekter 1	entfällt
1	0	fehlerhafter 1	1	0	fehlerhafter 1	0	0	korrekter 0	fehlerhafter 1	möglich
1	0	fehlerhafter 1	1	0	fehlerhafter 1	0	1	fehlerhafter 0	korrekter 1	entfällt
1	0	fehlerhafter 1	1	0	fehlerhafter 1	1	0	fehlerhafter 1	fehlerhafter 1	möglich
1	0	fehlerhafter 1	1	0	fehlerhafter 1	1	1	korrekter 1	korrekter 1	entfällt
1	0	fehlerhafter 1	1	1	korrekter 1	0	0	korrekter 0	korrekter 1	entfällt
1	0	fehlerhafter 1	1	1	korrekter 1	0	1	fehlerhafter 0	korrekter 1	entfällt
1	0	fehlerhafter 1	1	1	korrekter 1	1	0	fehlerhafter 1	korrekter 1	entfällt
1	0	fehlerhafter 1	1	1	korrekter 1	1	1	korrekter 1	korrekter 1	entfällt
1	1	korrekter 1	0	0	korrekter 0	0	0	korrekter 0	korrekter 1	entfällt
1	1	korrekter 1	0	0	korrekter 0	0	1	fehlerhafter 0	korrekter 1	entfällt
1	1	korrekter 1	0	0	korrekter 0	1	0	fehlerhafter 1	korrekter 1	entfällt
1	1	korrekter 1	0	0	korrekter 0	1	1	korrekter 1	korrekter 1	entfällt
1	1	korrekter 1	0	1	fehlerhafter 0	0	0	korrekter 0	korrekter 1	entfällt
1	1	korrekter 1	0	1	fehlerhafter 0	0	1	fehlerhafter 0	korrekter 1	entfällt
1	1	korrekter 1	0	1	fehlerhafter 0	1	0	fehlerhafter 1	korrekter 1	entfällt
1	1	korrekter 1	0	1	fehlerhafter 0	1	1	korrekter 1	korrekter 1	entfällt
1	1	korrekter 1	1	0	fehlerhafter 1	0	0	korrekter 0	korrekter 1	entfällt
1	1	korrekter 1	1	0	fehlerhafter 1	0	1	fehlerhafter 0	korrekter 1	entfällt
1	1	korrekter 1	1	0	fehlerhafter 1	1	0	fehlerhafter 1	korrekter 1	entfällt
1	1	korrekter 1	1	0	fehlerhafter 1	1	1	korrekter 1	korrekter 1	entfällt
1	1	korrekter 1	1	1	korrekter 1	0	0	korrekter 0	korrekter 1	entfällt
1	1	korrekter 1	1	1	korrekter 1	0	1	fehlerhafter 0	korrekter 1	entfällt
1	1	korrekter 1	1	1	korrekter 1	1	0	fehlerhafter 1	korrekter 1	entfällt
1	1	korrekter 1	1	1	korrekter 1	1	1	korrekter 1	korrekter 1	entfällt

Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) gGmbH

Schwertnergasse 1
50667 Köln
Telefon +49 221 2068-0
Telefax +49 221 2068-888

Forschungszentrum
Boltzmannstraße 14
85748 Garching b. München
Telefon +49 89 32004-0
Telefax +49 89 32004-300

Kurfürstendamm 200
10719 Berlin
Telefon +49 30 88589-0
Telefax +49 30 88589-111

Theodor-Heuss-Straße 4
38122 Braunschweig
Telefon +49 531 8012-0
Telefax +49 531 8012-200

www.grs.de

ISBN 978-3-947685-85-1